Efficacy of UNIX HIDS

GIAC GCIA Gold Certification

Author: Janusz Pazgier, januszp@live.com
Advisor: Lenny Zeltser

Accepted: May 7, 2020

Abstract

There has been an increase in UNIX-based adversarial activity, as enterprises and users shift towards the platform (WatchGuard, 2017). The focus of this paper is to demonstrate the effectiveness of three separately installed host-based intrusion detection systems (HIDS): OSSEC, Samhain, and Auditd, and their ability to detect specific MITRE ATT&CK tactics. Custom scripts implement the ATT&CK tactics of privilege escalation, persistence, and data exfiltration. The goal is to inform security professionals about the pros and cons of implementing each of these HIDS.

The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author.

Approved for Public Release; Distribution Unlimited. Public Release Case Number 20-1273

1. Introduction

If one is unfamiliar with a host-based intrusion detection system (HIDS) like OSSEC, Samhain, and Auditd, then one should reference an introductory article called *Intrusion Detection Systems: Definition, Need and Challenges* (Sarmah, 2001). In general, a HIDS serves as an intrusion detection system that monitors and collects data from a target system that should be protected, processes and correlates the gathered information, and initiates responses when evidence of an intrusion is detected (Giovanni Vigna, 2005). As claimed by Avast, "an IDS is focused on detection and is normally a passive system (doesn't take action on the detected threat), an IDS is rarely used as a standalone system. It's many times, coupled with a related system called an intrusion prevention system that actually takes action on the perceived threat" (Avast, 2020).

The MITRE ATT&CK framework is a globally accessible knowledge base of adversary tactics and techniques based on real-world observations (The MITRE Corporation, 2020). The framework consists of tactics like privilege escalation, persistence, and data exfiltration. Each of these tactics includes techniques that dig deeper into methods that adversaries use to perform the specific attacks. The ATT&CK framework provides granular detail of adversary behavior and is complementary to many other models and frameworks (The MITRE Corporation, 2020). This research reproduces these three adversarial tactics through custom scripts and demonstrates the efficacy of HIDS at detecting them.

This paper will focus on comparing the functionality of HIDS at detecting various types of attacks based on the MITRE ATT&CK framework, through the installation and configuration of three opensource host-based intrusion detection systems: OSSEC, Samhain and Auditd. In-depth installation and configuration details are provided to illustrate how the various parts of the HIDS interact with each other. The functionality of the HIDS is tested through analysis of alerts generated when executing different scripts that mock the functionality of tactics and techniques defined in the MITRE ATT&CK framework.

2. Pre-requisites

For the testing performed throughout, a Dell R710 server was configured with the hypervisor VMware ESXi installed. This server has 16 physical cores, 96 GB of RAM, and 2 TB of storage, allowing for the proper designation of resources for the preconfigured environment. Building and maintaining a virtual environment to perform these tests allowed for isolated testing of the features of the HIDS.

2.1. Setup

There are several versions of UNIX-based operating systems, including Linux (Lackmann, 2010). The tools outlined in this paper will be installed and configured on a CentOS 7 Linux x64 everything image with the GNOME desktop installed via the GUI installation of the operating system. A GNOME desktop installation is used instead of a server with a GUI to mimic the applicability of the HIDS at detecting attack methods on an end-user system.

To verify that all the packages on the system are up to date, the command "`yum install epel-release && yum update`" was run on the hosts. Next, the development tools package was installed on the hosts to make sure that the pre-requisite packages for installing the HIDS are compiled and run properly. These packages are installed with the command '`yum groupinstall "Development tools."`' Finally, the three scripts used for testing the ATT&CK framework tactics were uploaded to the Documents directory of the test (called "sans") user, which was added to the system. These scripts will aid in verifying the effectiveness of the HIDS at detecting various tactics and techniques.

Upon installing the pre-requisite packages, and moving the files to the Documents directory, a snapshot (saved state of the system) of the image is created as a backup in case something goes wrong in the testing. Next, the created snapshot is cloned (copied) into three separate virtual machines. It is beneficial to us three separate virtual machines for this testing because it allows for a standardized environment when installing the three different HIDS. Furthermore, it decreases the chances of cross-contaminated resources, rules, and logs.

2.2. Host-based Intrusion Detection Systems

To demonstrate the efficacy of Linux HIDS, the three tools (OSSEC, Samhain, and Auditd) were installed and configured to detect the adversarial tactics described in section 2.3. These HIDS were best suited for this research because they are open-source and free to install, configure, and use on Linux systems. Additionally, the tools provide similar core functionality, with differentiating additional features. A common approach to detecting intrusions on a system is file integrity monitoring (FIM). On UNIX-based operating systems, everything is a file. Therefore, FIM can go a long way when determining the inappropriate use and potential persistence implanted on a file system. Another method used to detect possible compromise on a Linux system is through monitoring of connections from the host, which could be exfiltration attempts of adversarial. Both are conventional approaches to detecting intrusions on a system and will be configured based on the documentation from the vendors of the tools.

2.3. Scripts

To compare the efficiency of the three HIDS at detecting adversarial tactics and techniques, the author developed most of the scripts in this research to mimic adversarial activities on the devices and to verify a standardized form of execution. Each of the techniques defined under a tactic in the ATT&CK framework includes examples of how adversarial use. Each script implements methodologies outlined in these definitions and examples. The Appendix contains these scripts and execution instructions.

2.3.1. Privilege Escalation

Privilege escalation is a tactic that consists of techniques that adversaries use to gain higher-level permissions on a system or network (The MITRE Corporation, n.d.). In his blog *Privilege escalation with setuid*, Julien Mourer created a program that demonstrates privilege escalation through misconfigured file configuration. This form of misconfiguration occurs when a system administrator provides non-wheel user access to a file that is owned by a user in the wheel group. Once compiled and executed on a host, it gives root shell access to an unprivileged user.

2.3.2. Persistence

Persistence is a tactic that an adversary uses to create ways to regain access into a system post-exploitation. Many of these techniques allow a user to make small modifications to the UNIX system to create a backdoor for future access. Appendix B contains the script designed to demonstrate the tactic of persistence through the creation of additional accounts, modification of file system permissions (misconfigured attribution), and .bashrc persistence. This script is a modification of a malicious script the author used during a cybersecurity competition. Once executed, the script creates a new user, modified the attribution of a file, and adds malicious lines to `/etc/bashrc`. In conclusion, the designed script creates a new user, modifies a files known association, and implements a form of .bashrc persistence.

2.3.3. Data Exfiltration

Data exfiltration consists of techniques that adversaries may use to steal data from a network (MITRE Corporation, 2019). Appendix C contains a script that mimics the tactic of exfiltration over an alternative protocol. Upon execution, the opens a new port on the system and waits for an external party to connect to it to download data.

3. OSSEC

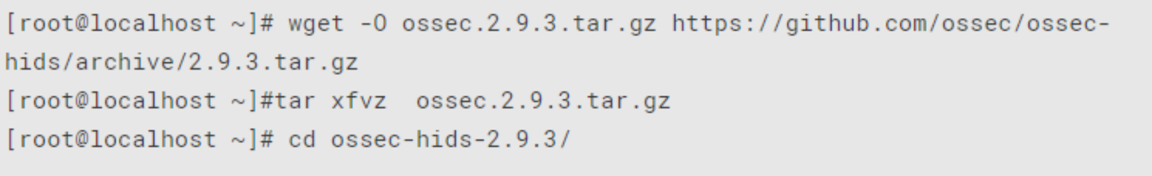
OSSEC is a database file integrity monitoring host-based intrusion detection system, which also serves as a log aggregator (OSSEC, 2020). Once configured and initialized, OSSEC generates a knowledge base of the known good state of a system. It uses this database to perform integrity checks on the filesystem to determine potentially unwanted modified files. On top of the FIM feature, OSSEC is capable of detecting rootkits implanted in known locations and performing real-time alerting. OSSEC contains a plethora of additional features and functionalities, which do not apply to this research.

3.1. Setup

3.1.1. Installation

The OSSEC website contains well-documented installation and configuration instructions for the tool. There are many ways to install and configure this tool for any environment. OSSEC can run in a server/agent environment where a server can manage thousands of nodes at a time. The focus of this research is on installing and configuring a local version of OSSEC. A local version has the same HIDS functionality; however, all remote services are disabled as the tool is just monitoring the local system.

The first step is to download the OSSEC tool from the source and unpack it. This can be done using the following commands:



```
[root@localhost ~]# wget -O ossec.2.9.3.tar.gz https://github.com/ossec/ossec-
hids/archive/2.9.3.tar.gz
[root@localhost ~]#tar xfvz  ossec.2.9.3.tar.gz
[root@localhost ~]# cd ossec-hids-2.9.3/
```

The OSSEC installation directory contains the `./install.sh` script, which prompts the user for installation parameters. These parameters include client/server, local version, and features to enable. Once executed and provided the proper input, the script will install and activate the features. However, if invalid information is provided at this step, then a full reinstall of the tool is required. For this research, all the functions related to FIM are installed to standardize the installation across the HIDS.

The install type selected is `local`, as OSSEC is running standalone without a server/client infrastructure. Next, the `integrity checking daemon` feature is enabled to perform FIM on the filesystem. In addition, `Rootkit detection` is installed to aid in detecting misconfigured SUIDs. Finally, the `active response` feature is disabled as the research focuses on passive monitoring of the system. These selected features enabled OSSEC as a tool that monitors the integrity of the system and reports any probable anomalies.

3.1.2. Rule creation

When configuring rules and alerts on OSSEC, one must be acquainted with the location of the OSSEC configuration files and their purpose. The primary storage location for all the tools files is `/var/ossec`. This directory also contains the bin/ directory, which contains all the executable of OSSEC, the `etc/` directory, which contains all the configuration files, and the rules/ directory, which contains rule configurations. To configure the functionality of the features in OSSEC, the `/etc/ossec.conf` needs to be modified.

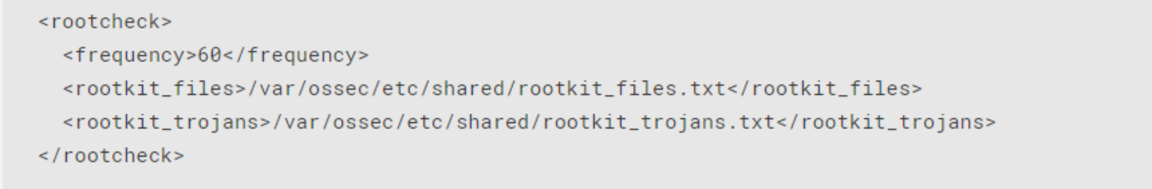
Upon performing a modification to either the `ossec.conf` or `local_rules.xml`, OSSEC requires a restart with the command "`var/ossec/bin/ossec-control restart`." Once restarted, OSSEC requires a few minutes to generate an FIM database and then will perform integrity checks based on the frequency defined in syscheck. Syscheck is the name of the integrity checking process inside OSSEC (OSSEC, 2020). By default, syscheck performs every 22 hours on a system. For this research, syscheck scans the operating system every 60 seconds.

3.2. Privilege Escalation

3.2.1. Rules created

This HIDS has a feature called Rootkit detection, which will help detect the privesc script, which has a misconfigured SUID bit set. According to the writers of the tool, the tool can "scan the whole filesystem looking for unusual files and permission problems. Files owned by root, with write permission to non-root users, are generally malicious, and the rootkit detection will look for them. The OSSEC Project also states, SUID files, hidden directories, and files will also be inspected" (OSSEC Project, 2020).

