**Vulnerability Assessment and Advanced Threat Defence in Simulated Phishing Attacks with Fileless Malware**

Yash Khandelwal

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

# Introduction

Malware refers to malicious software or program used to exploit computer systems and gain unauthorised access to a user’s personal information.

Unlike traditional file-based malware attacks, instead of using real malicious executables, these attacks leverage trusted, legitimate processes and built-in operating system tools to attack and stay undetected by signature-based antivirus software or intrusion-detection systems. These attacks do not download malicious files or write any content to disk, but rather exploit application vulnerabilities to inject malicious code directly into main memory. Common applications exploited include the Microsoft office suite, Windows PowerShell or Windows Management Instrumentation.

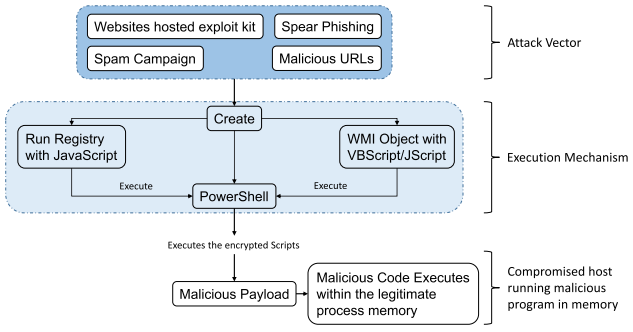


Figure 1. Infection flow of fileless malware [(Source)](https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0043-x)

# 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.

# Strategies Explored so far

## Code Obfuscation and Disabling Antimalware Scan Interface (AMSI)

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*

The script sets the non-public, static field for *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.

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), sends and receives data from the remote system and executes PowerShell commands on the remote system. The output of the executed commands is then sent back to the remote system and the connection is close with the remote system.

*Figure 3. Remote Command Execution Exploit*

There are several applications for the attack above since it enables remote, unauthorised access to the target system. This may range from simple reconnaissance, data theft, or in this case, malware deployment. These will be explored further in the coming weeks.

# Challenges, Lessons And the Path Forward

## 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.

## Payload

This is still an unclear area for my project since I am not sure whether I want the focus of my project to be on the damage of an impact or the attack vectors and vulnerabilities that are exploited by attackers and the challenges in early detection systems and incident response. Further work will need to be done in the coming weeks to clarify the scope of this investigation.

## Developing a persistence mechanism

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 particular 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.