Understanding, Detecting, & Preventing Modern Linux Rootkits

By Ben Stewart

The MITRE ATT&CK[1] matrix is a great tool for visualizing the real-world tactics and techniques of a malicious attacker looking to compromise an enterprise system. Although an attacker's end goal(s) can vary, the objectives usually require the ability to stay under the radar while they explore, setup a C&C, exfiltrate data, etc.

Rootkits 101

Although there are multiple ways to achieve advanced persistence and evade defenses, one of the most dangerous and devastating can be through the use of rootkits. Rootkits are post exploitation tools that can be ran stealthily on a target, often with the highest system privileges. They can be used to hide/modify programs, services, files, network information, other system components and there have been documented malware cases of them being active on victim systems for years and decades[2].

As with any adversarial tool, rootkits continue to evolve as defenses improve. Rootkits have targeted most major OSes and potentially due to the open source nature of Linux, we've seen them improve rather quickly. But why should we care? Because Linux is everywhere! As one of the most important pieces of computer software in the world, you probably knowingly or unknowingly interact it with it daily[3].

Now you realize why it's such an attractive target for attackers and you may be itching to understand how rootkits work. That's great, however, the truth is that rootkits can operate in a multitude of different ways. Rather than jumping right in, it might be best to start off with understanding how tools designed to detect rootkits work. Then we can extrapolate that functionality into our definition of how the modern Linux rootkit operates.

Detection

For Linux systems, two of the most popular, free rootkit detection tools are chrootkit and rkhunter. Both of these tools are executed locally and check for signs of a rootkit. Let's dive a little deeper into the first tool, chrootkit. This program relies primarily on a shell script to check system binaries for rootkit modifications. Perfect!

Dissecting the ~3,000 lines of code, we notice the core functionality of the script involves traversing the system searching for specific indicators of compromise signaling the potential presence of a rootkit. These indicators vary from a suspicious directory or file tied to a rootkit breed to checking specific commands/utilities. One specific example is checking the SSH daemon (sshd) against a regular expression value filled with different labels known to be used by certain rootkits. This trend is repeated across other commands (ls, netstat, ps, etc.) as well as for log files, other functions, and services.

Based on this brief analysis, it's clear that a rootkit is an incredibly useful tool for an attacker since it can be customized to accomplish a wide range of objectives and can modify system behavior to stay

under the radar. Although, this tool relies on a relatively naïve approach for detection, it can still be quite effective in detecting many "off the shelf" types of rootkits. But overall, it's important to remember that detection is difficult because a rootkit can corrupt the software intended to find it.

Damage

Now if we flip our mindset from defense to offense, we can now comprehend the scale of damage that can be done with a rootkit. Common objectives of rootkits include:

- Bypassing standard authentication and authorization mechanisms
- Vertical privilege escalation to root and/or system
- Concealing other malware by hiding processes, files, and directories
- Disabling detection mechanisms and other defense evasion techniques
- Pretty much anything your evil little heart desires

History

Keep in mind, this isn't an overnight phenomenon. Linux rootkits have been evolving over the years to keep pace with detection mechanisms. For example, some of the earliest Linux rootkits weren't as much of a rootkit as they were just series of backdoored commands that would prevent a system administrator from detecting a malicious process, shell, file, etc. However, implementing this was extremely tedious since you had to assume what tools/processes a sys Admin would use and secondly, it left a ton of forensic data on the target. Rootkits then evolved to target the order of precedence in dynamic linking libraries using ld_preload. Basically, in a nutshell, you can change what systems calls are doing by hijacking them and once again you blind the system administrator. This also leaves quite a big footprint and is easy to detect. How easy to detect? Referencing our chrootkit script, we can see that it's pretty much a one line conditional to detect these somewhat naïve rootkits.

But much to our dismay as security professionals, innovation hasn't stalled, and we've recently seen an uptick in malicious kernel module rootkits or LKM (Loadable Kernel Module) rootkits. Essentially, an attacker is able to insert a malicious kernel module and they can pretty much do whatever they want. Common activities include hiding processes, files, or the module itself but an attacker is only limited by their imagination since they can operate on behalf of the system.

Classifications

In the previous section, we mentioned a few different rootkits. The first two historical examples operate in user mode whereas the LKM rootkit is much more powerful and can operate on behalf of the system. Below we define those rootkit classifications:

- User mode Rootkits – Ring 3
    - Most common and easiest to implement
    - Hooking and/or injection behavior
    - Usually noisy and easy to detect

- Kernel mode Rootkits – Ring 0
    - Can run with highest OS privileges
    - Targets the kernel and associated drivers
    - Usually cloaked and harder to detect
- Hypervisor Rootkits – Ring -1
    - Primarily still in academia as proofs of concept
    - Result of an exploit in the hardware virtualization
    - Able to intercept hardware calls made by the OS
- Firmware & Hardware
    - A persistent malware image hidden hardware
    - Can be the result of a commercial organization (e.g. LoJack[4])
    - Extremely difficult to analyze or detect

Defense/Mitigation

As mentioned previously, detecting the presence of a rootkit and then the subsequent removal can be challenging (and even more challenging with kernel level rootkits). Since the OS has been subverted it can be a challenge to determine what can be trusted and what can't be.

However, it is not impossible! Most commercial based tools rely on detection mechanisms that are behavior or signature based. Essentially by flagging rootkit-like behavior or known malicious indicators, they are able to detect commonly identified rootkits. More advanced techniques include additional forms of integrity checking, booting from an alternative trusted medium, and analyzing memory dumps.

There are a few steps you can take today to protect yourself from malicious rootkits. First, it's important to always harden and audit your Linux systems following security best practices. Taking a defense in depth approach and following the principle of least privilege can greatly help to reduce the potential attack surface. Secondly, use a malware scanner (rkhunter, chrootkit, commercial option) with an automated security audit tool, like Lynis[5].

There is no one silver bullet to detecting and preventing rootkits. The defense strategy above provides a good amount of redundancy and can be used as a baseline but it's important that your defense doesn't remain static. Rootkits are always evolving and some newer varieties even attempt to detect and compromise scanners like chrootkit.

The evils of Linux rootkits are clear but taking a holistic system view no security and following certain best practices can go a long way to protect your systems, organizations, and reputation.

References:

- [1] https://attack.mitre.org/
- [2] https://www.theregister.co.uk/2020/04/07/winnti_linux_hacking/
- [3] https://www.wired.com/2016/08/linux-took-web-now-taking-world/
- [4] https://en.wikipedia.org/wiki/Absolute_Home_%26_Office
- [5] https://github.com/CISOfy/lynis

- https://github.com/Magentron/chkrootkit/blob/master/chkrootkit
- https://en.wikipedia.org/wiki/Rootkit
- Horsepill Rootkit