Network Device Integrity (NDI) Methodology

February 23, 2016

Table of Contents

Network Device Integrity (NDI) Methodology

Introduction

Computer networks are complicated systems of interconnected devices of different technologies, to include workstations, servers, network devices and peripherals. Network devices are the systems that transport data between other systems on the network, which generally includes routers, switches, firewalls and various other types of technologies. Even though network devices generally do not store or process information, the compromise of these systems could allow an adversary or a malicious insider to gain unauthorized access to the information that is stored, processed or transmitted by other systems on the network, violating confidentiality. That same information could also be modified, corrupted or destroyed, violating integrity or impacting availability. Depending on the operational requirements for the network, any such violations could have a severe impact to the mission that the network supports.

How do I know if my network device has been compromised?

It is critical for network administrators to verify the integrity of all systems attached to a network to obtain a reasonable level of confidence that a compromise has not occurred, affirming that confidentiality, integrity and availability of the information on the network has not been violated. The Network Device Integrity (NDI) methodology is a process to verify the integrity of systems on an operational network, specifically network devices, such as routers, switches and firewalls. Any discrepancy discovered during the verification process could be an indication of a compromise.

Unlike other types of common systems, the operating system of a network device or an embedded system may not provide the flexibility or interface to validate critical aspects of the system, such as the hardware, operating system files or the contents of memory. This methodology is designed to be system independent and can be adjusted as needed, depending on the type of system that is being verified. Some systems may not support all of the necessary functionality and thus a higher level of confidence can only be obtained by using more intrusive methods. To adequately detect a compromise, it is first necessary to understand the different methods an adversary can use to successfully compromise a network device.

Unauthorized Access

The most common compromise of a network device is when an unauthorized individual gains user or privileged level access through an administrative interface. This could be achieved via direct physical access to the network device or via a remote network administration service such as Secure Shell (SSH), Telnet, Hyper-Text Transfer Protocol (HTTP) or Simple Network Management Protocol (SNMP).

Unauthorized access can be achieved by stealing legitimate administrator credentials (username, password, two-factor authentication token), obtaining credentials that are improperly protected by administrators (e-mailing configurations, storing clear-text passwords in a document), or compromising a centralized authentication server (TACACS+, RADIUS, LDAP) and adding or modifying accesses. Unauthorized access can also be achieved when a network device is poorly configured and authentication mechanisms are insufficient or can be easily bypassed, such as weak credentials that can be guessed or cracked via brute-force methods, or when publicly known default credentials are left unchanged. It is also

common for network administrators to overlook upgrading the software on network devices, allowing unauthorized access to also be achieved through the exploitation of software vulnerabilities in unpatched or outdated software.

It should always be assumed that unauthorized access can be obtained with physical access since most network devices have a mechanism to perform a password reset or similar function. A reboot of a network device is generally required for the password to be cleared or to reset the configuration to a default state. A terminal server can be used to remotely access the console port of network devices and should always be treated the same as physical access.

Once unauthorized access to a network device is obtained, an adversary can view or modify configuration settings, elevate privileges to gain additional accesses, perform software modifications, or utilize the system to bypass existing access restrictions to compromise other systems on the network where critical information resides.

Detecting unauthorized access to a system can be difficult, especially when an adversary utilizes stolen administrator credentials. An adversary may login to a network device from a compromised administrator workstation using the credentials of the same administrator that generally uses that workstation to manage the network devices. In this case, it is extremely difficult to distinguish between legitimate and unauthorized accesses, especially on the device itself. Unauthorized access is better detected by examining and correlating syslog messages, Authentication, Authorization and Accounting (AAA) logs, SNMP trap logs, administrator workstation logs, and any other available log messages. Any abnormal activities could be an indication of unauthorized access.

Several types of compromises, including those described below, can only be achieved by powering off a network device or rebooting it. When the proper logging and networking monitoring functions are enabled on the network, it should be reasonably simple to detect when a network device has been rebooted outside of normal network operations. All log messages should be stored on remote log servers to prevent an adversary from clearing the logs, which would require the adversary to also compromise the log servers to hide any malicious actions performed.

Software Modification

A more complicated compromise of a network device is the modification or replacement of the software running on the network device. This could be a modification to the operating system files stored on the device, or a modification to the operating system code residing and executing in memory. Another similar type of compromise is the modification or replacement of the software that boots the device and initially loads the operating system, such as the device's firmware, boot loader, or basic input/output system (BIOS), which is generally stored on a separate read-only memory (ROM) chip located on the device's motherboard. Some devices may allow the boot software on the ROM to be accessed, modified or upgraded via the operating system after the device boots, while others may require a reboot into a separate mode where the ROM can be accessed.

Unauthorized access is generally a requirement for an adversary to perform a software modification of a network device. Once access is obtained, the adversary can overwrite the operating system files stored on the device or residing in memory. Many network devices may require a reboot for a software

modification to take effect when stored files are changed, while a modification of code in memory should be immediately applied without a reboot.

Modifying the software of a network device is generally not a trivial task. It requires understanding the hardware and architecture of the particular network device, which generally differs from common desktop computers. This may require reverse engineering the existing software to determine how the software can be modified in a way that is useful for the adversary. Aside from the difficulty, there are several publicly known cases[1] [2] [3] [4] where an adversary has successfully modified the operating system files stored on network devices for malicious purposes.

Fortunately, the detection of a software modification on a network device can be achieved by comparing the software currently stored on the system or running in memory with a known good, assuming all of the necessary information is accessible or can be acquired. Many enterprise network device vendors provide known goods via their publicly available websites either via a software download or by publishing cryptographic hashes of the operating system files.

To perform the comparison, the software on the network device needs to be cryptographic hashed or downloaded and compared with the known good. A software modification can be detected as long as the functionality to retrieve this information from the network device is available and a trusted known good exists. When a known good does not exist or is not available from the vendor, a large sample of the same information from different systems may be sufficient to provide a reasonable level of confidence, or it can be used to establish an internal repository of known goods. Regardless, a known good provided by the vendor should always be the most trusted source.

Hardware Modification

Another more complicated compromise of a network device is the modification or replacement of the hardware. Additional hardware could be added to the device, such as a wireless radio, to grant an adversary out-of-band access for command and control, or an inline tap to exfiltrate sensitive information from an internal area of the network through a backdoor connection. Individual counterfeit components could also be produced, or an entire device could be manufactured to masquerade as a legitimate device and sold strictly for monetary gain. Recycled parts could be introduced in counterfeit equipment that could be less reliable. Even though the operational function of counterfeit devices may appear to be the same as legitimate devices, they are generally less reliable and vendors may decline to provide assistance if a counterfeit system is discovered during a support call.

A hardware modification, replacement or addition will always require the adversary to have physical access to the network device. It is generally easier to perform hardware modifications through the supply chain before the device is delivered to the intended location. Otherwise additional hardware could be added to the device after it has been installed and may or may not require the device to be powered off, depending on the modification.

Hardware modifications can range from an individual integrated circuit (IC) on a peripheral to an entire device. Because there are numerous avenues for an adversary to modify a piece of hardware, it can be extremely difficult to detect a hardware modification. This process is highly system dependent because every type of system can be uniquely different.

Prevention

It is prudent to describe several basic mitigation concepts to prevent the different types of compromises described above before they even happen. Designing enterprise networks or securing all of the network devices[5] and other systems on enterprise networks can be a significantly complicated task. The items in the list provided below are a few high level concepts that should be followed to assist with preventing compromises of network devices.

