White Paper

Securing Linux

Against Negligent or Malicious Administrators

Background

Linux monitoring is deceptively difficult. The most common tools for performing monitoring - the Linux audit system, log journals and syslog sources - are all, at best, standardized by Linux distribution, and at worst, unique per host in an enterprise environment.

File-based logging can be spoofed by intruders, while kernel-based subsystems have performance issues. Many hosts will often be under low latency or high performance requirements, either due to cost saving measures on equipment, or due to an application that sees high utilization.There are few strong solutions today that don't leave gaping holes for intruders to achieve their low resource usage.

Additionally, there is a problem of interpreting these logs to derive useful meaning from them. A final consideration is the nature of Linux IT, which lends itself to believing it is secure, when in reality it is simply less targeted than Windows.

Many large enterprise organizations have well-established procedures to harden their perimeter and deploy Endpoint Detection and Response (EDR) tools on their Windows-based endpoints. We have often seen developer-friendly Linux environments not getting the same level of attention. Gaps in access control to the Internet, key management, and auditing often complicate the necessary alert monitoring for negligent or malicious network administrators. Below is some guidance for security teams to consider when dealing with effective Linux monitoring.

Auditing Credentials in Linux Environment: *Shared Root SSH Keys*

We often see shared SSH user keys for the simplification of managing the Linux or cloud environment. Accurate tracking of user activities is not possible when shared user keys are in place. All users having the ability to login as root allows them to make any changes they see fit, including removing evidence of their activities.

In one particular investigation, a company's environment was hosted in a cloud environment containing several unique SSH keys across 50 different, running servers. One server was using Active Directory authentication for SSH logins via the corporate Windows domain. Numerous servers had password authentication enabled for SSH connections which would allow any user with SSH privileges and the account password to login to the system regardless of having a valid key.

Most important to note, was the sharing of a root SSH key. Root keys should not be used at all and in most cases, it is impossible to attribute who used the key, especially if there is a security incident. SSH keys can be used effectively when each user is responsible for their own key. Root access can be enabled with the proper implementation of sudo and should be used when elevated privileges are required.

In summary, the sharing of SSH keys, especially a root key, should be prohibited. Implementation of sudo for elevated access should also be implemented. For Linux servers, we recommend switching from direct public/private key management to either an authentication certificate model, or Active Directory authentication. Though the former may be more effort initially, it permits for expiring access to servers, and also, revocation of access.

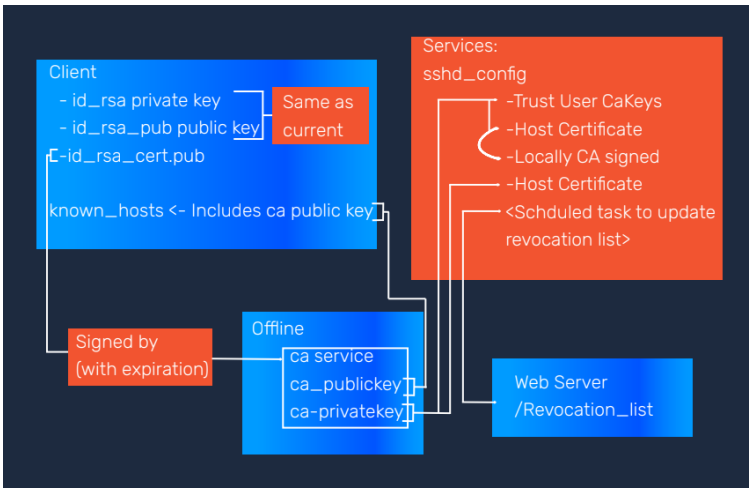
Remote Access from the Internet

We have seen Microsoft Remote Desktop Protocol (RDP) TCP port 3389 (default RDP port) typically accessed directly from the Internet in Linux environments that get compromised. RDP has been the target of many attacks and has been shown to have vulnerabilities in the past that make it a source of concern for security purposes. Since RDP also can be used to enumerate and fingerprint servers, it makes the potential for a breach much higher than exposing a single or few servers to effectively broker or proxy the remote access connection.

