



Interested in learning
more about security?

SANS Institute InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Immutability Disrupts the Linux Kill Chain

New exploits aimed at Linux systems are able to succeed by achieving root access to the OS. But what if you could lock down the OS and enforce security policies from outside of it? This Spotlight Paper explores the concept of immutability as a way of interdicting the Linux kill chain.

Copyright SANS Institute
Author Retains Full Rights

Immutability Disrupts the Linux Kill Chain

Written by **Hal Pomeranz**

February 2018

Sponsored by:

Immutable Systems

Regardless of the attack vector, successful Linux intrusions share common characteristics. The attacker adds malicious software to the system and executes it. Privilege escalation is the next step, allowing the attacker to modify system files and achieve persistence. Rootkits may then be installed onto the system to hide the attackers' presence.

Targeting these common tactics presents an opportunity to prevent or degrade Linux exploits, even without prior knowledge of the specific exploit used. However, typical Linux security solutions provide incomplete coverage or may provide notification only after a successful attack. Many solutions are difficult to administer, particularly across a large number of systems. And if attackers manage to achieve "superuser" privileges, they can disable or circumvent any security controls in the operating system.

Exploits, Defenses and Limitations

The recent waves of Apache Struts vulnerabilities provide a good example of the early stages of a Linux intrusion. A bug in the Jakarta multipart parser (CVE-2017-5638) allowed attackers to run arbitrary commands with the privileges of the web server process. Typical exploits downloaded executables with **wget** and then executed them. Malicious executables might join the compromised machine to a botnet, or perhaps give the attacker a privilege escalation path to superuser access.



Regardless of the malicious payload, the exploit is taking actions that are abnormal for a Linux server (as shown in Table 1).

No system should be calling out to an external server, pulling down new executables and writing them into the local file system. The executable has to be written to a directory where the web server process has access, which would typically not be one of the normal directories where executables reside. The system should not be running executables from nonstandard locations.

Egress filtering via a host- or network-based firewall could prevent the system from downloading the exploit. But the difficulty of managing firewall policy across a large number of distributed server instances often means that egress filters are either non-existent or incompletely implemented, so that the attackers can achieve their objective.

Properly deployed file-integrity monitoring software could inform the system owner of the new executable, but only *after* the attacker has succeeded. SELinux could prevent the new executable from being written to disk and/or being executed. However, the vast majority of organizations disable SELinux due to difficulty of administration.

Table 1. Exploit Actions, Defenses and Limitations

| Exploit Action | Typical Defense | Limitations |
|--|------------------------------------|---|
| Download executable from external site | Egress filtering via firewall | Difficult to manage firewall policies |
| Add executables to file system | File-integrity monitoring software | Reports <i>after</i> attacker is successful |
| Execute from writable directories | SELinux and AppArmor | Difficult to administer |

Privilege Escalation, Defenses and Limitations

With the ability to run arbitrary code on a Linux system, the attacker's next move is to achieve superuser access. Exploits such as the recent Stack Clash vulnerabilities (e.g., CVE-2017-1000366) coerce a legitimate set-UID executable into executing the attacker's malicious code. Vulnerabilities such as the recent `waitid()` exploit (CVE-2017-5123) write directly to kernel memory to change the effective UID of the attacker's process.

For these exploits to work, abnormal conditions are being created in the operating system (as shown in Table 2).

Stack Clash causes stack and heap memory to collide in the target process—a state that the operating system tries to prevent. The `waitid()` exploit changes the effective UID of the attacker's process in memory without using the typical Linux `setuid()` mechanism. The end result of these and many other privilege escalation exploits is to give the attacker a superuser command shell. Superuser shells should be rare enough that auditing or even selectively preventing them would not be a burden.

In-memory "guard pages" are designed to prevent stack and heap memory regions from overlapping. However, the recent Stack Clash vulnerabilities demonstrated weaknesses in the existing Linux guard page implementation. While these vulnerabilities caused developers to make changes in the Linux kernel, it seems probable that additional vulnerabilities will be found.