By default, the `rootcheck` functionality of the rootkit detection feature is commented out in the `ossec.conf` file. To enable `rootcheck`, the lines shown below need to be uncommented.

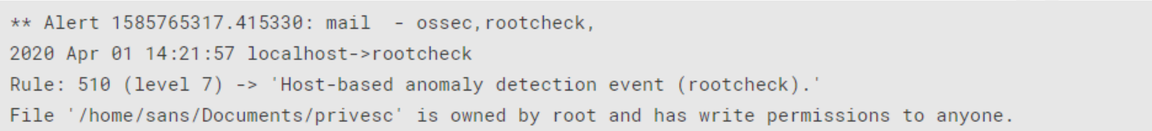


```
<rootcheck>
  <frequency>60</frequency>
  <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>
  <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootkit_trojans>
</rootcheck>
```

By default, `rootcheck` scans the system every 10 hours. However, this frequency is shortened to every 60 seconds to speed up the generation of the expected results.

3.2.2. Alerts generated

After enabling the `rootcheck` feature in the OSSEC config file, restarting OSSEC, and waiting approximately 30 minutes for OSSEC to thoroughly scan the system, the following alert was discovered:



```
** Alert 1585765317.415330: mail  - ossec,rootcheck,
2020 Apr 01 14:21:57 localhost->rootcheck
Rule: 510 (level 7) -> 'Host-based anomaly detection event (rootcheck).'
File '/home/sans/Documents/privesc' is owned by root and has write permissions to anyone.
```

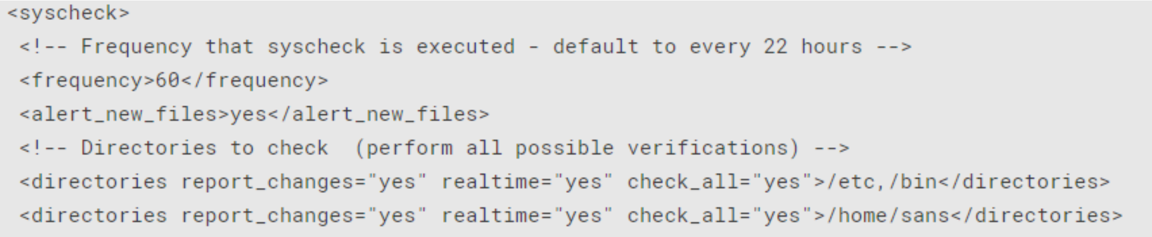
This alert demonstrates the rootcheck ability to detect files with misconfigured attributes with ease. Any additional files with misconfigured permissions would cause a similar alert as long as OSSEC was running on the system. This alert will not help in identifying a user that misused this file but can help identify misconfigured files on the operating system.

3.3. Persistence

3.3.1. Rules created

To detect the persistence in our persistence.sh script the modified files will be monitored when creating a new user (`/etc/passwd` and `/etc/shadow`), and any modifications created to the `/etc/bashrc` file. These forms of persistence are detected by verifying the `/etc/` directory in real-time against a database of the known good state of the system.

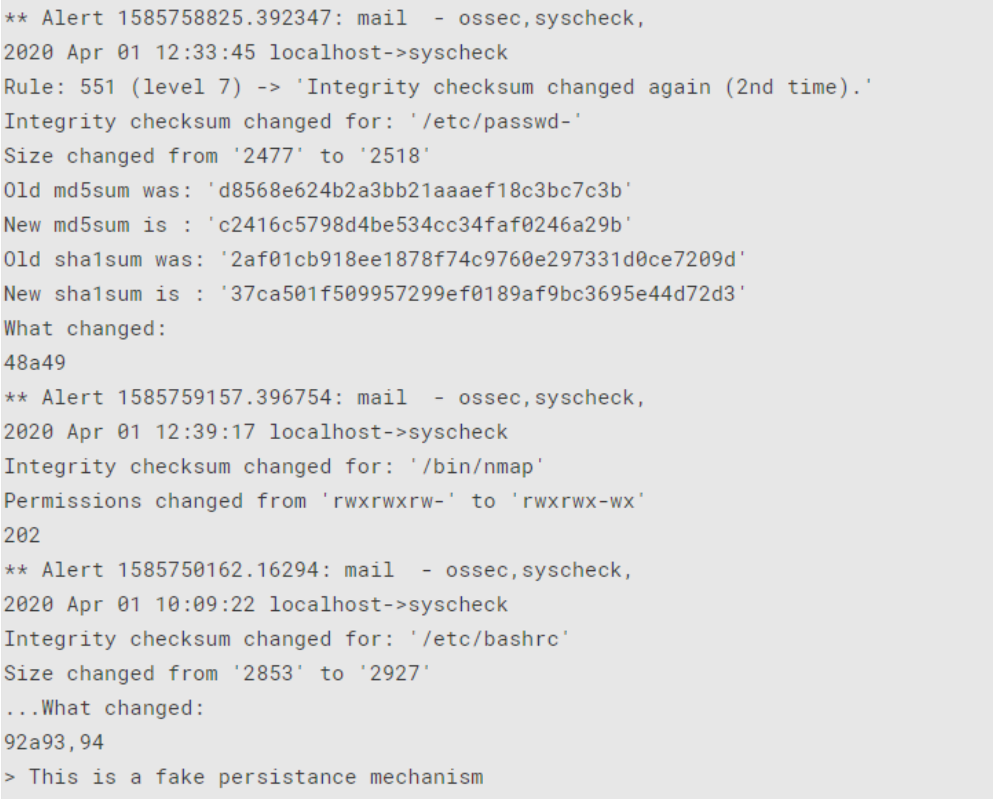
Additionally, the `/bin/` directory will be monitored for any modification to the binaries on the system. In OSSEC this verification can be enabled by making the following changes to the `/var/ossec/etc/ossec.conf` configuration file.



```
<syscheck>
 <!-- Frequency that syscheck is executed - default to every 22 hours -->
 <frequency>60</frequency>
 <alert_new_files>yes</alert_new_files>
 <!-- Directories to check  (perform all possible verifications) -->
 <directories report_changes="yes" realtime="yes" check_all="yes">/etc,/bin</directories>
 <directories report_changes="yes" realtime="yes" check_all="yes">/home/sans</directories>
```

3.3.2. Alerts generated

OSSEC generated the warning below after the execution of ./persistance.sh on the operating system. The tool detected the modified files, provided the location of the modified files, compared the hash (md5sum), and the contents that were changed (Note: Appendix D contains the full output):

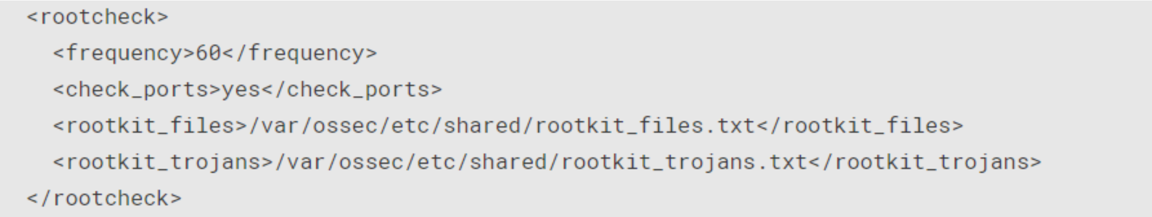


```
** Alert 1585758825.392347: mail  - ossec,syscheck,
2020 Apr 01 12:33:45 localhost->syscheck
Rule: 551 (level 7) -> 'Integrity checksum changed again (2nd time).'
Integrity checksum changed for: '/etc/passwd-'
Size changed from '2477' to '2518'
Old md5sum was: 'd8568e624b2a3bb21aaaef18c3bc7c3b'
New md5sum is : 'c2416c5798d4be534cc34faf0246a29b'
Old sha1sum was: '2af01cb918ee1878f74c9760e297331d0ce7209d'
New sha1sum is : '37ca501f509957299ef0189af9bc3695e44d72d3'
What changed:
48a49
** Alert 1585759157.396754: mail  - ossec,syscheck,
2020 Apr 01 12:39:17 localhost->syscheck
Integrity checksum changed for: '/bin/nmap'
Permissions changed from 'rwxrwxrw-' to 'rwxrwx-wx'
202
** Alert 1585750162.16294: mail  - ossec,syscheck,
2020 Apr 01 10:09:22 localhost->syscheck
Integrity checksum changed for: '/etc/bashrc'
Size changed from '2853' to '2927'
...What changed:
92a93,94
> This is a fake persistance mechanism
```

3.4. Data Exfiltration

3.4.1. Rules created

In addition to checking the files on the system for misconfigured SUIDs, the rootcheck feature can check for open ports. The port checking feature can be enabled by adding check_port to yes in the ossec.conf file, as seen below:

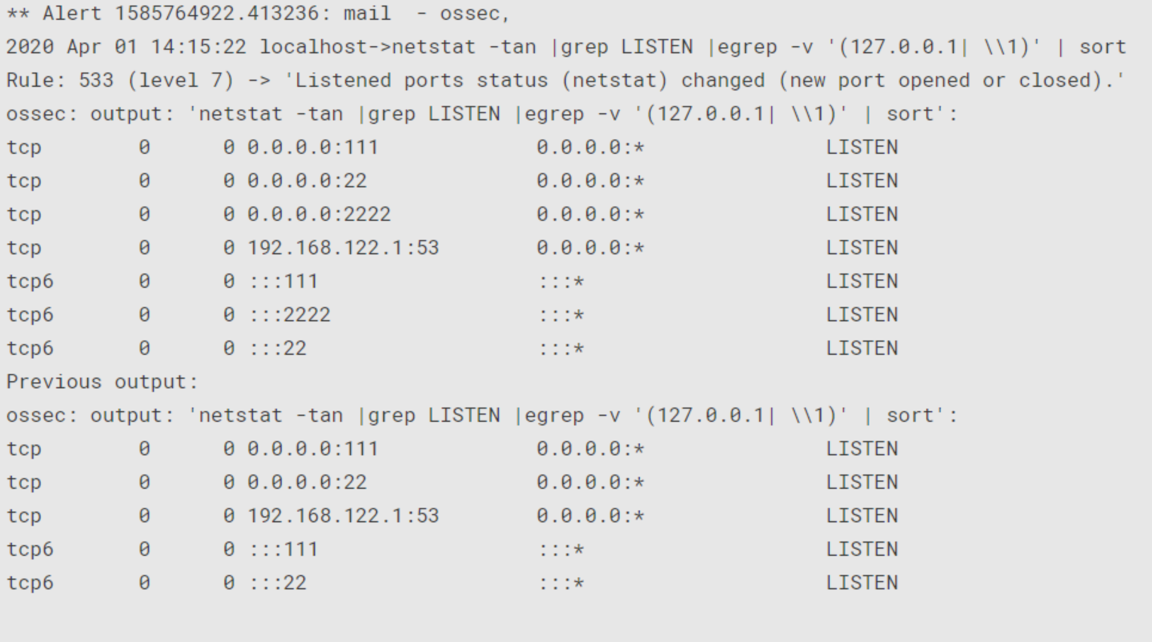


```
 <rootcheck>
   <frequency>60</frequency>
   <check_ports>yes</check_ports>
   <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>
   <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootkit_trojans>
 </rootcheck>
```

This configuration change will monitor the output of netstat for any ports that are open and will generate an alert when a new port is listening on the system. A nonstandard open port on an operating system could function as a means of exfiltrating data over an alternative protocol.