This practice allows for direct attack on the RDP service and increases the likelihood of an intrusion. Remote Desktop services can be installed easily on a single server or cluster and provide access to a group of windows servers. Similar to the recommendation above, we recommend transitioning from direct public/private key management to either an authentication certificate model, or Active Directory authentication. Though the former may be more effort initially, it permits for expiring access to servers, and also revocation of access. The recommendation for this configuration would be to use a completely offline server to create an SSH certificate-authority public/private key pair, much like with a company-internal SSL key pair, and then configure and sign certificates for all the servers.

Once this is done, each user will be requested to provide SSH public keys, which are then signed by the CA offline server. These in turn now can be used to access the servers with whatever accounts the administrator provided. Also, all the servers will be required to regularly update their revocation lists against a shared resource, like a web server, which will allow an administrator to instantly disable new logins to a system for a given user certificate.

We recommend limiting internet-facing RDP access via a web-based interface that effectively becomes the RDP connector. We recommend using the built-in capabilities provided by Microsoft, which may or may not require additional licensing. In pre-2008 Windows Server editions, this was referred to as Terminal Services, and is now called “Remote Desktop Services”.



SSH Key Revocation Process Diagram

Linux Auditing Generally Not Enabled

In many compromised or vulnerable environments, Linux servers were found to have auditing and logging disabled. Limited or no auditing prevents administrators and investigators from providing direct attribution of events on the systems, whether intentional or not. It is critical to have as much logging and auditing enabled on Linux servers as possible, to help diagnose problems, detect unauthorized changes, perform incident response, and assist in forensics when needed. Enterprises should be auditing and logging pre-existing Linux servers for possible security-related log events such as user logins, specific configuration changes, and user privilege changes. There are many solutions to monitor and audit the configuration of a Linux server. In addition to numerous commercial solutions with long pedigrees (such as Tripwire), within the Linux kernel itself lies the Audit Framework, a system auditing service capable of monitoring for file integrity, system calls, and much more. The Linux Audit Framework can be easily configured to send logs to a SIEM or centralized logging source, such as Splunk or an Elastic Stack.

Our recommendation is to configure the Linux Audit Framework to send file integrity data related to changes in its own configuration, the configuration of any services in question, the user account, and all logins back to an ELK (Elastic Search / Logstash /Kibana) stack configuration. This can be coupled with an alerting service such as PagerDuty or OpsGenie and ticketing systems like Jira to log and alert and automatically generate tickets for IT or Security Operations teams as desired.

Here is a breakdown of how such a system would work:

1. A setup file tells the audit framework what to audit for on a per-server basis including:
    - Security-related file changes/deletions
    - User logins
    - Time changes
    - User permission changes
2. A Linux daemon (similar to a Windows Service) queries the audit framework for all audited events.
3. Events are automatically sent to an Elastic Stack (ELK) instance.
4. Logs can be viewed via Kibana.
5. An alerting service such as elastalert.py constantly polls Elastic Search for anomalies.
6. Elastalert.py sends paging notifications when they are found.

No Key Management in Place

Oftentimes effective key management used to access the remote access (SSH) on the Linux servers within the audited range is lacking. Key management is a critical aspect of securing SSH when employees no longer need such remote access. Without key management, it becomes difficult to find and remove stale keys from the network. Stale keys allow unauthorized or de-authorized personnel to retain access, who can then either attack systems or expose the authentication data publicly, opening up attacks from anyone with SSH port access.

While this is often company specific and depends on the environment, moving public SSH key management to the Windows AD schema is usually a sound solution. The key criteria for this being a good solution is the Linux servers that can be administered more easily by the Windows system administrators who are not as comfortable with Linux.

With this solution, the authorized keys are dynamically provided to the Linux servers on demand, and do not statically reside on the Linux servers. This allows disabling of a Windows AD account to disable remote Linux SSH access too. Auditing will also be a critical piece of verifying that this configuration isn't being modified. Security teams will need to take steps outlined in the other recommendations within this paper, to ensure that attempts to circumvent the key management architecture are not attempted. This means watching the Linux servers at the file system for unauthorized keys, changes to SSHD service configurations and other anomalies.

For additional information, visit www.nisos.com or contact info@nisos.com.