Sandfly Security™

Protecting Linux® from SSH Key Risks

2025-01-10

Secure Shell (SSH) key security is a major risk for Linux systems. Stolen, orphaned, weak, or misconfigured keys can lead to immediate compromise and lateral movement. Attackers can also insert backdoor keys onto systems without notice. Below, we will highlight how Sandfly uses multiple approaches to identify SSH key risks, track their usage, and help identify threats they pose on a network.

The Risk of SSH

SSH is the de facto method to access Linux systems for administration. While SSH's encryption and authentication is a notable improvement over legacy unencrypted protocols, it has brought about new risks at the same time such as:

1. SSH private keys can be stolen and used for lateral movement.
2. SSH *authorized_keys* often have stale entries and can allow unknown access.
3. Attackers can insert backdoor keys that can exist for a long time without notice.
4. It is difficult to identify and remove known bad keys at scale.
5. SSH can forward network traffic bypassing firewalls and security segmentation.

These risks need to be addressed with specific solutions, and those solutions need to work across all Linux hosts whether in the cloud, on-prem, servers or embedded systems. Sandfly is designed to do just that.

Sandfly's SSH Security Strategy

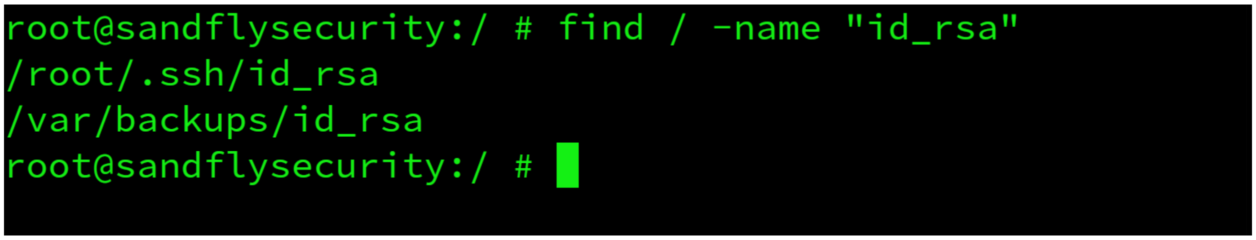
Sandfly incorporates multiple methods to deal with SSH security risks outlined above. In particular:

1. Tracking SSH keys by user, host, and time with our SSH Hunter.
2. Reporting on creation of new keys.
3. SSH Security Zones to alert if new keys show up in unauthorized areas.
4. Policy checks to find suspicious, malicious, or unusual keys.
5. Hunting for unencrypted private keys which can lead to immediate compromise.
6. Banning keys so they can be quickly identified and removed from hosts.
7. Monitoring SSH for risks such as traffic forwarding.

SSH Key Theft Attack Pattern

A very common SSH attack pattern is private key theft. Attackers will break into a host through whatever method gets them onto it. At that point, they will run simple commands to scour the hard drive to find private SSH keys (often unencrypted). If the key is encrypted, they will download it and do a brute-force attack with a password cracker like *Hashcat*.

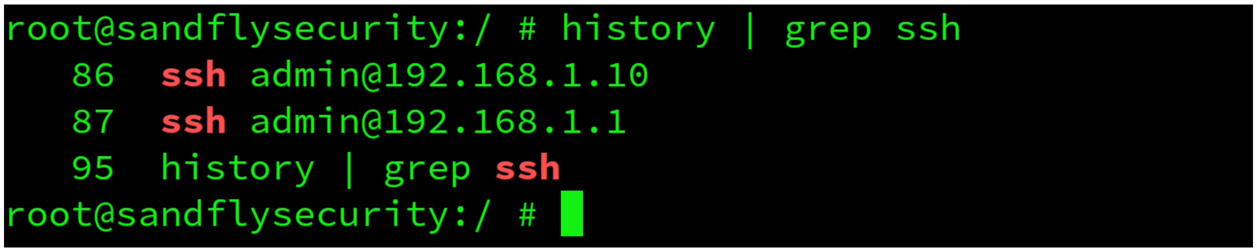
Below, we see a simple command run after compromising the *root* user that will show likely private keys for all users. Here keys for the *root* and *backups* user have been identified.



```
root@sandflysecurity:/ # find / -name "id_rsa"
/root/.ssh/id_rsa
/var/backups/id_rsa
root@sandflysecurity:/ #
```

Figure 1: SSH private key discovery with simple find command as root user.

Once keys are obtained, attackers will use the username of the owner along with connection data they grab from user's SSH *known_hosts* files, command history, login history, user ssh config files, or system connection logs to find next systems to target. For example, below we see a simple search of the compromised user's history shows systems they connected to and usernames used for login:



```
root@sandflysecurity:/ # history | grep ssh
   86  ssh admin@192.168.1.10
   87  ssh admin@192.168.1.1
   95  history | grep ssh
root@sandflysecurity:/ #
```

Figure 2: Attacker uses history command to see what systems to attack with stolen SSH key.