3.4.2. Alerts generated

As stated above, OSSEC will take the known state of open ports on a system and compare it to the current state to find anomalies. In this example, OSSEC determined that port 2222 is a newly opened port and alerted on it.



```
** Alert 1585764922.413236: mail  - ossec,
2020 Apr 01 14:15:22 localhost->netstat -tan |grep LISTEN |egrep -v '(127.0.0.1| \\1)' | sort
Rule: 533 (level 7) -> 'Listened ports status (netstat) changed (new port opened or closed).'
ossec: output: 'netstat -tan |grep LISTEN |egrep -v '(127.0.0.1| \\1)' | sort':
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::2222                 :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
Previous output:
ossec: output: 'netstat -tan |grep LISTEN |egrep -v '(127.0.0.1| \\1)' | sort':
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
```

The functionality in detecting communication over alternative protocol is something that is lacking in OSSEC. The detection of this adversarial behavior should be implemented through a network intrusion detection system that can generate a known good state of the traffic on a network and determine anomalies from that dataset.

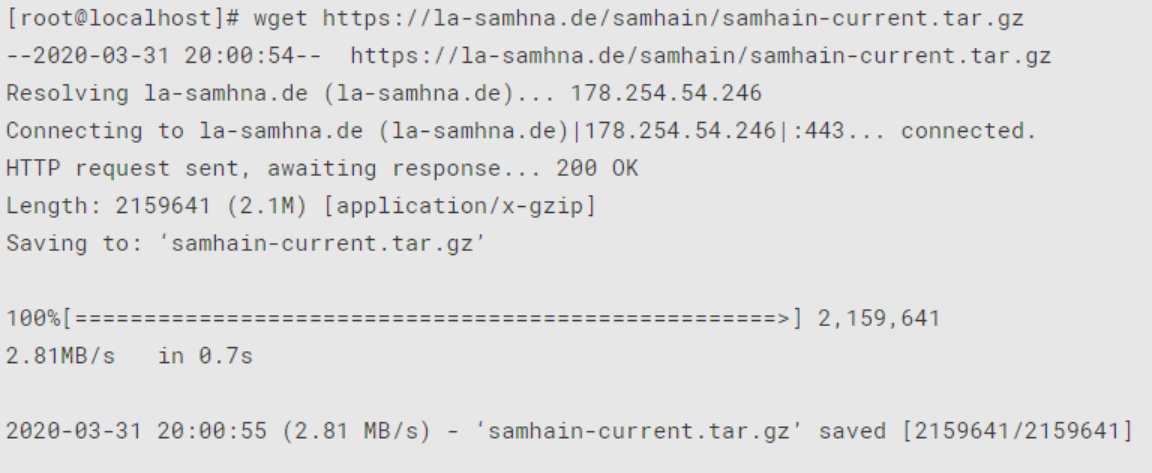
4. Samhain

Samhain is easy to install and configure for POSIX systems. This HIDS provides file integrity checking, log file monitoring/analysis, as well as rootkit detection, port monitoring, and detection of rogue SUID executables (Wichmann, 2006). According to another study performed by a SANS student, "Implementing tools like Samhain can provide organizations with the confidence that data is not modified in any unauthorized manner and help cover any regulatory compliances" (Nel, 2014).

4.1. Setup

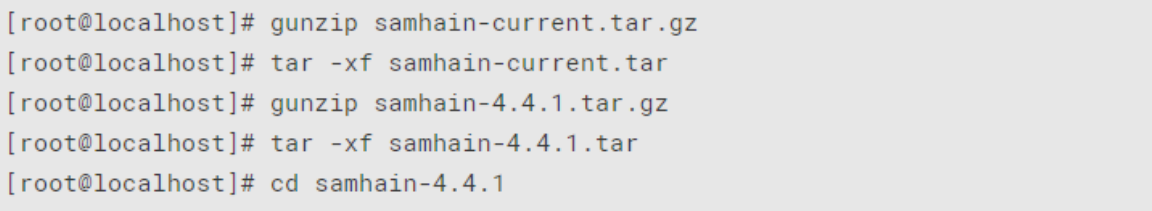
4.1.1. Installation

The command shown below downloads the latest version of Samhain from the website:



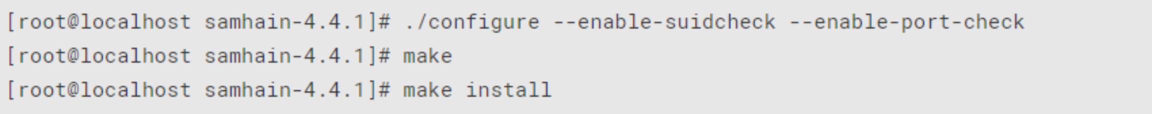
```
[root@localhost]# wget https://la-samhna.de/samhain/samhain-current.tar.gz
--2020-03-31 20:00:54--  https://la-samhna.de/samhain/samhain-current.tar.gz
Resolving la-samhna.de (la-samhna.de)... 178.254.54.246
Connecting to la-samhna.de (la-samhna.de)|178.254.54.246|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2159641 (2.1M) [application/x-gzip]
Saving to: ‘samhain-current.tar.gz’

100%[===================================================>] 2,159,641
2.81MB/s   in 0.7s

2020-03-31 20:00:55 (2.81 MB/s) - ‘samhain-current.tar.gz’ saved [2159641/2159641]
```

Next, the instructions on the website require the unpacking of the downloaded folder with the following commands:



```
[root@localhost]# gunzip samhain-current.tar.gz
[root@localhost]# tar -xf samhain-current.tar
[root@localhost]# gunzip samhain-4.4.1.tar.gz
[root@localhost]# tar -xf samhain-4.4.1.tar
[root@localhost]# cd samhain-4.4.1
```

Similarly to OSSEC, Samhain is installed and configured through the execution of an install script. For this research, suidcheck is enabled to detect any files with misconfigured SUID and GID bits. Additionally, port-check is enabled to detect unknown open ports on the system. The command below installs Samhain with these features enabled:



```
[root@localhost samhain-4.4.1]# ./configure --enable-suidcheck --enable-port-check
[root@localhost samhain-4.4.1]# make
[root@localhost samhain-4.4.1]# make install
```

The Samhain installation completes successfully after "`make install`" concludes.

Note: The procedures outlined above install Samhain as a standalone appliance. Samhain can be installed and configured to run as a server/agent environment where it can monitor multiple hosts with potentially different operating systems; however, for this paper, it was installed locally.

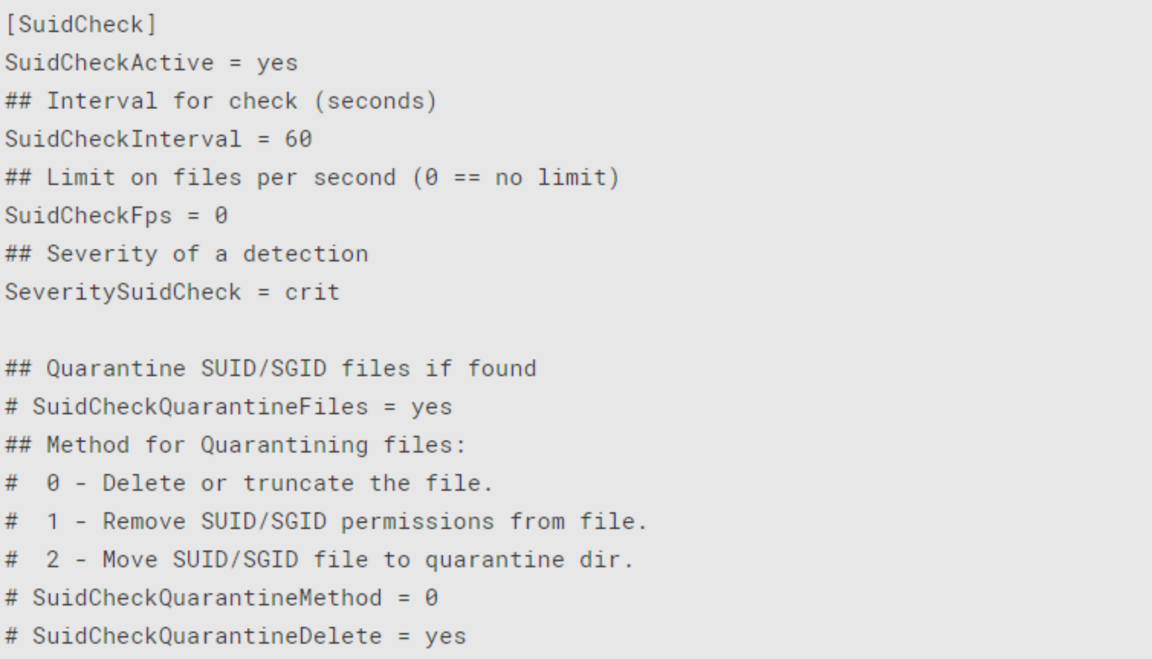
4.1.2. Rule creation

The configuration file for Samhain is located at `/etc/samhainrc`, and by default is commented out so that it can function on a large assortment of operating systems. To limit false positives, only the sections about detecting the adversarial activities generated by the scripts listed in section 2.3 will be uncommented (enabled). Upon performing an edit to the `/etc/samhainrc` configuration file, the Samhain database must be initialized through the execution of the command: "`./samhain -t init`." Once created, the database can be found in the directory "`/var/lib/samhain/samhain_file`." According to the tool's documentation, "One should delete or rename the baseline database file if they want to run "`samhain -t init`" a second time" (Wichmann, 2006).

4.2. Privilege Escalation

4.2.1. Rules created

Samhain has a set of features for detecting files with misconfigured setuid and setSGID bits. It is not only capable of alerting when it detects such a file, but it can also be configured to alert, quarantine, or delete the file in question. The following are the changes made to the `/etc/samhainrc`:

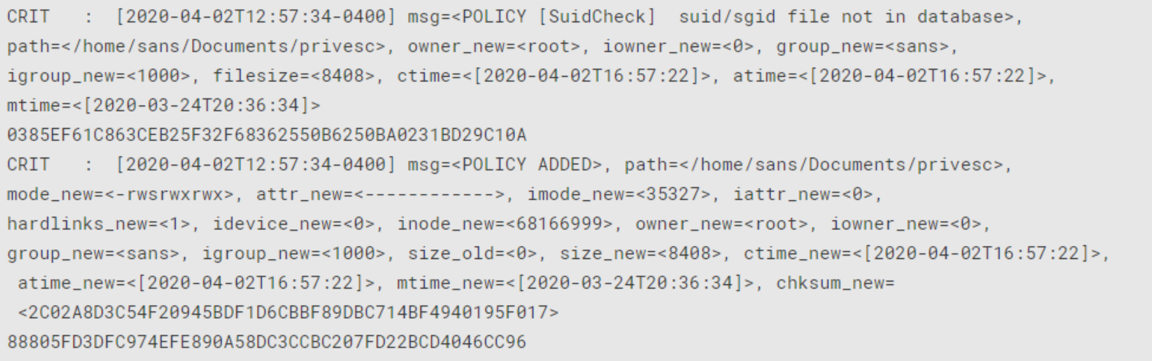


```
[SuidCheck]
SuidCheckActive = yes
## Interval for check (seconds)
SuidCheckInterval = 60
## Limit on files per second (0 == no limit)
SuidCheckFps = 0
## Severity of a detection
SeveritySuidCheck = crit

## Quarantine SUID/SGID files if found
# SuidCheckQuarantineFiles = yes
## Method for Quarantining files:
#  0 - Delete or truncate the file.
#  1 - Remove SUID/SGID permissions from file.
#  2 - Move SUID/SGID file to quarantine dir.
# SuidCheckQuarantineMethod = 0
# SuidCheckQuarantineDelete = yes
```

As can be seen in line two, the suid checker was activated, and the interval changed to every 60 seconds. Configuring Samhain to such a low interval is not recommended for production systems; however, for this study, it allows for quicker testing. Next, the limit of files checked per second was configured to infinite to allow for a more immediate examination of the entire filesystem. This value should be limited in a production environment. Further, the `SeveritySuidCheck` is set to crit, to classify the incident accordingly and for easier identification in the log file. Finally, prevention methods like quarantining are left commented out to limit the functionality of Samhain to alerting on the incident.

