Technical Disclosure Commons

Defensive Publications Series

October 08, 2018

DYNAMIC DEVICE COMPROMISE FOOTPRINT PRE-FILTERING

Eric Voit

Jared Pendleton

Chris Mccoy

DYNAMIC DEVICE COMPROMISE FOOTPRINT PRE-FILTERING

AUTHORS:
Eric Voit
Jared Pendleton
Chris Mccoy

ABSTRACT

Techniques are provided herein to enhance detection of compromised network devices by maintaining a list of network device indicators of compromise (i.e., a footprint), determining the viable footprints relevant to the device's specific deployment context, and placing checks for these footprints onto a device. Based on the reduced set of footprints, these footprints are placed in a cryptoprocessor (e.g., Trusted Platform Module (TPM)) to ensure that potentially relevant evidence cannot be silently discarded. As soon as a new footprint is characterized, devices may forward found instances of these footprints to a security controller. This allows the security controller to do remediation well prior to the installation of fixes/patches. Placement of events in a TPM also allows attacks on bare metal machines to be detected by virtual machines.

DETAILED DESCRIPTION

Certain types of Linux logs or operating system application level SYSLOG messages should not be seen with specific router configurations or deployment topologies. However, the router code which emits those messages is not self-aware of the platform configuration, or even the configuration of its networking peers. The result is that these messages themselves are not known by the emitting code as being more or less serious based on a specific deployment environment.

For example, the SYSLOG message "%PARSER-5-CFGLOG_LOGGEDCMD: User:unknown" should not be seen in any deployment which mandates the authentication of users. However, this message can appear because the code exists on the router, and if it does, it might be a footprint of an attack. As a concrete example of this, the SYSLOG message above will appear when a Vault 7 CIA Exploit targets routers with malformed Telnet packets.

Published by Technical Disclosure Commons, 2018

A Software Defined Networking (SDN) network controller has the ability to understand a router's deployment context, and this understanding may be tuned as features are added or removed from the router. As a result, it may be possible to dynamically maintain a list of SYSLOG (or Linux log) substrings which may correlate to known attacks for a deployment configuration. This substring list may be loaded into a router via technologies such as an Embedded Event Manager (EEM). The EEM may immediately report these possible footprints to both a security controller and a local router cryptoprocessor (e.g., Trusted Platform Module (TPM)).

Using the cryptoprocessor has several benefits. First, an attacker will not be able to remove footprints, even in a router which is isolated from its controller. Second, information on physical device attacks can be placed into cryptoprocessors, allowing compromise details to flow from physical devices to Virtual Machines (VMs) running thereon. This way an attack on a computing system could be shown to a virtual router running thereon. Third, when a controller and router are disconnected and re-connected, an immediate reporting of the pre-filtered hardware signed log messages can be accomplished. Using the cryptoprocessor helps validate that information was not deleted in the interim.

Effectively, these aforementioned benefits mean that Linux log or SYSLOG footprints specific to a software release, platform configuration, and/or neighbor platform configuration can be identified/filtered/prioritized, recorded in a local cryptoprocessor, and passed northbound for evaluation. The result will be fewer false positives being considered. From that point, the security controller will further evaluate these footprints.

Vulnerability detection is enhanced by maintaining an embedded list of detectable footprints so that as soon as a new router vulnerability footprint is characterized, a security controller can perform remediation on attacks (e.g., well prior to the installation of fixes/patches). Security relevant Linux log or SYSLOG message footprints are evaluated within a deployment context so that false positives can be reduced. For example, both the deployment context and known bugs list may be evaluated to determine whether the footprint can safely ignored or de-prioritized. Based on the reduced level of potential footprints, these footprints are placed in a cryptoprocessor (e.g., TPM) to ensure that potentially relevant evidence cannot be silently discarded.

The cryptoprocessor may identity itself by the security controller for correlation of logged footprints (as indicators of attack) across physical and virtual machines (e.g., a virtual router running on a computing system should not be trusted unless the TPM on the computing system also validates). Physical machine security is correlated with virtual machine security by allowing the VM to access trusted log information kept in the cryptoprocessor. This may be accomplished using a key being returned signed by the physical device's box TPM which must be used on the virtual box if it is desired to access the virtual TPM. Alternatively, the cryptoprocessor itself may be the means by which a physical device signals to VMs running thereon that it may have been compromised.