If successful, attackers utilize the new credentials to move. They are protected by the encryption of SSH and the legitimate login means they are extremely difficult to spot.

As the above process repeats, attackers will have a large number of private keys and credentials to access even more hosts. If they don't rock the boat too much, they are unlikely to be noticed without a deliberate search.

SSH Hunter

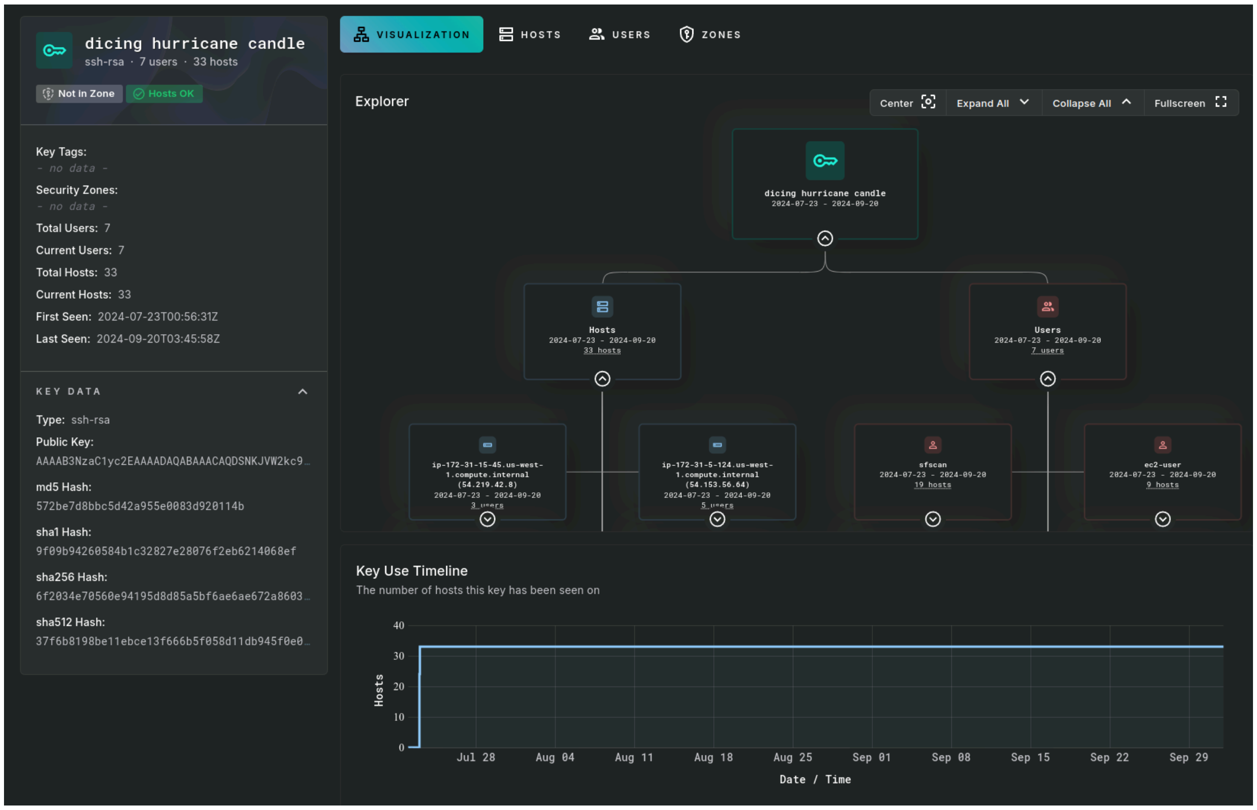
Sandfly has a specifically designed feature to counter SSH key risks like the above. We call this feature SSH Hunter.

The SSH Hunter collects information on the users, their public SSH keys, and the hosts the keys are seen on. We take that information and provide the following:

- SSH Security Zones to build protected areas where unauthorized keys cannot move.
- A list of all usernames and what keys they are using and where they are being used.
- A view into all keys, hosts, and users and where they have access.
- Key use timelines to find new or old keys that are moving across the network.
- Ability to ban a key making it immediately visible on any host where it is used.
- Cross-platform capability that works on any system Sandfly can monitor.

Once installed, Sandfly will track SSH keys and will build an ongoing timeline of where they are, who is using them, and where they are going. This means we also track a key if it returns again at a future date even if security teams thought it was eliminated (e.g. restored from a backup).