- Ensure physical access to network devices is properly restricted and some form of accountability exists to track who has physically accessed the systems. An adversary with physical access to a network device, even for a limited amount of time, could result in any of the compromises described above. Utilize a network access control (NAC) solution to prevent unauthorized systems from connecting to the network.

- Follow industry best practices for securing, managing and restricting remote access to network devices. Implement the concept of least privilege to ensure privileged level access requires additional authentication and is only used when it is required to restrict an adversary from elevating privileges from a compromised user account. Ensure accountability exists for all individuals that login to systems. Utilizing two-factor authentication can significantly decrease the ability for an adversary to steal or obtain legitimate credentials from an administrator. Do not store or transmit unprotected sensitive configuration information, including passwords and keys. Utilize encrypted protocols for remote administration and storage of sensitive information.

- Follow defense-in-depth when designing, building and re-configuring a network. Different areas of the network should be properly segregated and all administrative functions should occur on a separate management network not directly accessible from the operational network. The compromise of user workstations should not easily lead to the compromise of administrator workstations.

- Implement sufficient logging and establish policies and procedures to review the logs to ensure an unauthorized action is promptly detected and investigated. It is significantly easier to prevent additional compromises into the network when initial attempts are detected and handled.

- Verify the integrity of all software when upgrading or installing the software on a network device, and periodically verify the software running on the device to ensure it has not been tampered since the last upgrade. The integrity of all software should be verified by performing a cryptographic hash check on the files prior to copying them and after they have been copied to the network device to ensure an adversary has not influenced the software upgrade process.

- Purchase network devices through vendor approved distribution channels. Counterfeit or recycled equipment is generally sold for cheaper than approved resellers to entice buyers to purchase from their inventory. The higher cost and reasonable level of confidence achieved by purchasing a legitimate network device from an approved vendor can easily outweigh the additional costs of detecting and mitigating issues with acquiring a counterfeit piece of equipment.

Remediation

There are numerous procedures documented for remediating a compromised workstation or a security incident[6]. In most cases, a workstation can be easily removed from the network, wiped clean, rebuilt and replaced without significant impact. Unfortunately, this is generally not true for network devices. Removing a core router or another critical network device that is not load-balanced or redundant could have a significant impact on the availability of the network. Even removing a single switch from the edge of the network could disable network connectivity for a significant number of users.

My network device is compromised! What do I do?

There are several resources available that can provide additional guidance for remediating a compromised network device or embedded system[7] [8]. After detecting the existence of a successful compromise of a network device using any of the methods previously described, several steps can be followed to remediate the situation, but what should actually be done will depend on the operational requirements associated with the network. In some situations, it may not be acceptable to disable a portion of the network even to remediate a compromised network device.

Containment

The first step is to contain the compromise and ensure the adversary is unable to compromise other systems on the network. It is useless to remediate a single network device if the adversary has already been able to compromise other systems on the network. Discovering what other systems an adversary may have accessed may not be immediately available so it is also critical to perverse any forensic evidence, such as log messages, to investigate what other actions may have been performed by the adversary. Additional detection efforts may be required to determine if the adversary has accessed other network devices, workstations or servers throughout the network. It may be necessary to initially change some passwords and keys to contain the adversary if legitimate credentials were utilized.

To ensure containment of the adversary, it may be necessary to disconnect any external connections to other networks, including the Internet, partner networks, remote access or wireless connectivity. Even though the path used by the adversary may be obvious, it may be possible for the adversary to regain connectivity to the compromised system through another external connection, such as a wireless or cellular bridge. Investigating additional connections is necessary when a potential hardware compromise is detected or a wireless out-of-band connection exists.

Forensic Evidence

After the adversary is reasonably contained, forensic evidence should be obtained from the compromised system and any other systems that may have been accessible to the adversary. This includes any log messages or system status information that would be lost once the system is shut down. The following list can be used as an example of some items that should be obtained from a compromised system, but this list may not be complete.

- Current system clock for accurate log correlation.
- Current or previously logged in users.

- Network interface status and usage statistics.
- Established network connections.
- Neighboring devices.
- Routing tables and neighbors.
- Process listing.
- Memory usage.
- Status of authentication and time servers.
- Port security statistics.
- Access list statistics.

The compromised system can be disabled or shutdown and removed from the network after all forensic evidence has been obtained. If possible, it may be reasonable to leave the system powered on but completely disconnected from the network so additional forensic analysis can be performed on the system. This assumes there is not a hardware modification that would introduce adverse effects into the environment.

Be extremely cautious with replacing a compromised system with another system with a similar configuration as the compromised system. This may be desired to ensure network availability is restored without introducing unexpected changes into the network, but the adversary may be able to compromise the replacement system in the same manner if the original attack vector is not discovered and mitigated.

Discovery and Mitigation

Before restoring the network to an operational state, it is absolutely necessary to discover and mitigate the attack vector used by the adversary to compromise the network. If the vulnerability is not mitigated, it is likely the adversary will successfully compromise the same system or other systems in the network utilizing the same attack method. This may be an involved process of reviewing log messages and the forensic evidence obtained above to discover the vulnerability exploited by the adversary. It is important to note that the vulnerability may not be a technical issue (such as phishing), though it may be mitigated through technical controls that have not yet been implemented (such as e-mail scanning and quarantine).

It may be necessary to request or hire the assistance of more technically capable expertise to determine the cause of the compromise. Once the vulnerabilities are discovered, if more than one, they should be appropriately mitigated to ensure the adversary is unable to regain access to the network. It may also be necessary to perform a thorough vulnerability assessment of all systems on the network along with a mitigation implementation phase to reduce the vulnerabilities to an acceptable level prior to restoring the network to an operational state.

Change Passwords and Keys

If a network device has been compromised, either through unauthorized access, a software modification or even a hardware modification, it is likely that the adversary was able to obtain sensitive configuration information to include passwords or keys. After the adversary has been contained and all known compromised systems have been removed from the network, all credentials should be changed.

This includes local account passwords, centralized account passwords, centralized authentication keys, routing authentication keys, time synchronization keys, encryption and tunneling credentials (such as Virtual Private Networks), and any other passwords or shared keys that may be utilized by the network device. It may also be necessary to change administrator and user credentials on workstations and servers if the adversary was able to gain access to any of those systems as well. It may also be necessary to change passwords and keys more than once during the entire remediation process to ensure the adversary does not have any operational credentials when the network is fully restored.

When possible, it is preferred to utilize an out-of-band mechanism for changing passwords so the new passwords are not instantly captured by any lingering malware installed on systems. It is critical to follow best practice guidance for changing passwords, so the adversary cannot guess or infer the new passwords based on knowledge of the old passwords.

Restoration

Once all of the associated vulnerabilities have been mitigated and all known compromised systems have been removed from the network, it may be time to restore the network to an operational state. This may require installing replacement hardware or reconfiguring systems. Prior to re-establishing any connections that were disabled, it is crucial to ensure the proper logging and monitoring capabilities are implemented on the network to ensure any new compromises are immediately detected, investigated and responded to in a timely manner.

Network Device Integrity (NDI) - Unauthorized Access Detection

Unauthorized access via stolen credentials may be a common method used by an adversary to compromise a network device, but may be difficult to detect. There is no clear mechanism to distinguish unauthorized access from authorized access when legitimate credentials are used. Normal administrator behaviors must be captured and correlated so that abnormal activities can be detected and investigated. Most of the information necessary to perform this investigation may not be stored on the network device itself, and must be obtained from centralized log servers.