4.2.2. Alerts generated

Samhain generated the alert below after placing the privilege escalation script in the Documents folder of the test user and executing the script per the steps listed in Appendix A. Upon scanning the filesystem, Samhain generated an alert upon discovering a file with a misconfigured SUID.



```
CRIT   :  [2020-04-02T12:57:34-0400] msg=<POLICY [SuidCheck]  suid/sgid file not in database>,
path=</home/sans/Documents/privesc>, owner_new=<root>, iowner_new=<0>, group_new=<sans>,
igroup_new=<1000>, filesize=<8408>, ctime=<[2020-04-02T16:57:22]>, atime=<[2020-04-02T16:57:22]>,
mtime=<[2020-03-24T20:36:34]>
0385EF61C863CEB25F32F68362550B6250BA0231BD29C10A
CRIT   :  [2020-04-02T12:57:34-0400] msg=<POLICY ADDED>, path=</home/sans/Documents/privesc>,
mode_new=<-rwsrwxrwx>, attr_new=<------------>, imode_new=<35327>, iattr_new=<0>,
hardlinks_new=<1>, idevice_new=<0>, inode_new=<68166999>, owner_new=<root>, iowner_new=<0>,
group_new=<sans>, igroup_new=<1000>, size_old=<0>, size_new=<8408>, ctime_new=<[2020-04-02T16:57:22]>,
 atime_new=<[2020-04-02T16:57:22]>, mtime_new=<[2020-03-24T20:36:34]>, chksum_new=
 <2C02A8D3C54F20945BDF1D6CBBF89DBC714BF4940195F017>
88805FD3DFC974EFE890A58DC3CCBC207FD22BCD4046CC96
```

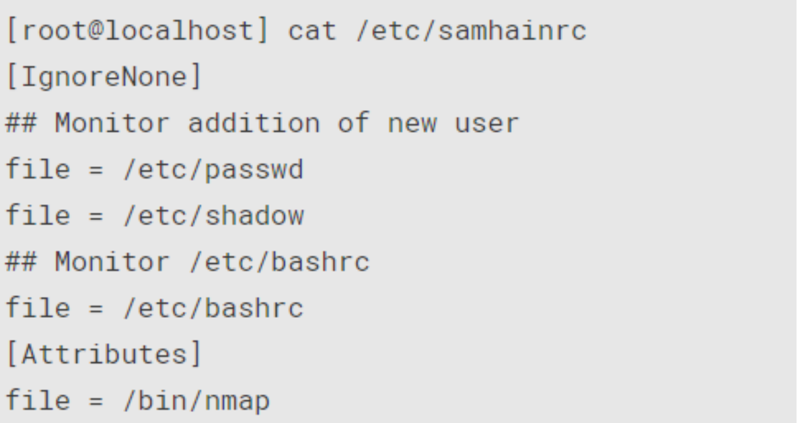
The alert above demonstrates Samhain’s ability to check the filesystem for misconfigured files continuously. Additionally, with the enablement of the `SuidCheckQuarentineFiles` feature, Samhain would quarantine this file and potentially remove it from an adversary's control.

4.3. Persistence

4.3.1. Rules created

One of the main functionalities of Samhain is File integrity monitoring. The `/etc/samhainrc` configuration file contains fields that allow the specifications of files and directories to monitor for file creation, modification, execution, and attribution. To detect the `persistence` script, the `[IgnoreNone]` feature of Samhain will be implemented, which tracks all changes, including access times of the files specified.

Additionally, the `[Attribution]` feature of Samhain is enabled, which will only monitor the change in the attribution of files defined. These changes are below:

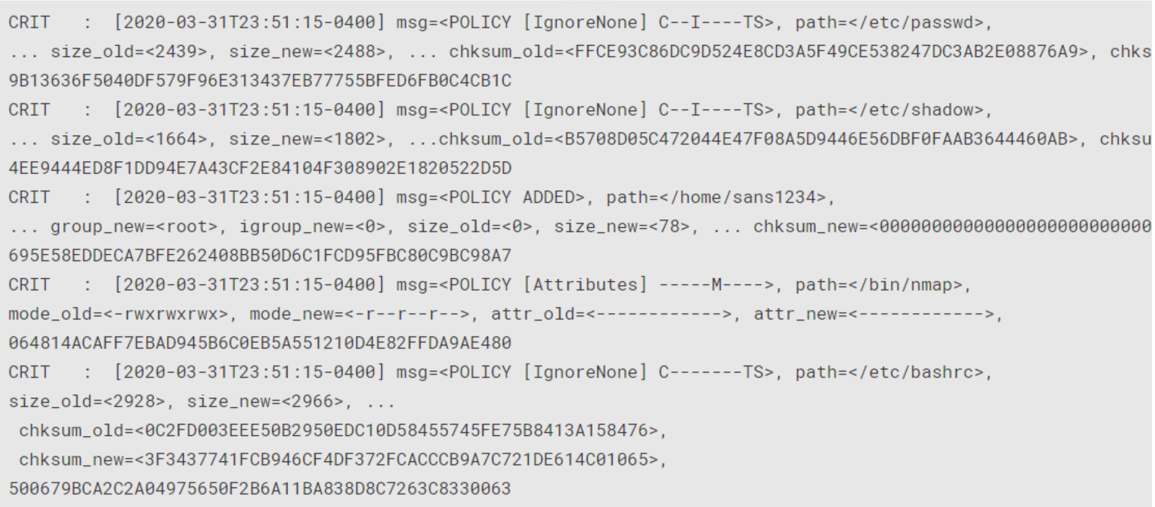


```
[root@localhost] cat /etc/samhainrc
[IgnoreNone]
## Monitor addition of new user
file = /etc/passwd
file = /etc/shadow
## Monitor /etc/bashrc
file = /etc/bashrc
[Attributes]
file = /bin/nmap
```

In the configuration snippet above, enables the monitoring of all changes to `/etc/passwd` and `/etc/shadow` files. These files are essential because any modification of these files implies the creation of a user on the system. Additionally, all changes to `/etc/bashrc` are tracked. Finally, only the monitoring of `/etc/nmap` is limited to attribution because the goal is to follow a misconfiguration of file permissions.

4.3.2. Alerts generated

Samhain generated the below alerts upon execution of the persistence script on the CentOS 7 system (Note: Appendix E contains the full output):



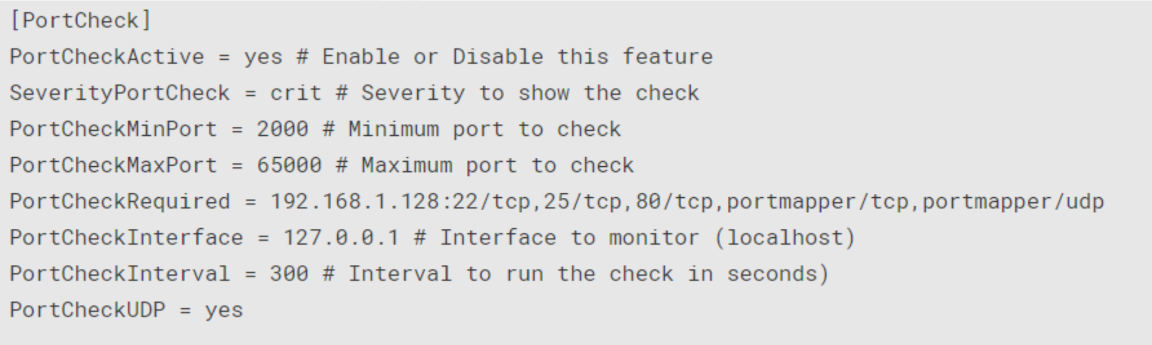
```
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY [IgnoreNone] C--I----TS>, path=</etc/passwd>,
... size_old=<2439>, size_new=<2488>, ... chksum_old=<FFCE93C86DC9D524E8CD3A5F49CE538247DC3AB2E08876A9>, chks
9B13636F5040DF579F96E313437EB77755BFED6FB0C4CB1C
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY [IgnoreNone] C--I----TS>, path=</etc/shadow>,
... size_old=<1664>, size_new=<1802>, ...chksum_old=<B5708D05C472044E47F08A5D9446E56DBF0FAAB3644460AB>, chksu
4EE9444ED8F1DD94E7A43CF2E84104F308902E1820522D5D
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY ADDED>, path=</home/sans1234>,
... group_new=<root>, igroup_new=<0>, size_old=<0>, size_new=<78>, ... chksum_new=<0000000000000000000000000
695E58EDDECA7BFE262408BB50D6C1FCD95FBC80C9BC98A7
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY [Attributes] -----M---->, path=</bin/nmap>,
mode_old=<-rwxrwxrwx>, mode_new=<-r--r--r-->, attr_old=<------------>, attr_new=<------------>,
064814ACAFF7EBAD945B6C0EB5A551210D4E82FFDA9AE480
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY [IgnoreNone] C-------TS>, path=</etc/bashrc>,
size_old=<2928>, size_new=<2966>, ...
 chksum_old=<0C2FD003EEE50B2950EDC10D58455745FE75B8413A158476>,
 chksum_new=<3F3437741FCB946CF4DF372FCACCCB9A7C721DE614C01065>,
500679BCA2C2A04975650F2B6A11BA838D8C7263C8330063
```

Samhain created an alert when the `/etc/passwd` and `/etc/shadow` file was modified. It did not alert on what modification was changed to the file; however, it was able to determine a difference between the known good state checksum and the new one. Additionally, it alerted that a new user `/home/` directory was added for a new user called `sans1234`. Next, Samhain alerted that the file permissions for `/bin/nmap` were changed and that the `/etc/bashrc` file was also modified in some way. Although these alerts were easy to configure, they did not generate enough verbosity to quickly determine what type of modifications were made to the files, only that changes were made.

