TECHNICAL UNIVERSITY OF DUBLIN – BLANCHARDSTOWN CAMPUS

Investigating Low Level Rootkit Stealth Techniques Against Modern Detection Mechanisms

Tytus Kopera
B00140074

4th year

Overview

A low level rootkit for Linux written in C, created with an emphasis on persistence and AV, EDR and user evasion. It would hook function calls to hide itself from basic diagnostics tools such as ls, ps and ss, encrypt itself to evade signature based detection, employ a variety of anti debugging techniques, such as static linking of libraries(usage of musl) and executable bit flipping, and minimizing resource usage so that even the user does not suspect anything running foul. Minimizing resource usage would be achieved through the use of a cryptominer which can have its desired resource usage specified(XMRig is the most obvious choice) and not targeting machines with weaker CPUs. Therefore there should also be a mechanism to check the machine's CPU.

What this rootkit should achieve is to allow a cryptominer(preferably one with variable resource usage) to run silently in a victim system, virtually undetectable by accident and by deliberate act, as well as being persistent enough to survive a system reboot, reinstall and wipe(barring disk formatting, although this can also be bypassed). The cryptocurrency would be sent to a to a randomly created address. All relevant information would be exfiltrated securely to an attacker controlled machine.

Methodology:

I. Perform literature review

    Though research should be implemented cyclically and throughout the project, the initial research focus should cover the widest range of relevant topics in rootkit/malware development, evasion techniques in malware, exfiltration techniques, secure communications, malware exfiltration infrastructure, malware persistence, malware detection techniques.

II. Develop a rootkit

    A. Implement AV/EDR evasion

        The rootkit's main aim is to facilitate its' own existence and the existence of the the cryptominer. It is of utmost importance that any inbuilt or downloaded antivirus or endpoint detection and response technologies do not flag the software as malicious and alert the user. Unseen and undetected is better than detected but deemed harmless, even if wrongly so.

    B. Diagnostic tools evasion

        Diagnostics tools such as "ls" and "ss" on machines which use the GNU toolset are known by most users and versatile. Even if the user or AV software don't suspect anything to be running afoul within their machine, the use of diagnostics and system tools can supply the user with hints or full answers as to what is happening on their system. Therefore it is critical that any command that shows any facet of the software on the system must be modified in such a way that information regarding the rootkit is not written to any output. The easiest way to achieve this would be with function call hooking.

C. Heuristic and signature based analysis evasion

The exploit which will be used for the injection of the payload into the system will be encrypted to bypass any form of signature based detection. The execution patterns or flow of the rookit will be modified or obscured to avoid heuristic based detection.

D. Anti-Debugging and Anti-Sandboxing features

1. Anti-debugging

A plethora of anti-debugging and anti-sandboxing techniques will be used to prevent research and testing of the rootkit in environments made for malware analysis. The executable will be executed with all the flags which inhibit debugging and instead of the GNU C library, musl will be used. This should decrease the size of the binary and working closer to the kernel with less abstractions. Additional measures will include but not be limited to: .symtab stripping, executable bit flipping, hiding FUNC symbols, stripping the binary, entry point obfuscation and binary encryption.

2. Anti-sandboxing

The rootkit will not run on machines with weak hardware, this will partly coincide with virtual machines which don't usually have many cores running. The rookit will check any virtual machine related artifacts in lcspu and check for timing anomalies.

E. Rootkit facilitates the stealthy and persistent mining of cryptominer and invisible exfiltration of tokens

Overall the rootkit must allow for XMRig, the chosen cryptominer, to be hidden while it runs. Additionally, XMRig's resource usage can be configured to "X" cores, potentially making it more difficult to notice. Any data/tokens mined will be sent to a wallet and the relevant data encrypted and transferred to a remote server.

The rootkit must also be persistent enough to remain on the system even when it is deleted, the machine is rebooted and the disk is formatted.

F. Runs on Linux machines with good enough CPU

The scope of this rootkit will be to target Linux machines with CPUs that have above 8 cores and a reasonably high clock speed. This is to minimize the percentage impact that the payload's activity may have on computer resources. Linux machines are also under-represented in the home PC market and over-represented in the commercial server market. This dichotomy may create an interesting environment to research

G. Research and implement additional techniques not outlined by this brief

Given the topic and my relative inexperience with cybersecurity, malware development and malware research, it should be expected that I, as a student and researcher, should continually research these relevant topics in order to make adjustment and improvements in my implementation of this project.

III. Implement and test the payload in isolation and as part of an exploit

A. Analyze system usage

Evaluate the strain which the rootkit has on the machine. CPU and RAM usage should be checked and evaluated. Ask the question is it suspiciously high.

B. Analyze machine with the help of various antivirus software

Download several free and non-free anti-malware software such as Malwarebytes, Bitdefender, Norton AntiVirus Plus and use these packages to try and locate, quarantine and remove the malware. This must be done throughout several power cycles to establish the persistence of the malware.

C. Analyze the machine forensically

Search the computer of any waste files, files in /tmp left behind by the malware etc.

D. Analyze network usage

Tools such as netstat,wireshark and ss will be used to investigate any malware activity.

E. Research and implement other analysis techniques

Other analysis techniques will be researched throughout the implementation of the project and if any techniques were to become relevant to the topic, they will be implemented aswell.

F. Evaluate the results of analysis

IV. Tweak rootkit and repeat process

After determining the efficacy of the rootkit against the malware analysis, the rootkit will be altered to bypass the tested analysis methods.

Optional additional features:

- Integrating this rootkit into metasploit as a payload