Administrators usually follow the same patterns in how they perform their administrative duties. This may include when they login, where they login from and how long sessions remain open. It is highly common to see the same administrative activities or behaviors repeated across the network. Any inconsistent activities or behaviors that seem out of the ordinary may be potential indicators of unauthorized access on a network device and should be investigated.

Unfortunately, there is not a standard process for correlating all of this information, and it will depend on how the network is configured and how systems are accessed. It is critical to understand normal administrator behaviors before abnormal activities can be detected. Several areas that can be investigated are described below.

Login Access

A primary area to investigate concerning unauthorized access is a list of accounts that have logged into a network device. If remote syslog logging or SNMP trap logging is properly enabled, this information may be available on a remote log server. If a centralized authentication server is utilized by the network devices and accounting is properly enabled, this information may also be better obtained from the Authentication, Authorization and Accounting (AAA) servers. It is critical to log when the access occurred, the source address and how long the session lasted.

Most employees, including administrators, have a consistent daily work schedule. This knowledge can be used to establish a timeframe when an administrator is expected to login to the network devices and how long the sessions should normally last. Any deviation outside of the normal behavior can be an indication of unauthorized access where the administrator's account credentials have been compromised.

Many administrators are also assigned a unique workstation, or a set of workstations, for performing administrative tasks. The source address of any connections to the network devices should generally be the same for each for administrator. This may depend on how the network is configured and what access restrictions are implemented. The source address may not be useful if all administrators first login to a centralized jump box to access the network devices. In this case, it may be necessary to obtain the access logs from the jump box in addition to the access logs from the network devices or centralized servers. Any account that is utilized from a different source address could be an indication of unauthorized access.

Configuration Changes

The configuration of a network device generally does not change very often once the configuration is in a stable working state. A change may occur when there is a problem with the network or authorized modifications need to be implemented. It is critical to monitor for any activities which involve modifying the configuration or any actions which would be considered high risk activities. Even though many of these actions can be legitimately used by administrators, they should be monitored to ensure they are initiated with the proper authorization. Some examples of high risk activities are listed below.

- Modification of access restrictions.
- Modification of authentication or authorization mechanisms.
- Modification of logging procedures.
- Modification of the boot process.
- Creation of new VLANs, tunnels, virtual interfaces or connections.
- Creation of new accounts.
- Configuration changes through non-standard methods (such as SNMP or console access).
- Outbound connections from a network device.
- Copying files to or from a network device.
- Clearing of log messages.
- Filling up the log buffer.
- Unscheduled reboots.
- Interface changes.
- Enabling debugging modes.
- Enabling shell access.
- Executing scripts.
- Starting new processes.
- Console port access.
- Connection or removal of external storage devices (such as USB).

Whenever a change is made to the configuration of a network device, it is critical for each device to log what administrator account made the change, when the change was made, and possibly what the exact change was. This information can be crucial for detecting unauthorized changes to the configuration.

Sophisticated adversaries may attempt to cover their tracks by clearing any log messages that were generated in connection with malicious activities. Without logs, it is difficult for network administrators to determine what may have happened after an event. If the logs on a network device or on the centralized log server have been cleared, removed or deleted, this is an instant indication that unauthorized access may have occurred.

Many network devices have a buffer to store some of the logs locally, even when they are sent to a remote log server. The adversary may attempt to overwrite the log buffer or generate additional log messages until the buffer fills up and rolls over, or clear the buffer by simply rebooting the device. Any of these actions could be an indication of unauthorized access.

Interface Changes

A malicious insider or an adversary that obtains physical access may connect an unauthorized system to the network to attempt to bypass security restrictions implemented on the existing workstations. Network devices, and especially switches, have multiple interface ports where physical network connections can be established. Whenever a system is connected or disconnected from an interface port on a network device, a notification is generally produced indicating that the status of the interface has changed to either up or down. These notifications can be useful to detect the existence of unauthorized systems on the network.

When possible, it is also critical to determine the media access control (MAC) address of any connected systems to determine if a newly connected system is unauthorized, or if an authorized system has simply been moved to a different port. A complete inventory of current equipment can assist with discovering unauthorized systems connected to the network. A network access control (NAC) solution, like 802.1X or even something as basic as port-security, can assist with discovering and blocking unauthorized systems connected to the network.

Physical Access

The logs on a network device should also be examined to determine if there were any attempts made to connect via the console port, which is usually just a basic serial connection. Physical access to a network device is generally required to connect to the console port. A terminal server can also be used to remotely access the console port of network devices. Most administrators do not utilize the console port when remote administration services are properly configured, and may only be used when the device cannot be accessed over the network. Any attempts to access the network device via the console port could be an indication of unauthorized access.

Many network devices also have a mechanism to configure how the device boots, through the device's firmware, boot loader, or basic input/output system (BIOS). There may be various settings that can be changed to influence the boot process, such as bypassing security mechanisms, ignoring the stored configuration or allowing the password to be reset. These settings are usually configured the same for all devices on the same network, and thus a device that has a different boot setting may be an indication of unauthorized access, or an attempt to physically access the device via the console port.

Unscheduled Reboots

Many network devices must be rebooted to upgrade the operating system or even when applying some configuration changes. These actions are usually scheduled by administrators to reduce the loss of availability of network connectivity. An unscheduled reboot of a network device can easily be an indication of unauthorized access or another potential problem. An unscheduled reboot is best discovered by reviewing the logs and system information or through network monitoring tools.

Network devices can be legitimately rebooted, due to specific administrator actions or upgrading the operating system. Whenever investigating a reboot of a network device, it is critical to check any logs or status messages to determine the cause for the reboot. A device that was powered on from a cold restart could have been accessed physically where a password reset was performed. A device that crashed and

was automatically restarted could have been due to an attempt to exploit a vulnerability or software flaw, such as a buffer overflow. A device could have been rebooted by an adversary to clear the logs or to boot into a modified version of the operating system.

Many network devices must be rebooted when upgrading the operating system. If the network device loads all of the operating code into memory during the boot process, any changes made to the operating system files stored on the network device will not be applied until the device is rebooted. Thus an adversary may have to force a device to reboot for any unauthorized software modifications to be applied.

It is important to determine how much time has passed since the last time the device was rebooted, and the reason for the last reboot. The longer a device has been up, the longer it has been since the stored files have been loaded into memory. Large up times (tens of weeks, or even years) generally point to several poor administrator practices. The software on a device that has not been rebooted in a long time has also not been legitimately upgraded in at least that same amount of time.

Rebooting a device periodically will ensure that the executable code running memory matches the stored operating system files, clearing out any potential memory-only implants. It is a good practice for network administrators to always verify the stored operating system files prior to rebooting a network device to ensure that the files are legitimate. It also confirms that the file storage mechanism is working. It is possible for the flash or hard drive storing the operating system files to fail or become disconnected, preventing the device from properly booting. It is critical for an administrator to discover such an issue prior to a reboot.

Software Management

Many network devices may only store one version of the operating system at a time, but others may be capable of storing more than one version simultaneously. If multiple versions of the operating system exist, the device may have been recently upgraded or network administrators may have failed to remove older versions after they were no longer needed. It could also be an indication of an attempted software modification. It is critical to verify all operating system files stored on a device, regardless if they are in use or not, to ensure a modified version of the operating system is not loaded on the next reboot.

It is also critical to verify what version of the operating system will load after the next reboot. If it is different than the current running version, it is possible the device was legitimately upgraded but the administrators were waiting for a scheduled time to reboot the network device. If the current running version of the operating system is no longer stored on the network device, it may have been removed prior to an upgrade because there was not enough space to store two different versions of the operating system. Any changes to the operating system image files and differences between the running and boot versions of the operating system could be an indication of a potential software modification and the adversary was waiting for a more favorable time to reboot the network device to reduce the probability of detection.