4.4. Data Exfiltration

4.4.1. Rules created

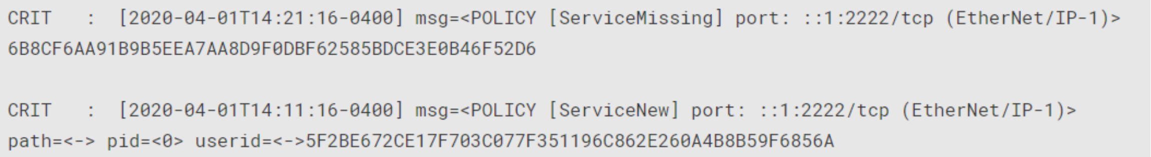
An additional feature of Samhain is the ability to monitor the open ports of the system. This ability allows possible data exfiltration through alternative methods and identification of regular operating ports from abnormal ports as an alert. This can be performed in Samhain by enabling the [PortCheck] feature and specifying the port range to check, the required ports, the interface to check on, and the interval to perform the checks. As seen in the samhainrc configuration below, the default required ports were unmodified; however, the min and max port value to check was modified to be from 2000 to 65000. Additionally, the interface to monitor was modified to localhost (127.0.0.1) to enable Samhain to watch all internet interfaces on the system. Finally, the interval to check ports was changed to 5 minutes:



```
[PortCheck]
PortCheckActive = yes # Enable or Disable this feature
SeverityPortCheck = crit # Severity to show the check
PortCheckMinPort = 2000 # Minimum port to check
PortCheckMaxPort = 65000 # Maximum port to check
PortCheckRequired = 192.168.1.128:22/tcp,25/tcp,80/tcp,portmapper/tcp,portmapper/udp
PortCheckInterface = 127.0.0.1 # Interface to monitor (localhost)
PortCheckInterval = 300 # Interval to run the check in seconds)
PortCheckUDP = yes
```

4.4.2. Alerts generated

After executing the data exfiltration script on this CentOS 7 machine that is running Samhain, the alerts below were generated based on the changes to samhainrc configuration file:



```
CRIT   :  [2020-04-01T14:21:16-0400] msg=<POLICY [ServiceMissing] port: ::1:2222/tcp (EtherNet/IP-1)>
6B8CF6AA91B9B5EEA7AA8D9F0DBF62585BDCE3E0B46F52D6

CRIT   :  [2020-04-01T14:11:16-0400] msg=<POLICY [ServiceNew] port: ::1:2222/tcp (EtherNet/IP-1)>
path=<-> pid=<0> userid=<->5F2BE672CE17F703C077F351196C862E260A4B8B59F6856A
```

The two CRIT warnings that were generated above show that a new TCP connection at port 2222 was created. The first alert warns that no service is linked to the newly created port, while the second alert states that a new service was created with that port. Additionally, Samhain is alerting that there is a port open to receive data, and that data was sent over the same port. This alert is useful in informing security professionals that an attempt was made to exfiltrate data out of the system through a specific port.

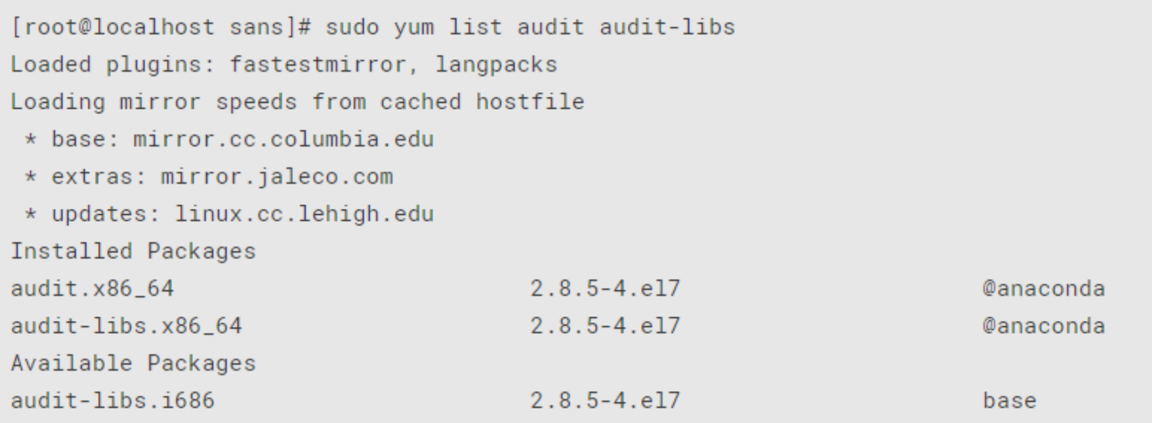
5. Auditd

Auditd is an essential feature to the Linux Kernel that collects system activity and generates alerts based on predetermined rules. Red Hat concurs when they note, "due to its close integration with the kernel, Auditd is capable of monitoring System calls, file access, and modifications as well as a list of preconfigured auditable events that are maintained by RHEL" (Redhat, n.d.).

5.1. Setup

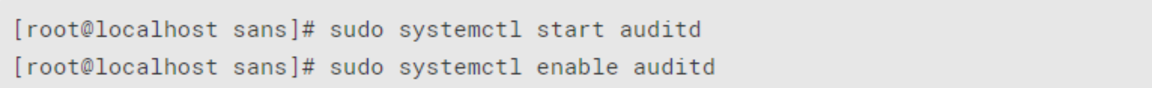
5.1.1. Installation

Auditd comes preconfigured on our version of CentOS 7, and therefore the installation is already done. However, to verify that all required packages are installed on the system, it is advised to run the following:



```
[root@localhost sans]# sudo yum list audit audit-libs
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.cc.columbia.edu
 * extras: mirror.jaleco.com
 * updates: linux.cc.lehigh.edu
Installed Packages
audit.x86_64                          2.8.5-4.el7                     @anaconda
audit-libs.x86_64                     2.8.5-4.el7                     @anaconda
Available Packages
audit-libs.i686                       2.8.5-4.el7                     base
```

Once all the packages are installed and updated to the version displayed above, then Auditd can be started and enabled on startup. As seen below:

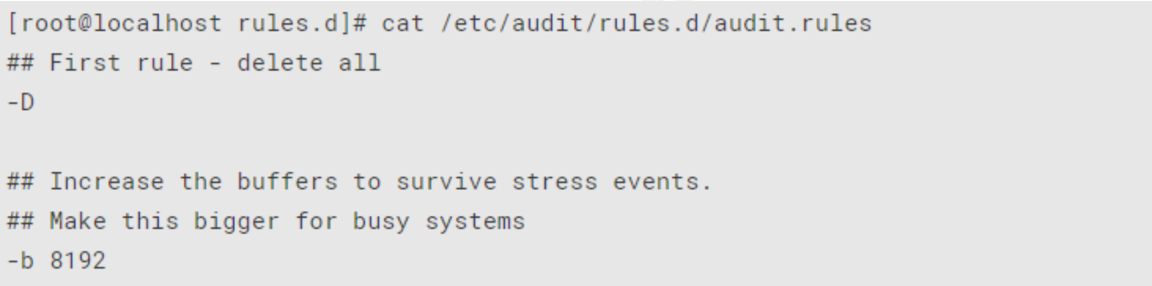


```
[root@localhost sans]# sudo systemctl start auditd
[root@localhost sans]# sudo systemctl enable auditd
```

Once Auditd is installed, the first rules can be written to alert on malicious activities on the system.

5.1.2. Rule creation

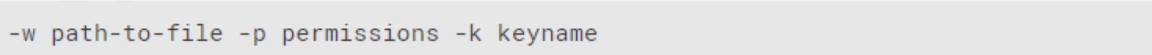
The working directory of Auditd is "`/etc/audit/`," and is the location of the audit configuration file (`auditd.conf`). According to the man page of Auditd, "the audit daemon itself has some configuration options that the admin may wish to customize. They are found in the `auditd.conf` file" (Grubb, 2020). For this research, no changes were made to this file. The rules that the audit daemon uses for alerts can be found in "`/etc/audit/rules.d/audit.rules`."



```
[root@localhost rules.d]# cat /etc/audit/rules.d/audit.rules
## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192
```

This file contains preconfigured default rules for auditd. Additional rules were written for detecting privilege escalation, persistence, and data exfiltration.

The creation of Auditd rules requires an understanding of rule structure, flags, and values. In Auditd, the rule structure for interacting with the filesystem is:

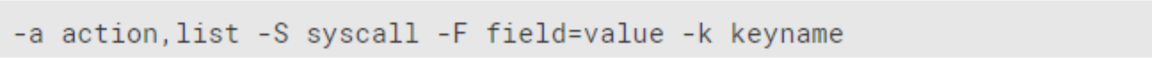


```
-w path-to-file -p permissions -k keyname
```

In the filesystem rule structure, the permission flags are any of the following (Grubb, AUDIT.RULES, n.d.):

- r: read of the file
- w: write to the file
- x: execute the file
- a: change the file attribute

Additionally, the auditd rule structure for interacting with system calls is:



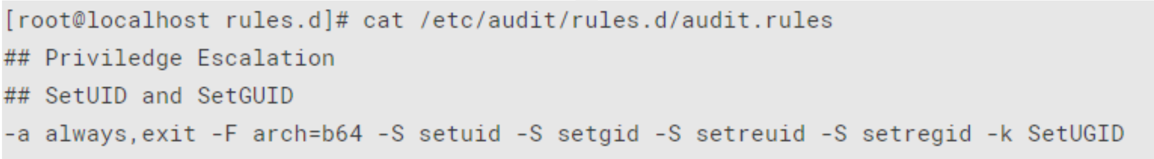
```
-a action,list -S syscall -F field=value -k keyname
```

In the system calls rule structure the `-a` option tells the kernel to append a rule at the end of the rule list. The `-S` options are the syscall name or number, which is vital in notifying the auditor which syscall to monitor. The option `-F` is used to determine what to match against before generating an alert, and finally, the `-k` is the key name to display to the log file when the alert is triggered (Grubb, n.d.).

5.2. Privilege Escalation

5.2.1. Rules created

To catch the privilege escalation script, one must create rules to identify any files that have misconfigured permissions and generate an alert when such a script is executed to gain escalated permissions. The rule below is one way that Auditd can be configured to catch this form of attack:

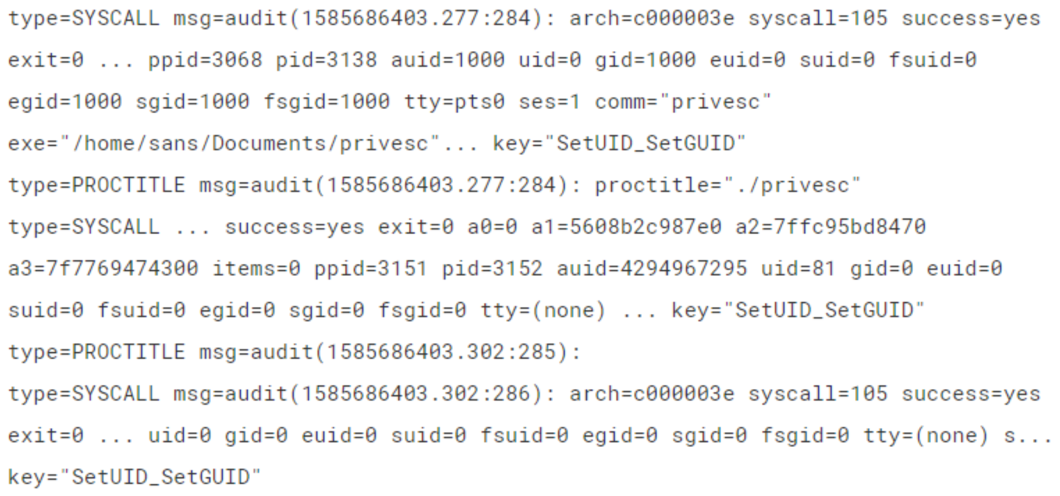


```
[root@localhost rules.d]# cat /etc/audit/rules.d/audit.rules
## Priviledge Escalation
## SetUID and SetGUID
-a always,exit -F arch=b64 -S setuid -S setgid -S setreuid -S setregid -k SetUGID
```

This rule starts by always appending an alert to the logging file upon the exit of a syscall. The `-F` flag specifies the architecture of the file. In this case, it is a `b64` (64-bit system). Next, the `-F` flags specify what syscalls to monitor, and include setuid, setgid, setreuid and setregid. All four of these subroutines reset the file user IDs. Finally, `-k` defines the key as `SetUGID` for identification purposes when parsing the logs in the next section.