A security controller may, in near-real-time, take action to clear/reset a compromised router where a known footprint is discovered/exposed. A security controller may route traffic around devices which are currently deemed as potentially compromised based on the latest logged footprint activity. For example, it might be some amount of time before a deeper look might be made on a device showing suspect SYSLOG messages. In this case, highly secure traffic should be routed around that device in the meantime. Discovered attack footprints may be immediately replicated across the universe of devices before a patch is available. EEM filters are updated continuously, and may be fully automated based on the posting of footprints within the latest security advisory/alert. EEM rule lists may be automatically created and continuously refined/tuned for a specific box/release/topology, as a security controller examines and dismisses real-time EEM events being observed.

There are several vulnerability footprints where hardware backed detectability is possible as soon as the footprint is identified. One example is the ROCEM Telnet exploit. The originator was the Vault 7 CIA leak. The objective was to modify switch memory so that any subsequent Telnet connections needed no password. The exploited vulnerability was the incorrect processing of malformed Connectivity Management Processor (CMP) - specific Telnet options. Detection/visibility may be provided by commands run in the Telnet session still log. The SYSLOG message "%PARSER-5-CFGLOG_LOGGEDCMD: User:unknown" can act as a footprint. The expectation is that a Command Line Interface (CLI) command with "user:unknown" should not be seen in deployments where a form of Authentication, Authorization, and Accounting (AAA) is required for access. Beyond

ROCEM, it is possible that this footprint for command logging could be a more general vulnerability footprint where AAA is mandatory.

A second example is Dynamic Host Configuration Protocol (DHCP) relay buffer overflow. Multiple instances of buffer overflow attacks exist. Each modifies memory in a way which allows an unauthenticated, remote attacker to gain control of the execution pointer to execute arbitrary code and gain full control of an affected system. The exploited vulnerability was the incorrect processing of malformed DHCP packets. Detection/visibility may be provided via the SYSLOG message "%DATACORRUPTION-1-DATAINCONSISTENCY: copy error, -PC= 0x851ACD58z -Traceback= 80C54290z 80B7D0A0z 823C1EA8z 80CA8EE4z 80CA9C00z 80CAA618z 80CAA81Cz 801E01B8z 801C55ACz." This SYSLOG message is sent when the operating system detects an inconsistency in its internal data structures. A subset of buffer overflow attacks which are yet-to-be-discovered may leave this footprint. Examples of such buffer overflow bugs that leave this footprint are known to exist. But exploits for these bugs have not been detected in the field. This message is still seen in non-buffer-overflow cases and may often be safely disregarded. If this was generalized, false positives where the deployment context is known would need to be disposed of. This can be done by examining the specific details of the message, as well as determining whether relevant features are active on a device.

A third example is counterfeit exposure after upgrade. The originators were grey market resellers. The objective was to trick a secure boot by making the Basic Input/Output System (BIOS) check acceptable. This is accomplished using counterfeiter solders in a daughterboard to make the BIOS appear acceptable. Detection/visibility is provided by the "%ILET-1- AUTHENTICATION FAIL" message that comes up when software is upgraded, and the daughterboard no longer has acceptable values. However this same message can also result from regular bugs (e.g., when a switch is inserted into a stack with a stack cable). Checking to see whether the SYSLOG message is of consequence can be performed by eliminating the known bugs from reporting.

A fourth example is pernicious test commands. Runtime infections often leverage test commands. The objective is to determine specific commands needed that may possibly write data in undesirable locations or crash the system. The exploited vulnerability is test commands with overbroad capabilities. Detection/visibility is provided by SYSLOG

messages which identify that a particular command was run. Certain messages might be unexpected, and these high-risk messages may be sent to a network center and/or the TPM.

For instance, “test mcu” has commands available to read/write to field-programmable gate array registers and inter-integrated circuit components. “test nvram” can lock nvram. “test platform software crimson” can lock, read, and write to a database, and some subcommands cause the system to crash. “test platform software fed” can allow for possible register manipulation. “test plat sof fed switch active xcvr lpn 1 1 dump 0 255” freezes the machine, and requires a power cycle. For “test platform software forwarding-manager,” some subcommands allow locking aspects related to crimson. “test platform software infrastructure” allows disabling watchdogs and dropping debugging processes. “test platform software process” may kill processes, which can cause the box to reload.

A related approach involves leveraging coupled roots of trust for measurement. Both Linux and another operating system can act as root-of-trust for measurement. They can place information into a cryptoprocessor (e.g., TPM). For certain data types (e.g., a user logging onto the system), this may result in both roots-of-trust logging information acting as footprint evidence. The result is that these two roots-of-trust participate in the logging of coupled measurements which are ultimately visible to a security controller. Where there are discrepancies in underlying evidence from these two sources, it may indicate that an issue exists with a root-of-trust. That is, different roots-of-trust may be compared with each other.