Another indicator to review when examining a network device is the operating system file timestamps. Unfortunately, the timestamps on a network device may not be accurate, especially if the clock is not properly synchronized with a centralized time server, or if the server was not available when the clock was set and operating system files were copied to the network device. Even though file timestamps may

not be reliable, they could still be used when correlating information and looking for anomalies to discover any potential unauthorized changes to the files.

Blocked Attempts

In addition to reviewing successful configuration changes, any actions that were blocked or denied by a network device could be an indication of an adversary attempting to gain unauthorized access, or an adversary that has already gained unauthorized access and is attempting to perform additional malicious activities. It is critical to investigate any actions that were blocked or denied, as this can lead to the discovery of a successful compromise or indicate the source of a potential adversary. The account utilized and the source address will be critical for detecting other attempts to gain unauthorized access. Some examples of blocked activities are listed below.

- Blocked inbound connections.
- Blocked outbound connections.
- Failed authentication (login) attempts, and locked out accounts.
- Failed authorization (action or command) attempts due to insufficient access.
- Denied connections due to insufficient (weak) encryption parameters.

Downgraded Encryption

Administrators are highly encouraged to utilize encrypted protocols for administrative tasks to ensure the confidentiality of sensitive configuration information and administrative credentials when accessing systems on the network. An adversary that is capable of collecting network traffic should not be able to easily retrieve this information by simply examining the traffic. Encryption is also utilized for Virtual Private Network (VPN) connections to provide confidentiality and protect communications between trusted sites.

An adversary that desires to collect the contents of encrypted communications may attempt to influence the connection negotiation process to force the protocol to utilize weaker protocols or keys that are more easily cracked. Most encrypted protocols begin with some form of unencrypted negotiation that can potentially be influenced by an adversary capable of modifying the traffic without violating the integrity of the client or server verification.

For example, an SSH server will immediately provide a connected client with the versions of the protocol that the server is capable of accepting. The client will respond with the versions it is capable of using, and the server will provide the encryption parameters that it can support. The client will choose from the supported encryption parameters and continue establishing the encrypted connection. If both the client and server are capable of accepting both SSH version 1 and 2, an adversary could modify the initial handshake to force both the client and server to negotiate using the weaker version 1 instead of the more secure version 2, or modify the available encryption parameters so the client can only choose from weaker parameters that are easier for the adversary to crack. Both the client and server will mutually agree on the weaker encryption parameters unless they are configured to only utilize higher versions of the protocol with strong encryption parameters.

The parameters utilized by encrypted protocols should be logged and reviewed to ensure that only the stronger encryption parameters are accepted by both ends of the communication. Any client attempting to connect that cannot support the stronger encryption parameters should be denied and the client software should be upgraded. Any attempt to utilize weaker encryption parameters could be an indication that an adversary is attempting to downgrade the encryption to enable information gathering to violate confidentiality, or an attempt to gain unauthorized access by bypassing existing authentication mechanisms.

Network Traffic Analysis

If a network device is compromised, the adversary may disable all of the monitoring capabilities and prevent insight into any malicious activities performed. Similarly, access logs may not be available from the network devices if logging has been not properly configured. Another useful method for detecting unauthorized access is to monitor network traffic.

A significant amount of information can be obtained by examining network traffic, to include source and destination addresses, the date and time of the activity, the amount of data transferred, and the duration of the activity. The existence of any remote access connections to a network device, authorized or unauthorized, should be visible on the network even when the communications are encrypted. Several types of anomalous behaviors that can be discovered in network traffic are listed below.

- Connections to a network device from an unexpected source address.
- Outbound connections with a source address assigned to a network device. Most network devices do not establish outbound connections to other systems unless initiated by an administrator logged into the device.
- Malformed packets or non-administrative traffic to or from a network device.
- Encryption parameters during initial connection handshakes and negotiations.
- Inbound connections that unexpectedly terminate at the network device, and potentially coincide with other outbound connections. Legitimate traffic may exist with this behavior if the network device is an endpoint for a Virtual Private Network (VPN), Generic Encapsulated Routing (GRE) tunnel, or provides proxy services.

The above behaviors are generally vendor independent and can be detected by examining network traffic regardless of the type of the system under investigation.

Network Device Integrity (NDI) - Software Modification Detection

The integrity of the software on a network device can be verified by determining whether or not any critical files or operating system code has been modified and is different from the original file that was provided by the vendor. This can be achieved by performing a hash-based verification of files stored on the network device and verifying the executable code running in memory by comparing it to legitimate operating system files or a known good copy of what is expected to be in memory.

It is critical to verify the executable code running in memory and not just the stored files because many network devices load the stored operating system files into memory when the device is first booted. The stored files are no longer necessary for the network device to continue operating after it boots. Since many network device are left operating for long periods of time without a reboot, potentially years, an adversary could modify the executable code running memory and there would be no indication of tampering if only the stored files were verified.

File Verification[9]

To verify an individual file, a cryptographic hash is computed based on the contents of a file and compared with a known good value. Any discrepancy with the comparison is an indication that the file has been modified, while an exact match provides a certain level of confidence that the file has not been modified. The process for computing these hashes is more complicated for network devices than other systems since there may not be a trusted method for computing the hashes or accessing the files, and thus multiple hashes must be obtained to achieve a higher level of confidence.

Some network devices may store the entire operating system in a single image file, while other network devices may have hundreds or thousands of executable files similar to common desktop and server operating systems. To fully verify the integrity of the operating system of a network device, it will be necessary to repeat this process for every file stored on the device that contains executable code. Even though file verification generally cannot be performed on configuration or text based files since they can change depending on how the system is configured, this process can be potentially used to detect configuration changes over time.

Online Hash

The online hash is computed by remotely logging into the network device and using the built-in functionality of the network device's operating system to compute the hash of the file. This piece of information is generally easy to acquire because it only requires remote network connectivity to the device and the proper access to login to the device and execute a series of commands. Any discrepancies between the online hash and a known good are an instant indication that the file has been modified.

Unfortunately, the online hash cannot always be trusted. If malicious modifications have been made to the operating system files, the adversary could have modified the hash computation functions to provide bogus information, which could be the correct hash for the unmodified version of a file, ultimately hiding the fact that the file has been modified.

Depending on the system and the vendor, there may be multiple methods available to compute and obtain an online hash. Since an adversary may attempt to hide an unauthorized presence by modifying the hash computation functions, it is recommended to utilize all methods of computing an online hash when additional methods exist. When available, different types of hashes should also be obtained (MD5, SHA1, SHA256, SHA512) to provide a higher level of confidence, since some common hash functions are known to have potential collisions.

Offline Hash

Instead of only relying on the operating system of the network device to compute the hash, the entire file should be copied from the device to a trusted system where the hash is computed. Copying files is significantly more complicated to perform over the network because it requires sufficient bandwidth, the capability to copy files, the proper access to copy files, the network must be configured to permit connections from the network device to the trusted system (a return route with no firewalls or access lists blocking the traffic), and the trusted system must be listening with the appropriate services to accept incoming files. The use of Network Address Translation (NAT) will also cause significant problems for return traffic. Regardless of the potential difficulty to acquire the file, the offline hash provides significantly greater confidence than the online hash because it is difficult for an adversary to influence.