SSH Hunter is the heart of Sandfly's SSH monitoring effort. It gives security teams immediate knowledge about the state of SSH keys and where they are being used.



dicing hurricane candle
ssh-rsa · 7 users · 33 hosts

Key Tags:
- no data -

Security Zones:
- no data -

Total Users: 7
Current Users: 7
Total Hosts: 33
Current Hosts: 33
First Seen: 2024-07-23T00:56:31Z
Last Seen: 2024-09-20T03:45:58Z

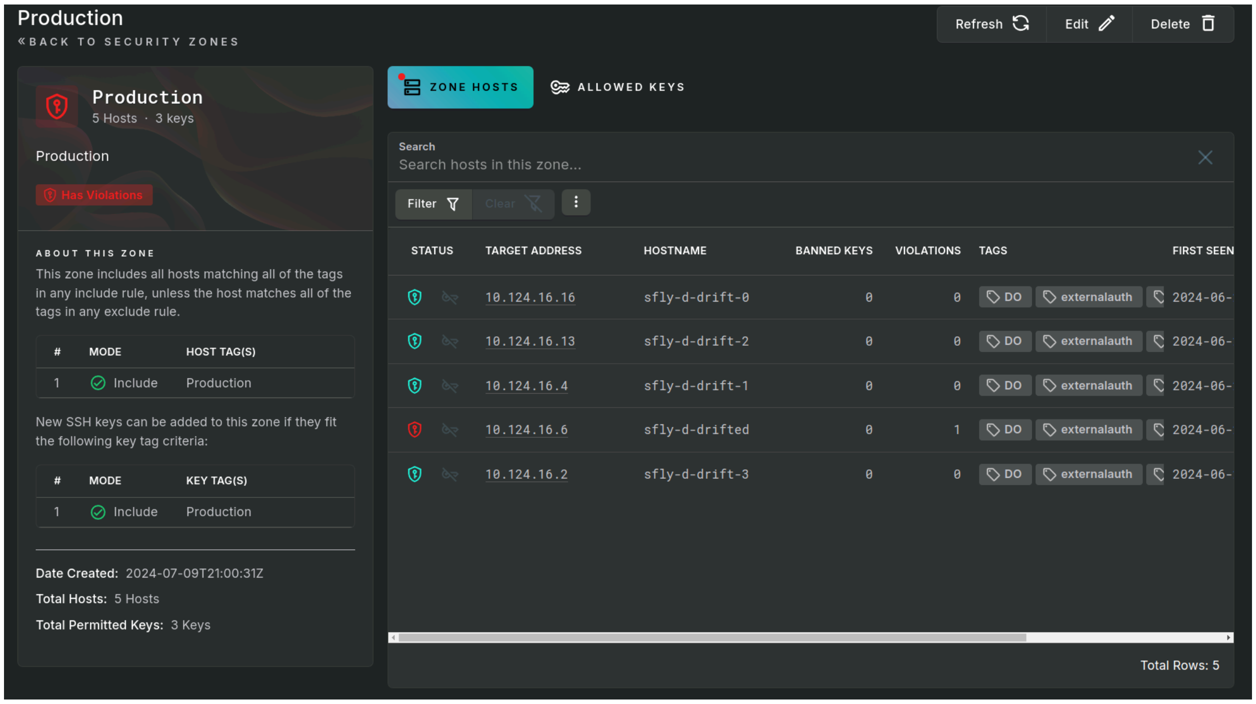
KEY DATA

Type: ssh-rsa
Public Key: AAAAB3NzaC1yc2EAAAADAQABAAACAQDSNKJVW2kc9…
md5 Hash: 572be7d8bbc5d42a955e0083d920114b
sha1 Hash: 9f09b94260584b1c32827e28076f2eb6214068ef
sha256 Hash: 6f2034e70560e94195d8d85a5bf6ae6ae672a8603…
sha512 Hash: 37f6b8198be11ebce13f666b5f058d11db945f0e0…

Figure 3: SSH Hunter viewing an SSH key and what hosts and users it belongs to.

SSH Security Zones

For further key control, Sandfly has a feature called SSH Security Zones. A security zone is a group of tagged hosts that form a protected area. Any key that is not authorized to run inside the zone is flagged as a violation.



Production
5 Hosts · 3 keys

Production

Has Violations

ABOUT THIS ZONE

This zone includes all hosts matching all of the tags in any include rule, unless the host matches all of the tags in any exclude rule.

#	MODE	HOST TAG(S)
1	Include	Production

New SSH keys can be added to this zone if they fit the following key tag criteria:

#	MODE	KEY TAG(S)
1	Include	Production

Date Created: 2024-07-09T21:00:31Z
Total Hosts: 5 Hosts
Total Permitted Keys: 3 Keys

STATUS	TARGET ADDRESS	HOSTNAME	BANNED KEYS	VIOLATIONS	TAGS	FIRST SEEN
	10.124.16.16	sfly-d-drift-0	0	0	DO externalauth	2024-06-
	10.124.16.13	sfly-d-drift-2	0	0	DO externalauth	2024-06-
	10.124.16.4	sfly-d-drift-1	0	0	DO externalauth	2024-06-
	10.124.16.6	sfly-d-drifted	0	1	DO externalauth	2024-06-
	10.124.16.2	sfly-d-drift-3	0	0	DO externalauth	2024-06-

Total Rows: 5

Figure 4: SSH Security Zone violation in a production environment.