Where roots of trust do not agree, there may be a security issue. A router’s base Linux operating system logs are not accessible to a casual user. In certain situations, it is possible to see these logs via the “request platform software system shell” command. But such access is not typical. And limiting/filtering such low level access is highly desired in secure systems. Raw Linux log data should not be exposed indiscriminately.

However, a limited subset of this Linux log information may be obtained another way. Comparing footprints from coupled-roots-of-trust can be performed if the Linux root-of-trust extends selected measurements to a cryptoprocessor (e.g., a TPM Platform Configuration Register (PCR)), and if measurements corresponding to what Linux placed into the TPM are always exposed by SYSLOG messages during non-compromised operations.

Normally Linux and SYSLOG report the same event. However, when there is a security compromise, measurements seen in Linux might not be reported to SYSLOG. As one example, consider the SYSLOG message "%BSHELL-6-SESSION-STARTED." This may be reported by SYSLOG during normal login. Therefore, SYSLOG may push the result to the security controller. However, such login messages may not be reported by the operating system to SYSLOG when a code injection exploit directly enables Linux shell access (bypassing operating system controls and logging).

The result is that a code injection exploit giving Linux shell access might result in a Linux log access footprint. And if there is no corresponding SYSLOG login, the TPM's PCR may be updated with a measurement which is not reflected in the SYSLOG being evaluated to see if the provided TPM PCR value is acceptable. Effectively, this will appear to security controllers as a near-real-time notification at the time of the hacker's login that log entries have been compromised, and the recorded SYSLOG information is incorrect.

Ultimately, contrasting information across Linux and the other operating system is only relevant where a differential is known to be a security indicator allowing a comparison of information from different roots-of-trust. In certain scenarios this may enable compromised (or bypassed) roots to be discovered by gaps in provided information.

Techniques described herein provide an automated feedback loop with a security controller to refine what to filter/extract from various platform logs for reporting into the cryptoprocessor. This is important as space in the cryptoprocessor is limited, and knowing the minimal set of events to log for any particular platform is essential. Effectively this process acts as a front-end to remove as many false-positive matches as is practical.

Determining whether unexpected code is exercised does not catch all types of unexpected code. Instead it catches just the unexpected code which happens to result in the emitting of a log message. This is a non-zero subset of attacks, and the corresponding fingerprints are explicitly known.

The security controller can continuously see what messages are being emitted by a specific platform. The security controller may be able to play a role in detecting new attack types. For example, if a new "%DATACORRUPTION-1-DATAINCONSISTENCY..." footprint started appearing in multiple routers where there has not been a recent software

upgrade may require more analysis to determine how large/small such generalized EEM filters would need to be to start to identify as-yet unfingerprinted attacks.

The footprints / indicators of compromise may be provided by subject matter experts. It is possible to use only explicit matching for known attacks. However, with reference to the aforementioned examples, messages starting with "%ILET-1-AUTHENTICATION FAIL" may be placed into a secured container except where a platform's configuration might make it a candidate for a known particular bug, which may only be seen in a very small subset of deployments.

When a bug is discovered, the set of impacted platforms and releases is closely tracked as the fix needs to be tightly articulated and communicated with the user. This may be automatable.

Logs can be erased, or pushing of entries delayed. Hardware validation of a small pre-filtered set of log evidence helps address this. The raw data from all relevant roots-of-trust may not be available (or streamable) to a log verifier. For example, Linux logs for a physical device might not be visible to a virtual router sitting thereon. However, if information is stored in a TPM, highly secured mechanisms may be implemented to allow a subset of security information to be passed from the physical device to the virtual device. This may apply even if the virtual device has no credentials to access the physical device. The raw data from all relevant roots-of-trust might not be real-time streamable to a log verifier. Pushing only the changing value of a PCR (e.g., via a YANG push) could be a trigger for a read of the log entries from the log generator. This eliminates the need for constant polling of specific logs, since the logs need only be acquired when there is a TPM PCR change.

In summary, techniques are provided herein to enhance detection of compromised network devices by maintaining a list of network device indicators of compromise (i.e., a footprint), determining the viable footprints relevant to the device's specific deployment context, and placing checks for these footprints onto a device. Based on the reduced set of footprints, these footprints are placed in a cryptoprocessor (e.g., TPM) to ensure that potentially relevant evidence cannot be silently discarded. As soon as a new footprint is characterized, devices may forward found instances of these footprints to a security

controller. This allows the security controller to do remediation well prior to the installation of fixes/patches.