**Vulnerability Assessment in Simulated Phishing Attacks with Fileless Malware**

Yash Khandelwal

Faculty of Computer Science and Engineering, University of New South Wales

# Introduction

Malware refers to any software designed to cause harm to user information, computer systems or networks. Conventional file-based malware attacks deploy actual malicious executables, whereas fileless malware attacks exploit trusted, legitimate processes and inherent operating system utilities to infiltrate and persist covertly, evading detection by signature-based antivirus software and intrusion-detection systems. These sophisticated attacks are engineered to operate within system memory, minimizing their digital footprint and leaving no discernible traces on physical hard drives.

A diagram of a computer program

Description automatically generated

Figure 1. Infection flow of fileless malware (Sudhakar & Kumar, 2020)

PowerShell provides a suitable environment for the deployment of malicious payloads while evading detection. The initial intrusion point within a domain provides adversaries with the potential to infiltrate other network hosts, leading to the propagation of malware, which can be a particularly concerning scenario within enterprise environments.

Following the successful execution of the initial stage through social engineering, this study conducts a reverse shell attack, aiming to initiate a shell session that redirects I/O connections of the target system’s shell to allow remote access for the attacker. In this context, the study investigates the inherent security vulnerabilities within PowerShell and outlines strategies for identifying and mitigating such threats.

# Experimental Setup

To investigate the intricacies of fileless malware and assess the effectiveness of defensive strategies, a controlled environment was established. A virtualised environment established using Hyper-V Manager enabled isolation and easy replication of test environments, and a segmented network configuration within the virtual environment ensured a separate, contained network offering precise control over the flow of traffic and data between different network segments.

# Attack Vectors

Traditional security tools include network firewalls and antivirus software that can prevent the execution of malicious scripts. In this section, we explore some of the strategies employed by cyber adversaries to exploit the digital domain.

## Code Obfuscation

The purpose of obfuscating malicious PowerShell is to create enough difference between the obfuscated file and the original such that the obfuscated file is not detected as malicious. Often, multiple layers of obfuscation become necessary to break malware signature detection. The files used in this study utilise multiple common PowerShell obfuscation techniques, including concatenation, reordering, escaping characters, mixing upper and lowercase characters, random whitespaces and base64 encoding.

The only file that the user interacts with only serves the purpose of downloading another file, *WinSecurityUpdate*, from a webpage.

## Antimalware Scan Interface (AMSI) Bypass

Microsoft AMSI provides defending systems with the capability to inspect all the *un*-obfuscated code executed by scripting engines.

*Figure 2. Disabling AMSI by modifying its internal behaviour* (Hendler, et al., 2020)

The identified technique utilises .NET’s reflection mechanism to set the private static property *asmiInitFailed* to *true*, effectively disabling AMSI by indicating that initialisation has failed, ensuring that no scan is done for the current process.

## Reverse Shell

Reverse shells aim to create shell sessions by having the server connect the client through I/O redirections, enabling attackers to gain remote access to the target system. The listener is a session handler to control a compromised victim that reports back to the command-and-control server on a specified IP and port.

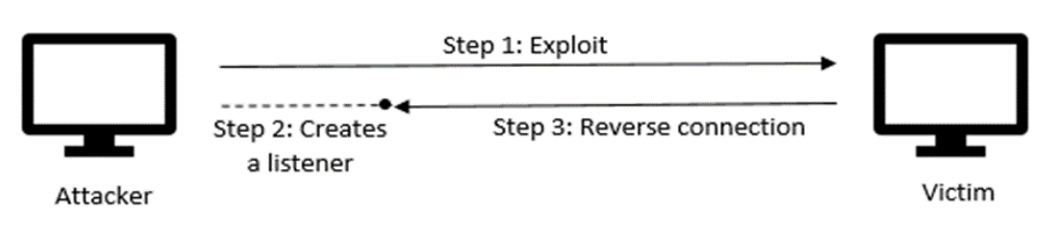


Figure 3. Reverse Shell Attack Mechanism

In the following script, a new TCP client object attempts to connect to an IP address on port 443 (this is a secure port, so connections are disguised as HTTPS traffic to avoid detection by network security measures). The client then sends and receives data from the remote system, becoming vulnerable to remote command execution on the compromised machine. The output of the executed commands is then sent back to the remote system.