When properly obtained, the offline hash is considered more trustworthy than the online hash. Similar to the problems with the online hash, an adversary could potentially modify the copy functions used by the operating system to provide the original unmodified file. A significant amount of sophistication is required to modify the operating system in this way, and may require an entire copy of the original file to also be stored on the system in addition to the modified file. Files can also be corrupted, intercepted or modified during transmission back to the trusted system, and can produce false positive results, depending on the protocols used. The offline file can be obtained multiple times using different transfer methods to increase the level of confidence, though this can be an extremely time consuming process.

Instead of copying files over the network, an even more trusted method of acquiring the offline hash is to obtain the file via physical access to the device. Some network devices store files on removable media such as flash cards or optical discs. This may require shutting down the device, opening the case, or rebooting the device in a limited access mode, which may not be feasible for operational networks.

Published Hash (Known Good)

The published hash is the known good hash that the online and offline hashes are compared to. This information generally must be obtained by the vendor of the network device or operating system. When available, additional information about the file should also be obtained, such as the file name, file size, operating system version, release date, device requirements, and any other potentially relevant information. When available, multiple hashes should also be obtained (MD5, SHA1, SHA256, SHA512). Some vendors freely provide this information via their website, while others may require an account, and yet others may not have a method to provide this information to their customers.

If the published hash is not available from the vendor but a legitimate copy of the original file can be obtained via a process to download software from the vendor, the published hash can be manually computed on that copy of the file. A legitimate copy of the original file would also be necessary to

perform a byte-for-byte comparison of the file to increase the level of confidence if comparing hashes is not considered a sufficient method for verifying the integrity.

Regardless of how it is obtained, the published hash must be a trusted piece of information. To properly discover a file modification, it is critical to compare the online and offline hashes with the published hash. The online and offline hashes cannot simply be compared to each other. When a file has been modified from the original, it is expected that the online and offline hashes should match exactly, but they would not match the trusted published hash, which is why the published hash is so critical to this process.

It is also possible for a single file to have multiple published hashes, and all of them may be valid for verification purposes. If the file has been legitimately updated by the vendor, it will have a different hash, and possibly a different file size. This is why additional information about the file should also be obtained from the vendor.

Hash Comparison

All three types of hashes should be compared for equality after they have been properly obtained. If all of the hashes could not be obtained, or the hashes were not equal, the cause of the issue should be determined. The fact that a hash could not be obtained through expected means could be an indication of a malicious software modification. Providing a reasonable explanation for the lack of information can ensure the results are as complete as possible and will assist with the analysis of the data and reduce the number of false positives. Several common scenarios are described below.

If the **online hash** could not be obtained, the network device itself may not support the functionality to compute the hash of the file. The lack of an online hash is not a severe concern since the offline hash is more trustworthy than the online hash, assuming the offline hash is successfully obtained. A missing online hash can be easily detected by examining of the output of any attempts to obtain the online hash. It is possible this may be an indication of a malicious modification of the operating system, though not as likely if the offline hash properly matches the published hash. Several potential reasons for missing an online hash are summarized below.

- The necessary accesses or permissions were not obtained on the network device.
- The network device did not support an online hashing functionality.
- A software bug in the network device operating system reported that the file did not exist.
- A malicious software modification prevented a file from being hashed online.

If the **offline hash** could not be obtained, it is likely something prevented the file from being copied off of the device. A firewall or router may have blocked the connection, or the trusted system may not have been properly configured to accept the incoming files. Again, it is possible this may also be an indication of a malicious modification of the operating system, and would be the result of a sophisticated adversary. Several potential reasons for missing an offline hash are summarized below.

- The necessary accesses or permissions were not obtained on the network device.
- A firewall or router blocked the connection from the network device to the trusted system.
- The network device was configured to prevent outbound connections.
- The credentials provided to copy the file were invalid on the trusted system.

- The network device did not support encrypted protocols.
- The copy process was not configured to attempt to utilize more than one protocol.
- The trusted system was not properly configured to accept files via the supported protocols.
- A return route did not exist between the network device and trusted system, due to Network Address Translation (NAT), Port Address Translation (PAT), or other routing issues.
- A software bug in the network device operating system reported that the file did not exist.
- The file was inaccessible due to a corrupted or disconnected file system.
- The file was corrupted during the transfer.
- The file transfer was incomplete.
- The file was stored in an unexpected or inaccessible location on the trusted system.
- The file system on the trusted system was full.
- A malicious software modification prevented the file from being copied offline.

If both the **online hash** and the **offline hash** were not obtained, it is likely that the file did not exist or could not be accessed. Many times this is caused by not having the correct privilege level for accessing the file or running the necessary commands on the device. In other cases, the file may have simply been deleted. Some network devices will load the entire operating system files into memory, and the stored file can be deleted once the operating system loads. This may happen when the network administrators have replaced or upgraded the operating system files on the device, but have not yet been rebooted the device to load the new files. The file system may also have been corrupted or disconnected, preventing the file from being accessed. Several potential reasons for missing both the online hash and offline hash are summarized below.

- The necessary accesses or permissions were not obtained on the network device.
- The file did not exist, was deleted or could not be accessed.
- The file was inaccessible due to a corrupted or disconnected file system.
- A malicious software modification prevented the file from being copied offline.

If the **published hash** is unavailable, a determination cannot be made about the integrity of the file because there is no known good value to compare to. Some vendors do not provide known good hashes so the necessary information is not available. And of course it is possible that the vendor did not create the file at all and thus a published hash does not exist. It may be necessary to contact the vendor directly to attempt to obtain a legitimate copy of the original file to manually compute the published hash. Several potential reasons for missing the correct published hash are summarized below.

- A known good hash was not published by the vendor.
- A legitimate copy of the file could not be obtained from the vendor.
- The file was renamed and a published hash did not exist with the corresponding name.
- The file was renamed and the wrong published hash was utilized for the corresponding name.
- A malicious software modification prevented the file from having a published hash.

If both the published hash and a copy of the original file cannot be obtained from the vendor, the best course of action is to compare the hashes across similar systems that have the exact same file. The more files with the same hash, the more likely the file has not been modified, especially when looking across multiple networks. This can be problematic if all files were cloned from the same source, such as a local

repository that contained a modified version of the file. The integrity of the original source of a file should be taken into account when using this method. This process does not provide a high level of confidence, but may be the only method available.

It is also possible that a file was renamed by a network administrator prior to it being copied to the network device. It may be necessary to verify the file name with the operating system version to ensure that the name of the file is correct. Otherwise the wrong published hash may be used during the comparison. And as previously noted, a file could be associated with multiple valid published hashes, so it may be necessary to compare more than hash before a valid match is discovered.

File corruption can occur during the transfer of the file, though this is not very likely when using reliable or encrypted transfer protocols. Some systems may behave differently when a file system becomes corrupt, and invalid data is returned when copying the file instead of presenting an error. This could result in differing online and offline hashes. In these cases, the entire offline contents of the file should be compared with a known good copy of the file to determine where the differences occur.

Files may be transferred to or from some systems using a serial console connection and the XMODEM protocol. XMODEM always transfers data in blocks of 128 bytes. Any remaining bytes in the last block are padded with the hexadecimal character 0x1A to indicate the end of the file. Some systems may incorrectly store the entire last block of the file with the padding. Both the online and offline hashes will not match the published hash because of the padded bytes at the end. If this behavior is discovered, the offline hash should be re-computed by removing all of the padded bytes at the end. If the file size was obtained from the vendor, it could be used ensure the file is truncated to the correct size.