For instance, customers can have a SSH Security Zone for “Production.” Inside this zone they tag all hosts they want with the production tag. They can then select what keys are allowed to operate inside the zone. Any key introduced into the zone that is not tagged correctly will generate an unauthorized alert.

This feature is particularly powerful for organizations that want to ensure only certain keys and users can access critical systems or parts of their network. New keys added by accident (e.g. a development key gets pushed to production), or keys added maliciously (e.g. a backdoor), are instantly spotted with the security zone approach.

Unencrypted SSH Private Key Detection

In league with the above, Sandfly recognizes that unencrypted SSH private keys are a major threat. As discussed, once an attacker gains access to a Linux system, they will often sweep the host for SSH keys. Any that are unencrypted give them immediate access to further systems.

Sandfly will check user's directories for unencrypted private keys. We also check high-risk directories such as */tmp* and */dev/shm* ramdisk for any kind of private key which may be present either from a configuration mistake, or sometimes by attackers as they exfiltrate data.

We also have the ability to search an entire file system for unencrypted private keys if desired to be sure this risk is not present anywhere on a host.

Encrypted SSH Private Key Detection

In addition to finding unencrypted SSH keys, Sandfly can also find encrypted versions. Encrypted keys represent a lower risk than unencrypted, but can still be stolen and often brute forced due to bad password selection by users. Organizations that want to be sure no private keys are left exposed on critical systems can use Sandfly to enforce this mandate.

SSH Policy and Key Validation Features

In addition to our SSH Hunter and Security Zone features, we have additional capabilities to audit SSH. These features are encompassed in our policy and threat hunting checks, some of which are detailed here.

SSH Authorized Key File Checks

We pay special attention to users' *authorized_keys* files with a variety of checks such as:

- Presence of *authorized_keys2* which is often done to evade detection.
- Banned key checks.
- *authorized_keys* created or modified within the last X hours (as defined by the user).
- *authorized_keys* file set as immutable (for malware for persistence).
- User defined matching on any element of the key (e.g. certain comments or key types).
- Options that enable port forwarding or suspicious activity.
- Default or inactive users with *authorized_keys* file present.
- Weak or malformed keys (e.g. old RSA 1024 bit keys in use).
- Excessive keys for a user (e.g. a single user has 10+ keys).
- Duplicate keys that allow SSH access.

Checking for *authorized_keys* file anomalies like the above can detect a large variety of intruders or misconfiguration problems that may lead to compromise.

SSH Known Hosts File Checks

Sandfly also analyzes *known_hosts* data for all users on a system. For alerting, we currently can do the following:

- The *root* user has a *known_hosts* (someone logged in as root and then went elsewhere).
- A *known_hosts* file was modified within the last 24 hours (or user defined).
- A default user (e.g. *uucp, php, lp*) has a *known_hosts* file (why are they using SSH?).
- A *known_hosts* file has an entry attribute that matches a value the user wants to look for (e.g. this user connected to a particular host in the past).

This data can help corroborate suspicious activity happening on a user's account involving SSH.

SSH Server Checks

We have further policy checks that can be enabled to search for risky or dangerous SSH server configurations as determined by the customer such as:

- Forwarding enabled.
- Password authentication enabled.
- *root* login enabled.
- Incorrect or dangerous permissions on critical files such as server keys.
- User sessions currently forwarding traffic with their SSH client.

SIEM Integrations

All SSH key and security zone information is available in our SIEM Integrations (e.g. Splunk, Elasticsearch, and Microsoft Sentinel). The data is also available with our REST API for custom analysis.

Password Auditor

While unrelated to SSH directly, Sandfly also includes a password auditor to find users with common and bad passwords. As many systems allow SSH with either SSH keys or passwords, finding users with weak passwords is still a priority. Sandfly's password auditor can also be customized to search for passwords security teams do not want to be used on the network. This can help locate notorious shared or default passwords found on many systems, including embedded systems and appliances.

You can read more about our agentless password auditor in our Whitepaper section here:

[Sandfly Agentless Password Auditing]

Secure and Track SSH Keys

SSH key theft is a major risk to Linux deployments. We encourage customers to keep a vigilant watch on these critical assets. Sandfly is here to help bring transparency and visibility into SSH.

Please reach out to us if you have any questions or comments. We are happy to discuss your requirements or provide a full product demonstration of our unique and powerful agentless SSH key tracking capabilities.

www.sandflysecurity.com