5.2.2. Alerts generated

The privilege escalation program was executed to gain a root shell on the system, and the following alert was found in "`/var/log/audit/audit.log`" (The alert has been condensed to save on space):



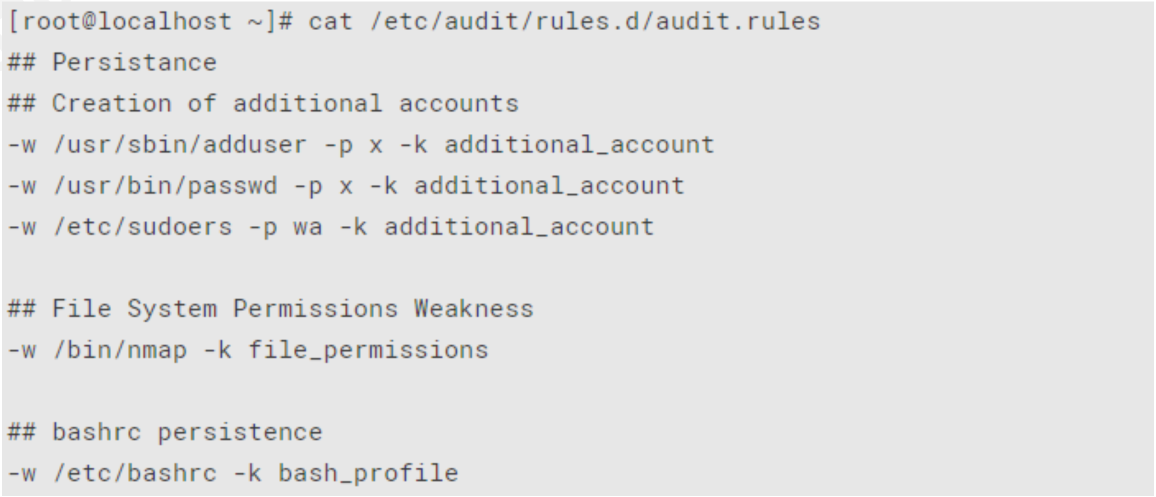
```
type=SYSCALL msg=audit(1585686403.277:284): arch=c000003e syscall=105 success=yes
exit=0 ... ppid=3068 pid=3138 auid=1000 uid=0 gid=1000 euid=0 suid=0 fsuid=0
egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=1 comm="privesc"
exe="/home/sans/Documents/privesc"... key="SetUID_SetGUID"
type=PROCTITLE msg=audit(1585686403.277:284): proctitle="./privesc"
type=SYSCALL ... success=yes exit=0 a0=0 a1=5608b2c987e0 a2=7ffc95bd8470
a3=7f7769474300 items=0 ppid=3151 pid=3152 auid=4294967295 uid=81 gid=0 euid=0
suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ... key="SetUID_SetGUID"
type=PROCTITLE msg=audit(1585686403.302:285):
type=SYSCALL msg=audit(1585686403.302:286): arch=c000003e syscall=105 success=yes
exit=0 ... uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) s...
key="SetUID_SetGUID"
```

As seen in the alert above, Auditd detected that a file called `privesc` created a syscall and went from UID `1000` to UID `0`, which is root. Through this one-line addition to the audit.rules, a system administrator could be alerted any time that a similar privilege escalation was performed on a Linux system.

5.3. Persistence

5.3.1. Rules created

To catch the methods of persistence defined in the `persistence.sh` script found in Appendix b; following rules were created:

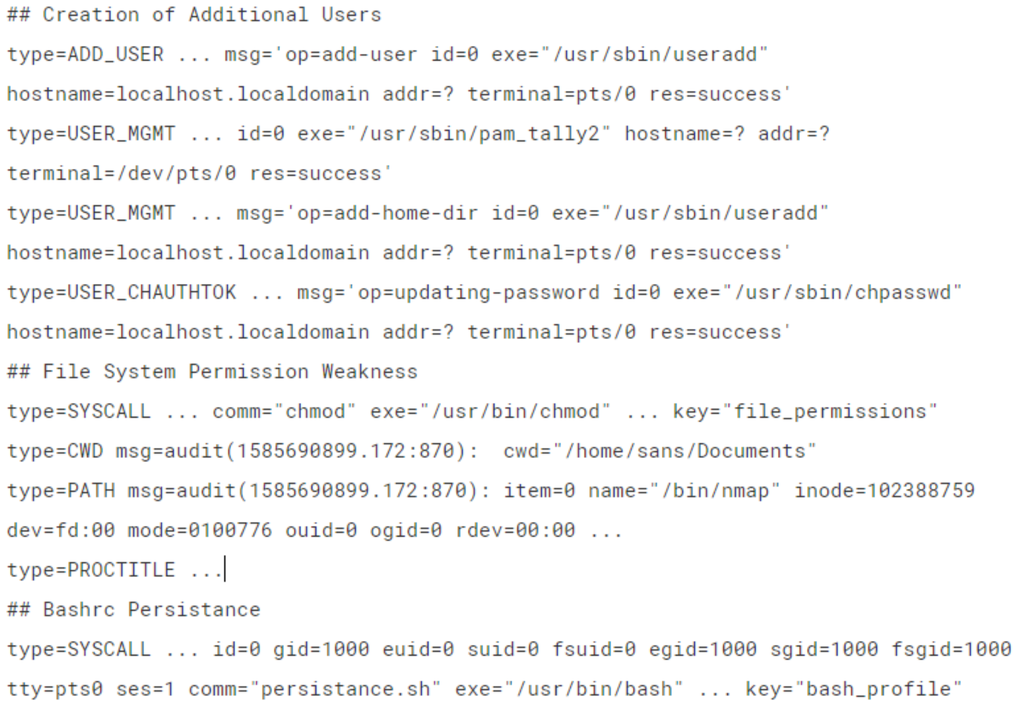


```
[root@localhost ~]# cat /etc/audit/rules.d/audit.rules
## Persistance
## Creation of additional accounts
-w /usr/sbin/adduser -p x -k additional_account
-w /usr/bin/passwd -p x -k additional_account
-w /etc/sudoers -p wa -k additional_account

## File System Permissions Weakness
-w /bin/nmap -k file_permissions

## bashrc persistence
-w /etc/bashrc -k bash_profile
```

The rules above begin by monitoring the execution of the `adduser`, `passwd`, and `sudoers` commands, which are used to create an account, create a password, and add the new account to sudoers. There are other ways to perform these actions, for example, one could modify the /etc/passwd and /etc/shadow files directly; however, those files are modified if the binary of adduser, passwd, or sudoers is executed. The next rule monitors the binary data of /bin/nmap for a change in attribution or permissions. Like privilege escalation demonstrated in the previous module, an adversary could create persistence on a system by messing with the permissions of a file that has an interactive shell. In this instance, if an adversary modified the permissions of Nmap, then they could gain rule access to the system by executing (gtfobins, n.d.): “nmap –interactive." The final rule created is to monitor the /etc/bashrc file for modifications. To enable full monitoring for file system permission weakness and bashrc persistence, the -p flags were not set for /bin/nmap and /etc/bashrc.

5.3.2. Alerts generated

The persistence.sh script was executed to create a new user, change the permissions of the nmap binary, and add a malicious line to /etc/bashrc. The following alerts were generated for the creation of additional accounts:



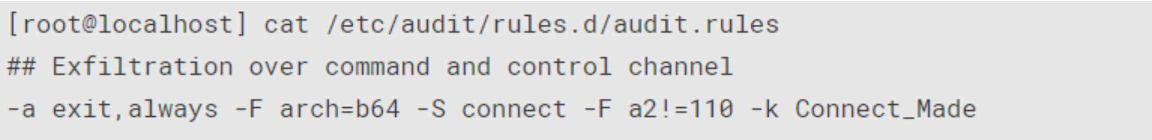
```
## Creation of Additional Users
type=ADD_USER ... msg='op=add-user id=0 exe="/usr/sbin/useradd"
hostname=localhost.localdomain addr=? terminal=pts/0 res=success'
type=USER_MGMT ... id=0 exe="/usr/sbin/pam_tally2" hostname=? addr=?
terminal=/dev/pts/0 res=success'
type=USER_MGMT ... msg='op=add-home-dir id=0 exe="/usr/sbin/useradd"
hostname=localhost.localdomain addr=? terminal=pts/0 res=success'
type=USER_CHAUTHTOK ... msg='op=updating-password id=0 exe="/usr/sbin/chpasswd"
hostname=localhost.localdomain addr=? terminal=pts/0 res=success'
## File System Permission Weakness
type=SYSCALL ... comm="chmod" exe="/usr/bin/chmod" ... key="file_permissions"
type=CWD msg=audit(1585690899.172:870):  cwd="/home/sans/Documents"
type=PATH msg=audit(1585690899.172:870): item=0 name="/bin/nmap" inode=102388759
dev=fd:00 mode=0100776 ouid=0 ogid=0 rdev=00:00 ...
type=PROCTITLE ...
## Bashrc Persistance
type=SYSCALL ... id=0 gid=1000 euid=0 suid=0 fsuid=0 egid=1000 sgid=1000 fsgid=1000
tty=pts0 ses=1 comm="persistance.sh" exe="/usr/bin/bash" ... key="bash_profile"
```

As seen in the `/var/logs/audit/audit.log` above, Auditd alerted that a new user was created, that the user had their password changed, and that the task completed successfully. Then, the log file continued to show that a syscall executed chmod on the file `/bin/nmap` and finally that the script named `persistence.sh` triggered the key `bash_profile`.