A file modification does not necessarily indicate that the file has been modified maliciously. The existence of padding or corruption will result in properly detecting a file modification, but does not indicate that the system has been compromised. It these cases, it is absolutely necessary to obtain a known good and trusted copy of the original file from the vendor. A byte-for-byte comparison should be performed to determine how the files differ. Additionally, the modified file can be reverse engineered to investigate how the differences could cause the functionality of the operating system to behave differently. It may also be necessary to perform the verification process described above multiple times to obtain a different data set to determine whether or not there was an error that occurred during the initial attempt. A file modification should be considered a malicious modification only after all other reasonable explanations have been ruled out, or the file attributes are consistent with a previously known bad file modification.

The following table provides a summary of some of the scenarios described above that could be encountered while performing software modification detection. The passing results include a relative level of confidence associated with the result.

Hash Comparison Logic	Result	Possible Reason(s)
online == offline == published	**PASS (High)**	File has not been modified
offline == published; missing online	**PASS (Medium)**	Online hash not supported by device Invalid accesses or permissions Potential malicious file modification
online == published; missing offline	**PASS (Low)**	Network restrictions (firewall, ACLs) Configuration (no outbound connections) Invalid accesses or permissions Trusted system incorrectly configured Potential malicious file modification
online == offline; missing published	**INCOMPLETE**	Vendor did not provide hash or file File was renamed File was not created by vendor Potential malicious file creation
(online == offline) != published	**FAIL**	File has been modified
(online == published) != offline online != published; missing offline offline != published; missing online	**FAIL**	File or file system corrupted File has been modified (malicious)
(offline == published) != online	**FAIL**	File has been modified (malicious)
missing online and offline	**INCOMPLETE**	File does not exist File cannot be accessed, wrong permissions File system corrupt or disconnected

Self-Verification

Some network devices may include a system dependent or proprietary feature for internally verifying the integrity of operating system files. This may be a pre-computed hash, checksum or cryptographic signature that is stored in the file itself, or in some other location, that is treated as a known good. The network device may internally verify a file before using it by re-calculating the hash or checksum, or by cryptographically verifying the signature. Since these verification mechanisms store the known good along with the file, it may be easy for an adversary to replace the hash or checksum to match what is internally computed. A cryptographic signature would be significantly harder for an adversary to successfully modify without having the original private key, but it may be possible to replace the public key used for verification if it is also stored on the network device.

If the vendor provides an internal feature to verify files, it should be used in addition to the file verification process described above to provide an overall higher level of confidence. Even though it may not provide a high level of confidence by itself, it can be used to quickly detect file modifications or prevent the execution of modified files. Some network devices may not have this verification feature enabled by default, so it may be necessary for administrators to enable it.

Higher Confidence Levels

In some cases, it may be necessary to provide even more confidence that a file has not been modified. A higher level of confidence can be obtained by attempting to verify a file using all methods available. This

includes obtaining all possible online hashes, obtaining an offline copy of the file using all protocols available, and utilizing all available self-verification features. This can be time consuming, but the more methods that provide a positive result, the more likely the file is legitimate and has not been modified.

If additional confidence is required, multiple cryptographic hash functions could be used rather than relying only on one, such as following the above process utilizing two or more hashes. Obviously this method requires the vendor to provide the published hashes from multiple cryptographic hash algorithms. If the vendor does not provide these additional hashes, it may be necessary to obtain a trusted copy of the original file and manually compute the published hashes.

Another method to further increase the level of confidence is to compare the entire offline file byte-for-byte with a trusted copy of the file. This provides significantly greater confidence than hashes since some hashes may be subject to collision attacks. It may be more difficult to obtain an original copy of file from the vendor, especially without a maintenance contract. This also requires an offline copy of the file from the network device, but fortunately this is already obtained for computing the offline hash.

Memory Verification

Similar to file verification, a cryptographic hash can be computed on the contents of memory and compared with a known good value. The contents of memory are always changing in every system, so it is useless to compute the hash on the entire contents of memory. The operating system code that resides in memory theoretically should remain static so the cryptographic hash should only be computed on those areas in memory where executable code resides. Determining the location of executable code in memory can be an extremely difficult task for some systems.

Unfortunately, performing a simple hash comparison may not be sufficient for verifying the memory of some systems. The operating system can implement address space layout randomization (ASLR)[10] to protect against buffer overflow attacks by randomly arranging the location in memory where code resides when the system first boots. It will be useless to compute the hash on these portions of memory if they are different every time the system is booted. For systems that implement ASLR, it will be necessary to rearrange the expected contents of memory to match the layout of the running system before a comparison can be made. Rather than using cryptographic hashes, it is likely more effective to compare the actual bytes in memory and note any differences.

Executable space protection[11] is another security mechanism where an operating system marks certain areas of memory as non-executable. Systems that do not implement such protections could allow code to be executed in other areas in memory which normally would not contain executable code and thus would not be verified through this process. Fortunately, it is highly likely that the areas of memory where executable code is expected to reside would need to be modified in order for the system to also execute any code residing in other areas of memory. For this reason, it is assumed that memory verification should only need to be performed on the areas of memory where executable code is expected to reside, as long as this process is complete and all areas of executable memory are verified.

Network device vendors may also implement other methods for attempting to prevent executable code from being modified in memory or from executing code in other areas of memory. These methods will

significantly complicate the processes used for verifying the contents of memory, and will be system dependent cases that will need to be handled separately.

Regardless, in order to perform memory verification on a network device, there must be an administrative function that allows the contents of memory to be hashed or copied offline. If such a function does not exist for a specific device, it is likely that memory verification cannot be performed on that device. In those cases, it is recommended to perform file verification, and if all hash comparisons are correct, reboot the device to ensure the legitimate files are executed during the boot process.

But even if an administrative function does exist for hashing and copying the contents of memory, it is highly unlikely that vendors will provide a known good cryptographic hash for executable code residing in memory. It will be necessary to establish a set of known good hashes of memory for specific versions of the operating system and each model. Similar to file verification, systems running the same version of the operating system can be compared with each other. This can be a tedious process but can be used to quickly determine whether or not executable code has been modified in memory. Unfortunately, this process will not be sufficient for systems utilizing ASLR or other vendor proprietary methods.

Since all executable code in memory should be derived from one or more files stored on the device, it theoretically should be possible to generate the known good contents of memory from known good copies of the operating system files. This would be a significantly more tedious process for even a single version of the operating system, and requires knowledge of how the operating system files are copied into memory for execution during the boot process.

Firmware Verification

Even if all of the operating system files stored on a network device were properly verified and the contents of memory were assumed to be correct since legitimate files were used when the device booted, it is still possible that a software modification exists in the firmware, boot loader, or basic input/output system (BIOS) that is executed when the device boots. This is software that is generally stored on a read-only memory (ROM) chip located on the device's motherboard and is used to load the operating system files when the network device is first turned on or rebooted. For some devices, the firmware may be the same thing as the operating system. The term *firmware* will be used to refer to the software used to boot the device and load the operating system files, and not the actual operating system files.

A sophisticated modification of the firmware could cause legitimate operating system files to be modified as they are loaded into memory, resulting in a memory-only operating system software modification that would not be detected by verifying the stored files, even if the device was rebooted with files known to be legitimate. Fortunately, it is assumed that a malicious modification of the firmware would also result in a modification to the executable code in memory. So for those systems that do support an administrative interface to access the contents of memory, a firmware modification could be detected by performing memory verification in addition to file modification, even though it may not be obvious that a firmware modification may have been the source of a detected memory modification.

Some systems may allow the ROM chip containing the firmware to be accessed while the operating system is running. Other systems may only allow the firmware to be accessed from a minimal interface that is only accessible with physical access and after rebooting the system into a non-operational mode.

