**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. While file-based malware attacks utilise real malicious executables, fileless malware attacks leverage trusted, legitimate processes and built-in operating system tools to infiltrate and stay undetected from signature-based antivirus software or intrusion-detection systems. These attacks are designed to run in system memory with a small footprint, leaving no artifacts 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 entry point within the domain gives adversaries potential access to other hosts in the network and spread malware, which can especially be detrimental in an enterprise environment.

Once the first stage of the attack is successfully executed 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 so that an attacker can access it remotely. In doing so, the study explores the security vulnerabilities inherent to PowerShell and provides some strategies of identifying and preventing such attacks.

# 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.

## 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.

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 3. Remote Command Execution Exploit*

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

# Challenges

## 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 4. 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. For this to be achieved, it must be the payload of malware which does offer persistence.

A key feature of fileless malware attacks is a persistent ability to spy on the compromised system or dynamically run malicious program in memory. Different strategies have been investigated, including using Poweliks trojans (which use system registries to achieve persistence) and Javascript code added to registry as a form of Windows registry malware attacks. Continuing investigation on these will be a key focus for the upcoming weeks.

## Detection by monitoring behaviour of the system

Another area of focus for this project involves a brief overview of defensive strategies against fileless malware attacks. This requires consideration of two aspects; privilege escalation and monitoring the security events for the program execution. Attackers aim to gain root access to their victim’s machine to take full privilege of PowerShell and hence, several essential features need to be monitored in a comprehensive defence strategy. This includes remote command execution by PowerShell, change of standard user privilege to administrative privilege and programs that may be executing in the main memory and could be considered malicious. Furthermore, it is essential to identify the principal sources of information, such as network traffic, network connections and suspicious modifications to registry keys.

Several questions arise in the detection and collection of suspicious files, such as “Do we scan all text files, scripts and XML files?” or “Do we need a parser to interpret whether an interpreted file has malicious code?” or “Do we block suspicious strings/comments from running in scripts?”. These all need to be considered to balance false positives in early detection systems.

# Concluding Remarks