*Figure 4. Remote Command Execution Exploit*

The attack opens the door for simple reconnaissance, data theft, or even malware deployment.

# Challenges and Strategies

## Bypassing Firewalls

In my efforts trying to bypass the antimalware detection system, I failed to recognise how the enterprise firewall on my work device would be monitoring and filtering incoming and outgoing network traffic. This was before the virtual environment had been setup and since I had not isolated the testing environment, I got a very interesting message from a security analyst from my company.

A screenshot of a chat

Description automatically generated

Figure 5. Preliminary Reverse Shell Attack detected by Enterprise Firewall

Not only does this stress the importance of isolating a test environment before conducting tests of such nature, it intrigued me how the enterprise firewall overlayed the default security mechanisms on my device. Nevertheless, it was a reminder to be careful in executing scripts, even if I knew they were harmless.

## Persistence

On its own, a reverse shell connection does not offer persistence capabilities. To achieve persistence, it requires being embedded within malware that provides such functionality.

During my exploration of various methods for establishing persistent fileless malware, my investigation led me to Kovter, a click-fraud trojan that utilizes a fileless persistence mechanism through a registry-based attack. The concept involved creating a registry value that remains hidden from regedit and stores the malware executable within the registry. Executing this plan, however, required a significant learning curve beyond the scope of this investigation. Nonetheless, investigating about this technique provided valuable insights into how an attacker could gain long-term control over the victim's system, underscoring the seriousness of such an attack.

## Prevention Strategies and Methodologies for Detection through System Behavioural Monitoring

From a server perspective, it is difficult to block all reverse shell connections when using a networked system. However, there are steps that can be taken to enhance system defences and mitigate risk. These measures involve restricting outgoing connectivity to specific ports and remote IP addresses, setting up a proxy server with limited destinations and more stringent controls, and mitigating vulnerabilities, such as code injection exploits.

Additionally, preventing phishing frauds requires cybersecurity awareness and an understanding of the strategies employed by attackers in social engineering. This study involved the creation of a Microsoft phishing email that was designed to instill a sense of urgency in the victim, making it more likely for them to follow through with the provided instructions. The email could also be accompanied with a malicious helpline intended to persuade the user to comply. One of the challenges in this study involved bypassing Microsoft Defender SmartScreen, which flags custom scripts as unknown as suspicious. While a strategy to bypass this security was investigated, and [exploits](https://gridinsoft.com/blogs/microsoft-smartscreen-bypass-technique/) were readily available, this step could not be implemented due to the study’s constraints. However, with more effort in social engineering, this security measure can be easily circumvented.

One effective strategy for detecting reverse shell connections involves monitoring and analysing packet data traffic between the client system or internal network and external networks. This analysis entails assessing the transmission directions and payload sizes of monitored packets relative to a predefined traffic pattern (Althouse, et al., 2017).

Another strategy involves regular monitoring of open ports on a system to identify unusual or unauthorised services running on non-standard ports. Also, though the attack conducted in this study bypasses this be communicating on a secure port, tracking of well-known protocols (i.e., HTTP, SSH, RDP) on unexpected ports is often a strategy that could help indicate signs of malicious activity.

# Concluding Remarks

The increasing prevalence of non-malware attacks underscores the necessity for security defenders to enhance their detection and prevention strategies. Furthermore, adopting a zero-trust approach, balancing usability, and security, and being prepared with an incident response plan is critical to minimising the scope and impact of these attacks. This mindset requires organizations to assume that no user or system is inherently trusted, enforcing strict access controls, continuous monitoring, and micro-segmentation to limit lateral movement. This approach not only fosters a more resilient security posture but also acknowledges the dynamic and persistent nature of contemporary cyber threats.

# References

Althouse, J. B., Salusky, W. R. & Atkinson, J. S., 2017. Reverse Shell Network Intrusion Detection. *US Patent Application 20170339166.*

Hendler, D., Kels, S. & Rubin, A., 2020. AMSI-Based Detection of Malicious PowerShell Code Using Contextual Embeddings. *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security.*

Sudhakar & Kumar, S., 2020. An emerging threat Fileless malware: a survey and research challenges. *Cybersecurity,* Volume 1.