Other systems may only provide write access to upgrade the firmware and require the ROM chip to be physically removed from the device and accessed via a specialized chip reader to read the contents. Regardless of the method used to acquire the firmware, a cryptographic hash can be computed on the firmware and compared to a known good, similar to file verification. It will be highly system dependent whether or not a specific network device supports verification of the firmware in this manner.

Unfortunately, the only option to verify the integrity of the firmware on some systems may be to simply overwrite it with a known good copy provided by the vendor.

Rootkit Detection

One final area to investigate when verifying the integrity of a network device is to attempt to obtain the same information using multiple methods. Many rootkits attempt to hide their existence from users and administrators by hooking various operating system calls and modifying the output before it is provided to the caller. If the same information can be retrieved through multiple methods, it is possible that the adversary may not have hooked all of them.

As an example, an adversary could insert a malicious section into the configuration and then hide it by hooking the call that retrieves the configuration. When the configuration is displayed to an administrator, the hook could remove the malicious section, hiding it from view. But if the administrator is able to retrieve the configuration using a different call that the adversary was not aware of or did not hook, the malicious section could easily be seen by comparing it with the output from the other call.

At a minimum, the configuration of a network device should be obtained through multiple methods, if supported by the network device. If also supported, other information should also be obtained in the same manner, such as established network connections, local accounts, routing tables and any other information that would be beneficial for an adversary to modify to prevent detection.

Network Device Integrity (NDI) - Hardware Modification Detection

It can be extremely difficult to verify the integrity of any piece of hardware, even when physical access is available. Some components can be detected as counterfeit by performing an X-ray and discovering irregularities by comparing the result with other similar equipment[12]. Some other studies have shown that a malicious hardware modification can be detected by physically testing the integrated circuits (IC) and identifying unknown functionality[13]. Wireless hardware modifications can be detected by analyzing radio frequency (RF) signals that are emanating unexpectedly from a device. Unfortunately, all of these methods require direct physical examination of a device and cannot be quickly repeated for a large number of devices in a short amount of time.

Physically examining every network device is probably not feasible for most networks due to physically separate locations or the vast size of the network. Regardless, there are several actions that can be performed on some systems when only remote access is available. Even though these steps may not provide a high level of confidence, they may still provide a basic indication of a potential hardware issue.

Unique Identifiers

Most network devices are hardcoded with a unique serial number. Even individual pieces of equipment that can be added or removed may also have their own unique serial numbers, such as pluggable add-on cards, interfaces or blades. Serial numbers often follow a very specific format according to each vendor, and the serial numbers assigned to a device can be verified to conform to this specification. Some vendors may also provide a service that allows customers to validate serial numbers as legitimate pieces of equipment and verify where they were shipped and who they were sold to.

It is assumed that an adversary producing counterfeit equipment will already know the expected serial number format. To conform to the expected format, the adversary may simply duplicate or clone known legitimate serial numbers. A duplicate serial number can be detected by gathering serial numbers from unique devices and looking for more than one instance of the same serial number. Since the serial number may be commonly used to identify unique devices based on the physical hardware, it will be necessary to utilize a different unique attribute to identify devices independent of the hardware. The configuration of each system should be unique, because each system should at least have a different IP address assigned to it, and thus a cryptographic hash of the configuration should be sufficient to uniquely identify devices. This assumes no two devices have identical configurations, the configurations of the network devices do not change while data is obtained from all the devices on the network, and the same cryptographic hash is used for systems of the same type. This unique identifier can be correlated with serial numbers to detect potentially duplicate serial numbers.

Another unique piece of information that should be assigned to a network interface card attached to a network device is the media access control (MAC) address. The MAC address should be unique for each physical interface. The MAC addresses on all network devices can be obtained and correlated similar to serial numbers. For some devices, it may only be necessary to obtain the chassis MAC address if subsequent addresses on the same piece of equipment are incremental.

Operating Statistics

Many network devices also have a mechanism to monitor specific attributes about the operating status of the hardware itself. This includes voltages, temperatures and fan speeds. Generally these values should be within a normal expected range, as specified by the vendor. These ranges will be vendor and system specific, and it depends on the vendor whether or not this information is available. Any deviation of these values may be a potential indication that the hardware is counterfeit. Independent of detecting hardware modifications, this information can also detect potential hardware failures before they occur.

Attributes like voltages and temperatures can be obtained by physically examining a device with external gauges, but some of this information may be available when only remote access is available. A network device must be built with internal sensors and be capable of monitoring itself and include a mechanism to reliably report this type of information to network administrators. When this information is available, these attributes can be compared with other legitimate systems, which may be inconsistent in counterfeit equipment. Unfortunately, these attributes will likely provide a lower level of confidence than physically examining a system.

Network Traffic Analysis

Another method to detect counterfeit equipment is to passively monitor network traffic that is processed by a device and calculating the time elapsed for the device to process a packet[14]. When a legitimate piece of hardware is installed, an individual packet of the same size should theoretically take the same amount of time for a network device to process. Any deviations from normal expected behavior could be an indication of a potential hardware modification.

Similar to unauthorized access detection, a hardware modification may also be detectable by passively monitoring the contents of network traffic or examining the source and destination addresses. The introduction of unexpected network connections that originate from the network device itself could be an indication of a hardware modification that exfiltrates sensitive information to an external destination. If the software on a network device is exhibiting this type of behavior and is verified to be legitimate, the source of the unexpected network connections could be an additional piece of hardware added to the network device.

Fortunately, analyzing network traffic is generally not system specific, so the detection of unexpected network connections can be performed for any type of system, as long as the passive sensors monitoring the traffic are installed in appropriate locations on the network around the suspect systems.

Network Device Integrity (NDI) - Worksheets

The following pages contain several template worksheets that can be used for performing Network Device Integrity (NDI). These worksheets can assist with ensuring all of the necessary information is obtained from each device and the most critical areas are reviewed during the analysis of that information. After these worksheets have been filled in with the relevant information, the methodology provided above can be performed with this information to verify the integrity of the network device, along with any vendor specific cases.

These worksheets can be printed double sided and the corresponding information key will be printed on the back side of each worksheet for reference.

Network Device Integrity (NDI) - System Information		
Date / Time		
Host Name		
Model		
Device Serial Number		
Add'l Serial Numbers		
Unique Attribute **Hash of Stored Configuration**		
IP Addresses	**MAC Addresses**	
Uptime		
Firmware Version(s) **Firmware / Boot loader / BIOS**		
Boot Settings		
Last Reboot Reason		
Self-Verification **Enabled?**		
		Exists?
Running OS Version		
Boot OS Version(s)		
Other OS Version(s) **Exists but not used**		
Operating Statistics **Voltages, Temperatures, Fan Speeds**		
Rootkit Detection **Output Comparison**		
Observations		

System Information Worksheet

This worksheet contains a table of information that should be obtained from each unique network device that is examined.

