Injecting a PowerShell Trojan Horse via Email

Alex Sheriff   
Group Lead  
*Department of Software Engineering*  
*Monmouth University*West Long Branch, NJ, USA  
s1270704@monmouth.edu

Kimberly Kraemer  
Group Coordinator  
*Department of Computer Science*  
*Monmouth University*   
West Long Branch, NJ, USA  
s1131610@monmouth.edu

*Abstract*— Exploiting the abilities of Windows PowerShell to inject malware quietly and manipulate a user’s system is an increasingly popular way for hackers to infect Windows computers. Trojan horse malware injected in this manner can be written to affect a system in various ways, ranging from displaying a seemingly harmless pop-up message to potentially taking over a user’s entire system or downloading and implementing serious ransomware. To demonstrate the potential power in the ease of creating and injecting malware using this technique, this paper will analyze the process of creating a simple program that displays a number of irritating popups on a victim’s computer screen at start-up, and injecting it through an email attachment sent to a particular target.

Keywords—trojan horse, malware, PowerShell

# Introduction

Malware comes in many forms and can be encountered by anyone with a computer. A trojan horse can do a various amount of damage once the malware gains access to someone’s computer system, but it must first somehow be injected into the system. One rising method of injecting a trojan horse into a computer system is by attaching a Windows PowerShell program into an email attachment that a recipient can download. One 2016 study even found 38% of attacks seen by one IT vendor and its partners involved the use of PowerShell [1]. This type of attack is often classified as fileless malware. Fileless malware uses pre-existing tools on the target’s machine, disguising it as a normally running process while writing to the host’s PC’s RAM and causing problems thereafter until the machine is rebooted [2].

Although this type of malware (like any) has its downsides, it can be extremely powerful because it allows access to the heart of a computer, and it is not usually detected by antivirus software [2]. Because the process happens so discretely, it is not likely to alarm the user that anything is happening, and because PowerShell is such an advanced scripting engine for a Windows machine, malware that uses it to its advantage is capable of doing a lot of damage before a user even notices anything is wrong.

Malware that uses PowerShell can be shaped into a virus, worm, or any other type of malware, but a trojan horse is arguably the most common type of malware that is distributed via an email attachment and targets specific users. A trojan horse is less concerned with replicating itself and more concerned with injecting the malware quietly onto a user’s system by disguising itself as something useful. This paper presents an attempt to create just that: a trojan horse that is sent via an email attachment targeted towards a particular type of subject and disguised to convince that subject to unknowingly install malware which uses Windows PowerShell to its advantage.

# RELATED WORK

This paper is related to trojan horses that use Windows PowerShell to their advantage once downloaded to execute commands discreetly.

One example that demonstrates the power of using of Windows PowerShell in a trojan horse attack is known as the Dimnie attacks. In 2017, researchers from Palo Alto Networks discovered a trojan horse targeted towards developers, specifically those who were present on GitHub [3]. Hackers sent emails to targets that appeared to be requests for development-related work which would result in payment; each email contained a .gz attachment, containing a Word document with hidden malware inside [3]. Once downloaded, if the code could execute, then the malware would run PowerShell commands to reach out to an external server and install Dimnie [3]. Once installed, Dimnie was able to download even more malicious code into the memory of other Windows processes, and various modules got to work on stealing data, keylogging, screen grabbing, and more [3]. There were various types of hacking involved in this process, but in essence it began as a trojan horse scheme, targeting developers and convincing them to download malicious software unknowingly. The use of Windows PowerShell allowed the hackers to inject the extremely powerful Dimnie without raising alarm, since the commands were disguised as normal system functions.