Table 2. Privilege Escalation, Defenses and Limitations

| Exploit Action | Typical Defense | Limitations |
|--|-------------------------|------------------------------|
| Collide stack and heap memory | In-memory "guard pages" | Weaknesses in implementation |
| Effective UID change without <code>setuid()</code> | None | N/A |
| Spawn new superuser process | SELinux and AppArmor | Difficult to administer |

SELinux may prevent the attacker’s superuser processes from running. However, as noted earlier, many organizations lack the ability to properly deploy and manage SELinux.

If attackers achieve interactive superuser access, they can disable any security controls in the operating system. This includes reconfiguring or turning off host-based firewalls used for egress filtering, subverting file-integrity monitoring software and disabling SELinux controls. This is a fundamental failing in the Linux security model—nothing in the operating system can block the superuser.

Persistence, Defenses and Limitations

Once attackers have achieved superuser access, it is straightforward for them to modify system startup scripts, scheduled tasks or other configuration files so that their malware will persist across system reboots. The attackers may deploy a rootkit to hide their modifications from the system owner. Modern loadable kernel module (LKM) rootkits such as Phalanx2, KBEAST and Reptile are widely available. (See Table 3.)

Unfortunately, at the point the attacker has achieved superuser privileges, the typical defense against unauthorized configuration file changes—file-integrity monitoring—can be subverted or disabled by the attacker with privileged access. Similarly, the attacker can override attempts by the operating system to prevent the introduction of malicious kernel modules.

Table 3. Defenses Thwarted by Superuser Access

| Exploit Action | Typical Defense | Limitations |
|----------------------------|---|-----------------------|
| Modify system config files | File-integrity monitoring | Superuser can disable |
| Kernel hooking | Policy prevents loading malicious modules | Superuser can disable |

Interdiction Through Immutability

Modern, containerized Linux deployments have well-defined behaviors. Large portions of the file system do not change during normal operation. Process UID changes are atypical. Kernel system calls and IDT hooks should not happen. Stack and heap memory should not collide. In other words, large portions of the operating system can be considered “immutable” under normal workloads.

A security policy solution that enforces this immutability blocks the abnormal behaviors of exploits and rootkits—regardless of the specific exploit used—while allowing normal system processes to operate. For example, the following policies would block or degrade the types of exploits covered in this paper:

- Prevent network access to unauthorized sites
- Block unauthorized file system changes
- Prevent unexpected process UID changes
- Monitor superuser processes for suspicious behaviors
- Block system call hooks and IDT hooks used by LKM rootkits
- Prevent collision of process memory regions

Looking Forward

The trick is stopping an attacker with privileged access from subverting these policies. An ideal solution would stand outside the operating system, where it would be invisible and unreachable to an attacker inside the OS.

With increasing numbers of virtualized Linux deployments comes the opportunity to implement security controls outside of the operating system. A hypervisor with security capabilities would be able to interdict memory, file system and network access, applying security policy controls to render critical portions of the operating system immutable. This level of immutability would prevent or seriously degrade typical Linux exploits—regardless of the actual exploit being attempted. Furthermore, the security controls would be invisible to attackers and exploits operating in the Linux OS and could not be subverted from within the OS.

About the Author

Hal Pomeranz is a SANS faculty fellow and the author/instructor for the Securing Linux/Unix (SEC506) course. He is an independent digital forensic investigator who has consulted on cases ranging from intellectual property theft, to employee sabotage, to organized cyber crime and malicious software infrastructures. In the SANS DFIR curriculum, Hal teaches Advanced Digital Forensics, Incident Response, and Threat Hunting (FOR508), Advanced Network Forensics and Analysis (FOR572), Mac Forensics Analysis (FOR518), and Reverse-Engineering Malware: Malware Analysis Tools and Techniques (FOR610).