- **Date / Time** - when the provided information was obtained (include time zone)
- **Host Name** - name assigned to the network device (may not necessarily be unique)
- **Model** - vendor assigned hardware model
- **Device Serial Number** - serial number assigned to the device (or chassis)
- **Add'l Serial Numbers** - any other relevant serial numbers assigned to peripheral equipment
- **Unique Attribute** - a piece of information that can uniquely identify a network device independent of the hardware, such as a cryptographic hash of the configuration
- **IP Addresses** - relevant IPv4 and IPv6 addresses assigned to the network device
- **MAC Addresses** - relevant MAC addresses assigned to the network interfaces
- **Uptime** - how long the device has been up
- **Firmware Version(s)** - firmware, boot loader or BIOS used to boot the device; if more than one version exists, note which version is used
- **Boot Settings** - firmware, boot loader or BIOS settings that determine how the device boots
- **Last Reboot Reason** - why the device was last rebooted
- **Self-Verification** - whether or not self-verification is enabled for executable files
- **Running OS Version** - current running version of the operating system
- **Boot OS Version(s)** - versions of the operating system that may attempt to be loaded after the next reboot of the device
- **Exists?** - it should be noted if the running OS version or any of the configured boot OS versions do not exist on the file system
- **Other OS Version(s)** - any other executable files stored on the network device that are currently not in use and will still not be used after a reboot
- **Operating Statistics** - voltages, temperatures, fan speeds and similar relevant information
- **Rootkit Detection** - any relevant information concerning the comparison of similar outputs obtained via different methods
- **Observations** - any other relevant observations about the network device

Network Device Integrity (NDI) - Unauthorized Access Detection			
Device Identifier Host Name / Serial Num / IP Addr			
Login Access			
Date / Time	**Account**	**Source Address**	**Duration**
Configuration Changes			
Date / Time	**Account**	**Source Address**	**Change Detail**
Blocked Attempts			
Date / Time	**Account**	**Source Address**	**Attempted Action**
Interface Changes			
Date / Time	**Interface**		**State**
Other Events			
Date / Time	**Event**		

Unauthorized Access Detection Worksheet

This worksheet contains a table of information that should be obtained for each event that occurred concerning a network device that should be investigated. Obviously, there are probably more events than can fit on this worksheet, and some events may contain relevant information not requested by this worksheet. Regardless, this worksheet can still be used as a reference to determine what types of events should be reviewed first.

- **Device Identifier** - unique host name, serial number or IP address for the network device
- **Date / Time** - timestamp when the event occurred
- **Account** - user account associated with the corresponding action
- **Source Address** - if known, the source address of the event
- **Duration** - the length of time the administrative connection was active
- **Change Detail** - what information was changed in the configuration
- **Attempted Action** - what action was attempted that was blocked or denied
- **Interface** - the interface name or identifier that was related to the event
- **State** - the new status of the interface according to the event (up, down, etc)
- **Event** - detailed information about the event that occurred

Network Device Integrity (NDI) - File / Memory / Firmware Verification			
Device Identifier **Host Name / Serial Num / IP Addr**			
File Name			
File Path			
File Version			
File Size			
File Timestamp			
File Permissions			
Hash Type			
Online Hash(es) **and Source**			
Offline Hash(es) **and Source**			
Published Hash(es) **Known Good**			
Published File Name			
Published File Path			
Published File Version			
Published File Size			
Published Release Date			
Hash Comparison **Pass / Fail / Incomplete**		**File Size Comparison** **Pass / Fail / Incomplete**	
Self-Verification **Checksum or Other Hash(es)** **and Verification Result**			
Observations			

File / Memory / Firmware Verification Worksheet

This worksheet contains a table of information that should be obtained from each file on a network device that should be verified. This should include operating system files and other executables, and any other files that do not normally change and could potentially be used by an adversary to perform malicious activities. Additional copies of the worksheet can be used for different files, or for the same file but using a different type of cryptographic hash. This worksheet can also apply to the contents of memory and the firmware in addition to stored files, even though all of the fields may not apply in each situation.

- **Device Identifier** - unique host name, serial number or IP address for the network device
- **File Name** - name of the file
- **File Path** - path of the file in reference to where it was stored on the network device
- **File Version** - vendor defined version of the file, if applicable
- **File Size** - the number of bytes contained in the file
- **File Timestamp** - the date and time the file was last written on the network device
- **File Permissions** - the permissions assigned to the file
- **Hash Type** - type of cryptographic hash utilized in the fields below
- **Online Hash(es)** - any online hashes obtained for the above file, including the source such as the command utilized to obtain the hash
- **Offline Hash(es)** - any offline hashes associated with the above file, including the source such as the protocol utilized to obtain the offline file
- **Published Hash(es)** - any published hashes associated with the above file
- **Published File Name** - name of the file associated with the above published file
- **Published File Path** - path of the file associated with the above published file
- **Published File Version** - operating system version associated with the above published file
- **Published File Size** - number of bytes contained in the above published file
- **Published Release Date** - date the above published file was released by the vendor
- **Hash Comparison** - result of comparing the online, offline and published hashes
- **File Size Comparison** - result of comparing the file size with the published file size
- **Self-Verification** - results of any vendor specific self-verification features
- **Observations** - any other relevant observations about the file

References

[1] Cisco Event Response: SYNful Knock Malware [October 9, 2015]
http://www.cisco.com/c/en/us/about/security-center/event-response/synful-knock html
[Accessed February 23, 2016]

[2] Evolution of attacks on Cisco IOS devices [October 8, 2015]
http://blogs.cisco.com/security/evolution-of-attacks-on-cisco-ios-devices
[Accessed February 23, 2016]

[3] Nasty Cisco Attack [August 19, 2015]
https://www.schneier.com/blogs/archives/2015/08/nasty_cisco_att.html
[Accessed February 23, 2016]

[4] Cisco network kit warning: Watch out for malware in the firmware [August 13, 2015]
http://www.theregister.co.uk/2015/08/13/cisco_warning_malware_in_firmware/
[Accessed February 23, 2016]

[5] Cisco Guide to Harden Cisco IOS Devices [January 6, 2016]
http://www.cisco.com/c/en/us/support/docs/ip/access-lists/13608-21 html
[Accessed February 23, 2016]

[6] Computer Security Incident Handling Guide [August 2012]
http://nvlpubs.nist.gov/nistpubs/SpecialPublicatiosn/NIST.SP.800-61r2.pdf
[Accessed February 23, 2016]

[7] Targeted Cyber Intrusion Detection and Mitigation Strategies (Update B) | ISC-CERT [February 6, 2013]
https://ics-cert.us-cert.gov/tips/ICS-TIP-12-146-01B
[Accessed February 23, 2016]

[8] A Hack Attack [No Date]
http://www.cisco.com/cisco/web/solutions/small_business/resource_center/articles/secure_my_business/hack_attack/index.html
[Accessed February 23, 2016]

[9] File verification - Wikipedia, the free encyclopedia [February 15, 2016]
https://en.wikipedia.org/wiki/File_verification
[Accessed February 23, 2016]

[10] Address space layout randomization - Wikipedia, the free encyclopedia [February 20, 2016]
https://en.wikipedia.org/wiki/Address_space_layout_randomization
[Accessed February 23, 2016]

[11] Executable space protection - Wikipedia, the free encyclopedia [February 3, 2016]
https://en.wikipedia.org/wiki/Executable_space_protection
[Accessed February 23, 2016]

[12] The Hidden Dangers of Chop-Shop Electronics - IEEE Spectrum [September 20, 2013]
http://spectrum.ieee.org/semiconductors/processors/the-hidden-dangers-of-chopshop-electronics
[Accessed February 23, 2016]

[13] Addressing the Challenges of Hardware Assurance in Reconfigurable Systems [2013]
http://ersaconf.org/ersa13/papers/Robinson-Hardware-Assurance.pdf
[Accessed February 23, 2016]

[14] A Network-based Approach to Counterfeit Detection [November 2013]
http://www2.ece.gatech.edu/cap/papers/HST13%20Paper%20186%20A%20Network-based%20Approach%20to%20Counterfeit%20Detection.pdf
[Accessed February 23, 2016]