# PROPOSED WORK

The proposed experiment involves creating an email targeted towards people who may be interested in downloading a new holiday cookie recipe. The email will contain an attachment named spritzCookieRecipe. To trick the user into installing the malware, the spritzCookieRecipe program, when installed by the user, will open a webpage that displays a cookie recipe. However, without the user’s knowledge, the program will also run a C++ program that writes a windows command file (.cmd) containing a PowerShell Script and saves it to the startup directory.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PC Number | Trial No. | (y/n) Sent/Received | Issues encountered | Potential Solution(s) |
| 1 | 1 | n | Could not attach .exe to email | Send .exe as Google Drive attachment |
|  | 2 | y | n/a | n/a |
| 2 | 1 | y | Error messages displayed upon trying to run | Build C++ packages into .exe so user doesn’t need to have them installed |
|  | 2 | y | n/a | n/a |
| 3 | 1 | y | n/a | n/a |

Figure 1 outlines the general approach for the trojan horse. It starts with the target, who downloads the email attachment. That attachment does two things: opens a webpage containing a recipe and runs a C++ program that writes a PowerShell script to a file. That PowerShell script will then go on to display a set number of annoying pop-ups at start-up.

While our program only uses PowerShell to display the pop-ups at start-up, the malware is intentionally kept simple. Our goal is to demonstrate that a person could implement PowerShell commands without alerting the user that anything is happening, not to do actual damage to anyone’s computer.

# EXPERIMENTS

To test spritzCookieRecipe.exe, it was sent to 3 test emails and installed it on 3 separate Windows PCs. Figure 2 shows the results of each run-through. Potential solutions were tested in the trial following the detected issue.

PC 1 was the PC that the C++ program was created on. This is why after resolving the email attachment issue, when downloaded and run, the executable did as it was supposed to; that PC already had the necessary packages installed, so it ran the .exe with no problem. However, when tested on PC 2, it became clear that this would be an issue for most PCs. So, the C++ code was rebuilt as a release instead of a Debug and the runtime library was changed to multi-threaded instead of multi-threaded DLL. Once this was adjusted, PCs 2 and 3 had no problems running the executable file, and they all ran the program at startup thereafter.

Although our PowerShell trojan horse executable did as it was supposed to after some adjustment, it should be noted that Windows does display a warning whenever a person tries to download executable files. This warning would not necessarily stop someone from downloading it anyway if the trojan email led them to believe they should. However, in a situation where someone were truly trying to be completely inconspicuous, this would likely not be acceptable, so doing something like injecting the malware into a corrupted word doc like the creators of the Dimnie attacks did may be a better approach than sending an executable file.

# CONCLUSION

Though our program is harmless, with malicious intent one could use PowerShell to truly create a more hidden and harmful piece of malware. With minor tweaks to the spritzCookieRecipe program, and a specific demographic, our malware can become a formidable tool for scamming.   
  
 As an example, we could have spritzCookieRecipe display two pop-ups. The original, which we could have pop-up over a hundred times instead of 6, and a new one. The new pop-up could disguise itself as a virus detection alert, advocating that the user calls “Windows Support” for technical assistance. The phone number listed could be registered on a burner phone to not be associated with the scammer.

If the user calls the phone number for help, the scammer can charge any amount of money to “troubleshoot” and fix the problem. Once the scammer gains remote control of the user's desktop, the malware can be deleted in less than a minute.

The target audience in this scenario would be the elderly, since they would likely be too innocent to realize they are being defrauded.

This simple example demonstrates that even using the core of the spritzCookieRecipe program, with some enhancements a scammer could cause technical or financial damage to unsuspecting individuals.

Executing PowerShell commands via fileless malware can allow an individual to access a victim's system and obtain personal data without that person realizing until it is too late. With these commands typically being housed in email attachments, it only further reinforces the idea of only opening emails and email attachments from sources you trust.

# References

1. T. Greene, "Power Shell is a powerful malware tool," *Network World (Online),* 2016. Available: https://ezproxy.monmouth.edu/login?url=https://www-proquest-com.ezproxy.monmouth.edu/docview/1780625743?accountid=12532.
2. D. Crookes, "Fileless malware," *Web User,* pp. 38-39, 2018. Available: https://ezproxy.monmouth.edu/login?url=https://www-proquest-com.ezproxy.monmouth.edu/docview/2132197114?accountid=12532.
3. L. Constantin, "Open-source developers targeted in sophisticated malware attack," *JavaWorld,* 2017. Available: https://ezproxy.monmouth.edu/login?url=https://www-proquest-com.ezproxy.monmouth.edu/docview/1884289704?accountid=125