5.4. Data Exfiltration

5.4.1. Rules created

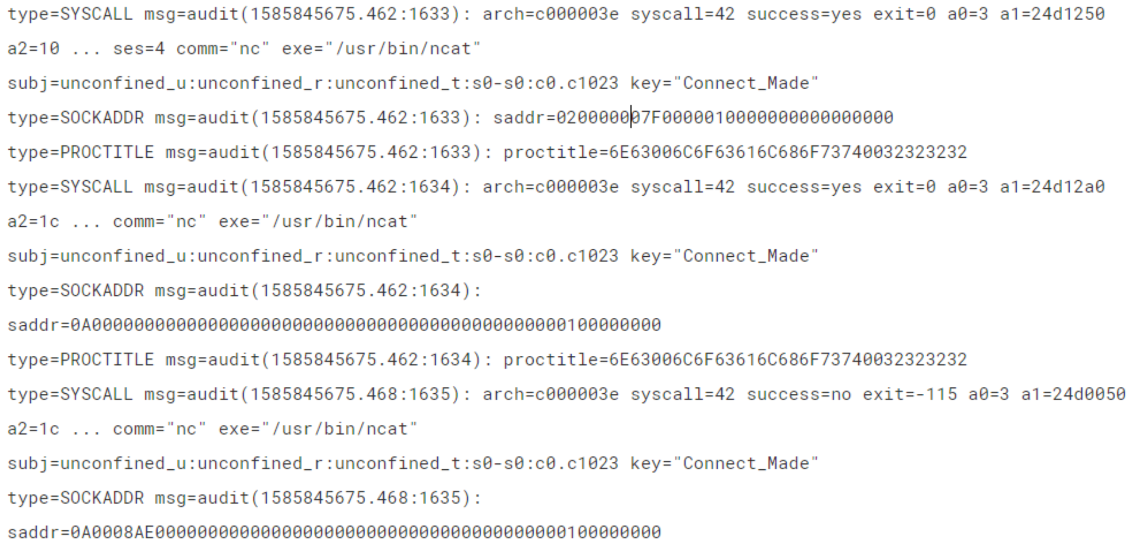
The documentation for writing rules for detecting data exfiltration over alternative protocols in Auditd is suboptimal. The rule below was gathered from a GitHub page, which contains Auditd rules that map to the MITRE ATT&CK framework (bfuzzy, 2018). This rule was developed to catch exfiltration attempts over a command and control channel. However, it is capable of demonstrating the alerting of a connection over an alternative protocol. The first `-F` is used to indicate that the architecture is `x64` (64-bit system). Next, outbound connections could indicate the exfiltration of data; therefore, the syscall `connect` is monitored. This syscall will be useful in capturing TCP connections, which is what the scripts in section 2.3.3 will create. The final part of this rule is the second `-F` flag, where the `a2` stands for the field, and `110` is the value. The definition of this value is not well documented, and additional research is needed to point out the correlation of values. Knowing these two values allows us to create a rule that is looking for a TCP connection that did not time out.



```
[root@localhost] cat /etc/audit/rules.d/audit.rules
## Exfiltration over command and control channel
-a exit,always -F arch=b64 -S connect -F a2!=110 -k Connect_Made
```

5.4.2. Alerts generated

After executing the data exfiltration script, the following alert was generated by Auditd (Note: Note: Appendix F contains the full output):



```
type=SYSCALL msg=audit(1585845675.462:1633): arch=c000003e syscall=42 success=yes exit=0 a0=3 a1=24d1250
a2=10 ... ses=4 comm="nc" exe="/usr/bin/ncat"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="Connect_Made"
type=SOCKADDR msg=audit(1585845675.462:1633): saddr=020000007F0000010000000000000000
type=PROCTITLE msg=audit(1585845675.462:1633): proctitle=6E63006C6F63616C686F73740032323232
type=SYSCALL msg=audit(1585845675.462:1634): arch=c000003e syscall=42 success=yes exit=0 a0=3 a1=24d12a0
a2=1c ... comm="nc" exe="/usr/bin/ncat"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="Connect_Made"
type=SOCKADDR msg=audit(1585845675.462:1634):
saddr=0A00000000000000000000000000000000000000000000000100000000
type=PROCTITLE msg=audit(1585845675.462:1634): proctitle=6E63006C6F63616C686F73740032323232
type=SYSCALL msg=audit(1585845675.468:1635): arch=c000003e syscall=42 success=no exit=-115 a0=3 a1=24d0050
a2=1c ... comm="nc" exe="/usr/bin/ncat"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="Connect_Made"
type=SOCKADDR msg=audit(1585845675.468:1635):
saddr=0A0008AE000000000000000000000000000000000000000100000000
```

In the alert above, Auditd alerted that a new connection was made through the use of nc or netcat binary. As can be seen in this alert, the `a2` value for these syscalls is not equal to `110`, which is what the created rule checks. This alert shows that a connection to exfiltrate data was made; however, it does not show what data has been exfiltrated.

6. Recommendations and Implications

This research examined the efficacy of three separately installed HIDS; OSSEC, Samhain, and Auditd, at detecting various types of tactics and techniques defined by the MITRE ATT&CK framework. The tactics that were tested are privilege escalation, persistence, and data exfiltration. The tools were installed and configured on three separate standardized CentOS 7 operating systems, and their efficiency was tested through the detection of adversarial activities generated by scripts, which can be found in the Appendix. Figure 1 summarizes, which HIDS detected which tactic and technique from the ATT&CK framework and are explained further below.

MITRE ATT&CK Matrix		**Host-based Intrusion Detection Systems**		
Tactics	**Technique**	**OSSEC**	**Samhain**	**Auditd**
Privilege Escalation	Setuid and Setguid	Alerted presence of misconfigured file.	Alerted presence of misconfigured file.	Alerted execution of misconfigured file.
Persistence	Create Account	Alerted modification of /etc/passwd and /etc/shadow. Presented exact changes.	Alerted modification of /etc/passwd and /etc/shadow.	Alerted when binary adduser and passwd was executed.
	Setuid and Setguid	Alerted on modification of attributes of a file against known good state of filesystem.	Alerted on modification of attributes of a file against known good state of filesystem.	Alerted when attributes of monitored file were modified.
	.bashrc persistance	Alerted when /etc/bashrc was modified. Presented exact changes.	Alerted when /etc/bashrc was modified.	Alerted when /etc/bashrc was modified.
Data Exfiltration	Exfiltration over alternative protocol	Alerted when new port was opened on system. Failed to detect file being exfiltrated.	Alerted when new port was opened on system. Failed to detect file being exfiltrated.	Alerted when new connection was established. Failed to detect file being exfiltrated.

Figure 1 Summary of Tactic detection by HIDS

6.1. Privilege Escalation

The privilege escalation script executed on the CentOS 7 hosts relied on the technique of SUID and SGID misconfiguration, which allows a file that is owned by a regular user to be performed with root privileges. This can lead to an escalation of privileges from an unprivileged user to root. As can be seen in Figure 1, each of the tools generated an alert upon discovering a misconfigured file, but the alerts showed different levels of information across platforms. OSSEC and Samhain were able to find the malicious file and alert that it exists but were not able to identify that it ran. Auditd, which looks for the exact syscall, was able to inform when the file was executed but was not able to show the misconfigured permissions. It is concluded that for these tools to work at their full potential to detect this specific technique, then Auditd should be configured on the system, and OSSEC and Samhain should aggregate the logs generated by Auditd and alert on them.

6.2. Persistence

An adversary can use persistence to create ways to regain access to a compromised host post-exploitation. The persistence script executed in this paper creates an additional account, modifies the permissions of binary, and adds persistence to the /etc/bashrc. Each of the HIDS used in this paper were able to discover this script in the configured manner. Both OSSEC and Samhain were able to detect that the /etc/passwd and /etc/shadow file were modified from the original, that the permissions of /bin/nmap were altered and that /etc/bashrc was changed. Additionally, as seen in Figure 1, Auditd caught when the command adduser and passwd were executed, when /bin/nmap had its permissions modified, and /etc/bashrc was changed. All three of the tools were able to alert on the execution of this persistence script successfully. OSSEC was able to provide the most verbose information about the changes to the files.

6.3. Data Exfiltration

The tactic of data exfiltration is used by an adversary to take valuable files and information from a compromised system back to the attack system. The script used to exfiltrate data by transferring a file between predefined ports, which represents the tactic of exfiltration over an alternative protocol. Both OSSEC and Samhain used the known good state of the system to determine when a new port was created, but these tools would not have caught the attack if that connection existed when the database was initialized. Auditd could only alert that a socket connect has been established. Neither of the tools was able to detect the file that was fed over the network. It is difficult for a HIDS to determine what is an exfiltration attempt and what is not. Network intrusion detection systems should be used to monitor network traffic, generate a known good state, and spot data exfiltration anomalies (Rapid7, 2017).

7. Conclusion

With the increased adversarial activity on Unix based operating systems, security professionals need to be educated on which open-source HIDS most effectively detect potential vectors of compromise. This research compared the efficacy of OSSEC, Samhain, and Auditd at identifying various types of adversarial tactics and techniques gathered from the MITRE ATT&CK framework. It has been concluded that the HIDS chosen for this paper have their unique benefits. For example, the installation of OSSEC was straight forward and allowed for the basic logging of incidents, and easy enabling of features upon limited configuration. On the contrary, Samhain required an in-depth configuration to generate similar alerts as OSSEC. Finally, Auditd comes preinstalled on most Linux operating systems, and its granular creation of rules enables alerting of UNIX system calls. Each of these HIDS can detect privilege escalation, persistence, and data exfiltration in their way. However, their full potential at detecting adversarial activities is achieved when they are configured to work together.

References

Avast. (2020). *What is an Intrusion Detection System (IDS)?* Retrieved from Avast: https://smb.avast.com/answers/intrusion-detection-system-ids

bfuzzy. (2018, November 26). *auditd-attack*. Retrieved from Github: https://github.com/bfuzzy/auditd-attack/blob/master/auditd-attack.rules

gabrielfalcao. (2013). Retrieved from Github: https://gist.github.com/gabrielfalcao/4216897

Giovanni Vigna, C. K. (2005, June 15). *Host-Based Intrusion Detection .* Retrieved from University of California, Santa Barbara: https://sites.cs.ucsb.edu/~chris/research/doc/infsec05_hids.pdf

Grubb, S. (2020). *auditd - Linux Man Page*. Retrieved from die.net: https://linux.die.net/man/8/auditd

Grubb, S. (n.d.). *AUDIT.RULES*. Retrieved from jangnan.org: https://linux.jangnan.org/usr/share/man/man7/audit.rules.7.gz.html

Grubb, S. (n.d.). *audit.rules(7) - Linux man page*. Retrieved from die.net: https://linux.die.net/man/7/audit.rules

gtfobins. (n.d.). *nmap*. Retrieved from gtfobins.github.io: https://gtfobins.github.io/gtfobins/nmap/

Lackmann, G. (2010). *An introduction to UNIX/Linux.* Retrieved from North Carolina State University: https://projects.ncsu.edu/atmos_collaboration/gary/mea443/ppt/443_F10_linux.pdf

MITRE Corporation. (2019, July 19). *Exfiltration*. Retrieved from MITRE ATT&CK: https://attack.mitre.org/tactics/TA0010/

Mourer, J. (2014, September 12). *Privilege escalation with setuid*. Retrieved from Getkey: https://getkey.eu/blog/541346dd/privilege-escalation-with-setuid

Nel, M. (2014, April 24). *sans.org.* Retrieved from SAMHAIN: Host Based Intrusion Detection via File: https://www.sans.org/reading-room/whitepapers/detection/samhain-host-based-intrusion-detection-file-integrity-monitoring-34567

OSSEC. (2020). *Syscheck*. Retrieved from ossec.net: https://www.ossec.net/docs/manual/syscheck/index.html

OSSEC. (2020). *The World's Most Widely Used Host-based Intrusion Detection System*. Retrieved from OSSEC: https://www.ossec.net/

OSSEC Project. (2020). *Rootcheck.* Retrieved from ossec: https://www.ossec.net/docs/manual/rootcheck/manual-rootcheck.html

OSSEC Project Team. (n.d.). *OSSEC Documentation*. Retrieved from OSSEC: https://www.ossec.net/docs/

Rapid7. (2017, Jan 11). *The Pros & Cons of Intrusion Detection Systems*. Retrieved from Rapid7: https://blog.rapid7.com/2017/01/11/the-pros-cons-of-intrusion-detection-systems/

Redhat. (n.d.). *Audit Record Types*. Retrieved from redhat.com: https://access.redhat.com/articles/4409591#audit-record-types-2