Sponsor

SANS would like to thank this spotlight paper's sponsor:





Upcoming SANS Training

[Click Here for a full list of all Upcoming SANS Events by Location](#)

| | | | |
|--|----------------------|-----------------------------|------------|
| SANS Paris June 2018 | Paris, FR | Jun 25, 2018 - Jun 30, 2018 | Live Event |
| SANS Minneapolis 2018 | Minneapolis, MNUS | Jun 25, 2018 - Jun 30, 2018 | Live Event |
| SANS Vancouver 2018 | Vancouver, BCCA | Jun 25, 2018 - Jun 30, 2018 | Live Event |
| SANS London July 2018 | London, GB | Jul 02, 2018 - Jul 07, 2018 | Live Event |
| SANS Cyber Defence Singapore 2018 | Singapore, SG | Jul 09, 2018 - Jul 14, 2018 | Live Event |
| SANS Charlotte 2018 | Charlotte, NCUS | Jul 09, 2018 - Jul 14, 2018 | Live Event |
| SANSFIRE 2018 | Washington, DCUS | Jul 14, 2018 - Jul 21, 2018 | Live Event |
| SANS Cyber Defence Bangalore 2018 | Bangalore, IN | Jul 16, 2018 - Jul 28, 2018 | Live Event |
| SANS Pen Test Berlin 2018 | Berlin, DE | Jul 23, 2018 - Jul 28, 2018 | Live Event |
| SANS Riyadh July 2018 | Riyadh, SA | Jul 28, 2018 - Aug 02, 2018 | Live Event |
| Security Operations Summit & Training 2018 | New Orleans, LAUS | Jul 30, 2018 - Aug 06, 2018 | Live Event |
| SANS Pittsburgh 2018 | Pittsburgh, PAUS | Jul 30, 2018 - Aug 04, 2018 | Live Event |
| SANS San Antonio 2018 | San Antonio, TXUS | Aug 06, 2018 - Aug 11, 2018 | Live Event |
| SANS August Sydney 2018 | Sydney, AU | Aug 06, 2018 - Aug 25, 2018 | Live Event |
| SANS Boston Summer 2018 | Boston, MAUS | Aug 06, 2018 - Aug 11, 2018 | Live Event |
| Security Awareness Summit & Training 2018 | Charleston, SCUS | Aug 06, 2018 - Aug 15, 2018 | Live Event |
| SANS Hyderabad 2018 | Hyderabad, IN | Aug 06, 2018 - Aug 11, 2018 | Live Event |
| SANS New York City Summer 2018 | New York City, NYUS | Aug 13, 2018 - Aug 18, 2018 | Live Event |
| SANS Northern Virginia- Alexandria 2018 | Alexandria, VAUS | Aug 13, 2018 - Aug 18, 2018 | Live Event |
| SANS Krakow 2018 | Krakow, PL | Aug 20, 2018 - Aug 25, 2018 | Live Event |
| SANS Chicago 2018 | Chicago, ILUS | Aug 20, 2018 - Aug 25, 2018 | Live Event |
| Data Breach Summit & Training 2018 | New York City, NYUS | Aug 20, 2018 - Aug 27, 2018 | Live Event |
| SANS Prague 2018 | Prague, CZ | Aug 20, 2018 - Aug 25, 2018 | Live Event |
| SANS Virginia Beach 2018 | Virginia Beach, VAUS | Aug 20, 2018 - Aug 31, 2018 | Live Event |
| SANS San Francisco Summer 2018 | San Francisco, CAUS | Aug 26, 2018 - Aug 31, 2018 | Live Event |
| SANS Copenhagen August 2018 | Copenhagen, DK | Aug 27, 2018 - Sep 01, 2018 | Live Event |
| SANS SEC504 @ Bangalore 2018 | Bangalore, IN | Aug 27, 2018 - Sep 01, 2018 | Live Event |
| SANS Wellington 2018 | Wellington, NZ | Sep 03, 2018 - Sep 08, 2018 | Live Event |
| SANS Amsterdam September 2018 | Amsterdam, NL | Sep 03, 2018 - Sep 08, 2018 | Live Event |
| SANS Tokyo Autumn 2018 | Tokyo, JP | Sep 03, 2018 - Sep 15, 2018 | Live Event |
| SANS Tampa-Clearwater 2018 | Tampa, FLUS | Sep 04, 2018 - Sep 09, 2018 | Live Event |
| SANS MGT516 Beta One 2018 | Arlington, VAUS | Sep 04, 2018 - Sep 08, 2018 | Live Event |
| SANS Cyber Defence Canberra 2018 | OnlineAU | Jun 25, 2018 - Jul 07, 2018 | Live Event |
| SANS OnDemand | Books & MP3s OnlyUS | Anytime | Self Paced |