SANS Institute. (2005). Retrieved from cyber-defense.sans.org: https://cyber-defense.sans.org/resources/papers/gsec/host-vs-network-based-intrusion-detection-systems-102574

Sarmah, A. (2001). *Intrusion Detection Systems: Definition, Need and Challenges.* Retrieved from Sans: https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-systems-definition-challenges-343

The MITRE Corporation. (2020). Retrieved from MITRE ATT&CK: https://attack.mitre.org/

The MITRE Corporation. (2020). *Frequently Asked Questions*. Retrieved from MITRE ATT&CK: https://attack.mitre.org/resources/faq/

The MITRE Corporation. (n.d.). *Privilege Escalation*. Retrieved from mitre.org: https://attack.mitre.org/tactics/TA0004/

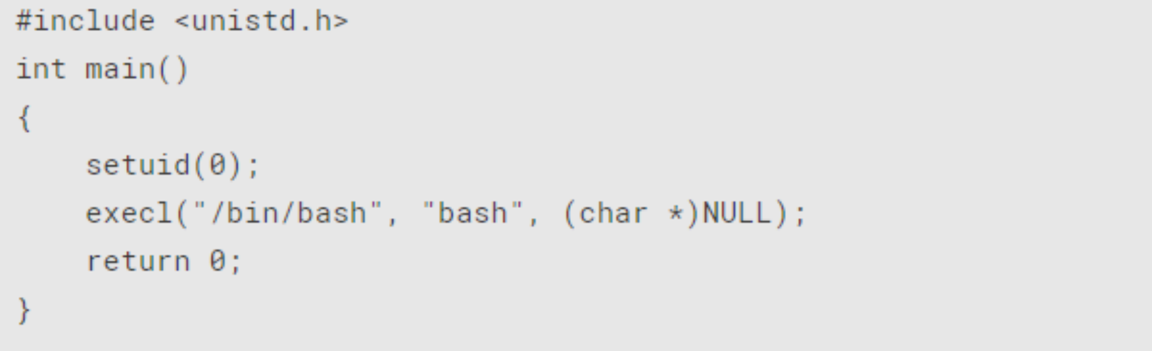
Tripwire. (2019, March 16). *Open Source Tripwire*. Retrieved from Github: https://github.com/Tripwire/tripwire-open-source

WatchGuard. (2017). *Internet Security Report - Q1 2017*. Retrieved from WatchGuard: https://www.watchguard.com/wgrd-resource-center/security-report-q1-2017

Wichmann, R. (2006). *Initialize the baseline database*. Retrieved from samhna: https://www.la-samhna.de/samhain/manual/installation-initialize.html

Wichmann, R. (2006). *The SAMHAIN file integrity / host-based intrusion detection system*. Retrieved from Samhain labs: https://la-samhna.de/samhain/

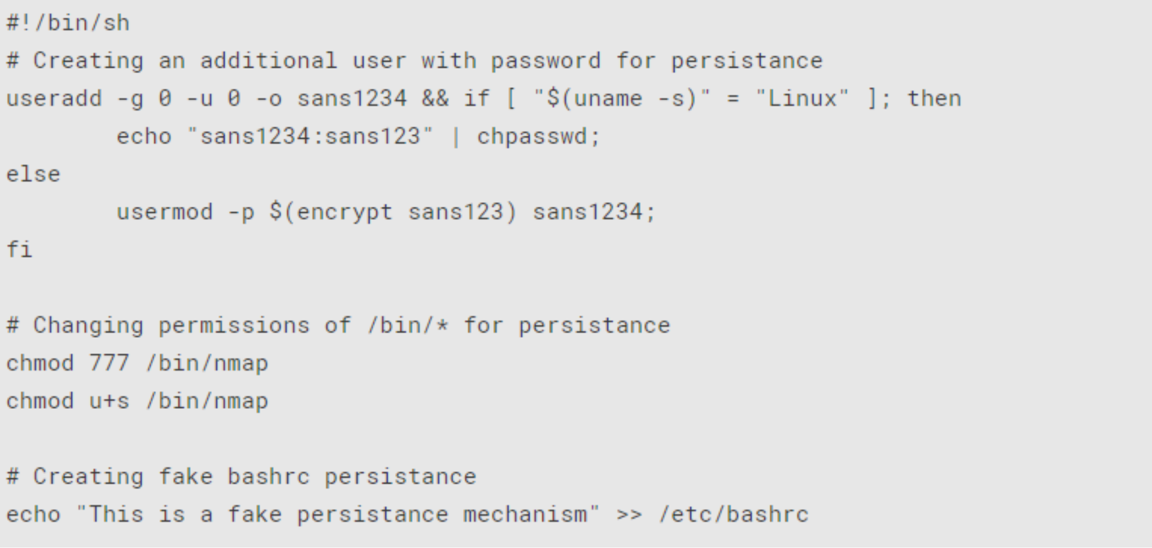
8. Appendix A – Privilege Escalation Program



```
#include <unistd.h>
int main()
{
    setuid(0);
    execl("/bin/bash", "bash", (char *)NULL);
    return 0;
}
```

To execute this program to test your HIDS, one must first compile this code with "`gcc filename.c -o filename`" and misconfigure its permissions "`chown root filename`" and "`chmod u+s filename`." Afterward, this file can be executed from an unprivileged user to gain root access.

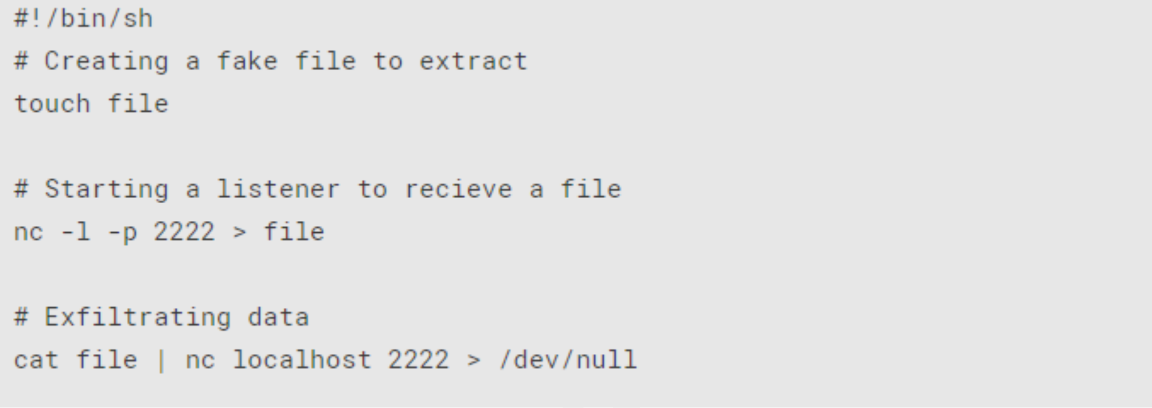
9. Appendix B – Persistence Script



```
#!/bin/sh
# Creating an additional user with password for persistance
useradd -g 0 -u 0 -o sans1234 && if [ "$(uname -s)" = "Linux" ]; then
        echo "sans1234:sans123" | chpasswd;
else
        usermod -p $(encrypt sans123) sans1234;
fi

# Changing permissions of /bin/* for persistance
chmod 777 /bin/nmap
chmod u+s /bin/nmap

# Creating fake bashrc persistance
echo "This is a fake persistance mechanism" >> /etc/bashrc
```

Note: This script must be made executable (`chmod u+x filename`) before it can be run.

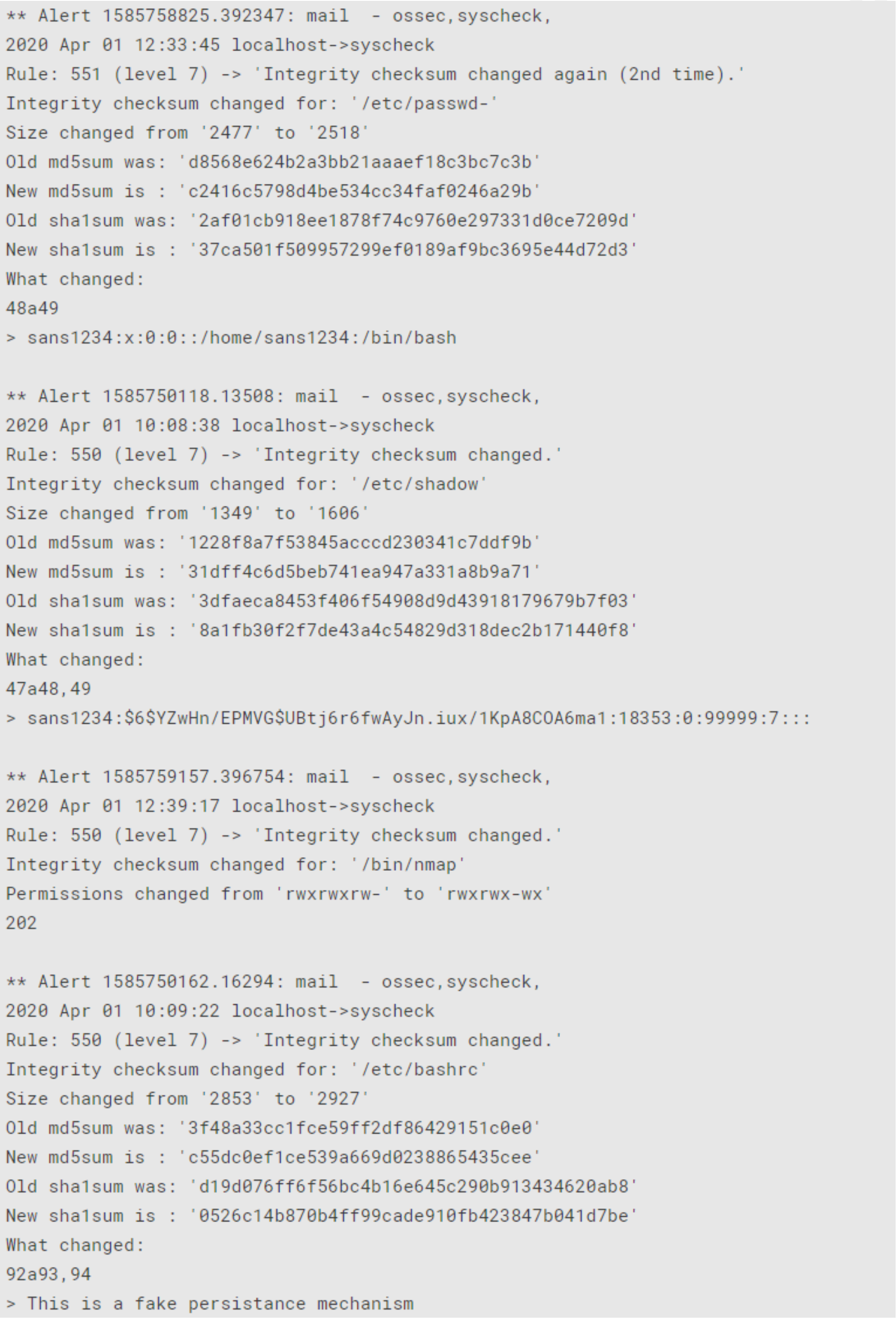
10. Appendix C – Data Exfiltration Script



```
#!/bin/sh
# Creating a fake file to extract
touch file

# Starting a listener to recieve a file
nc -l -p 2222 > file

# Exfiltrating data
cat file | nc localhost 2222 > /dev/null
```

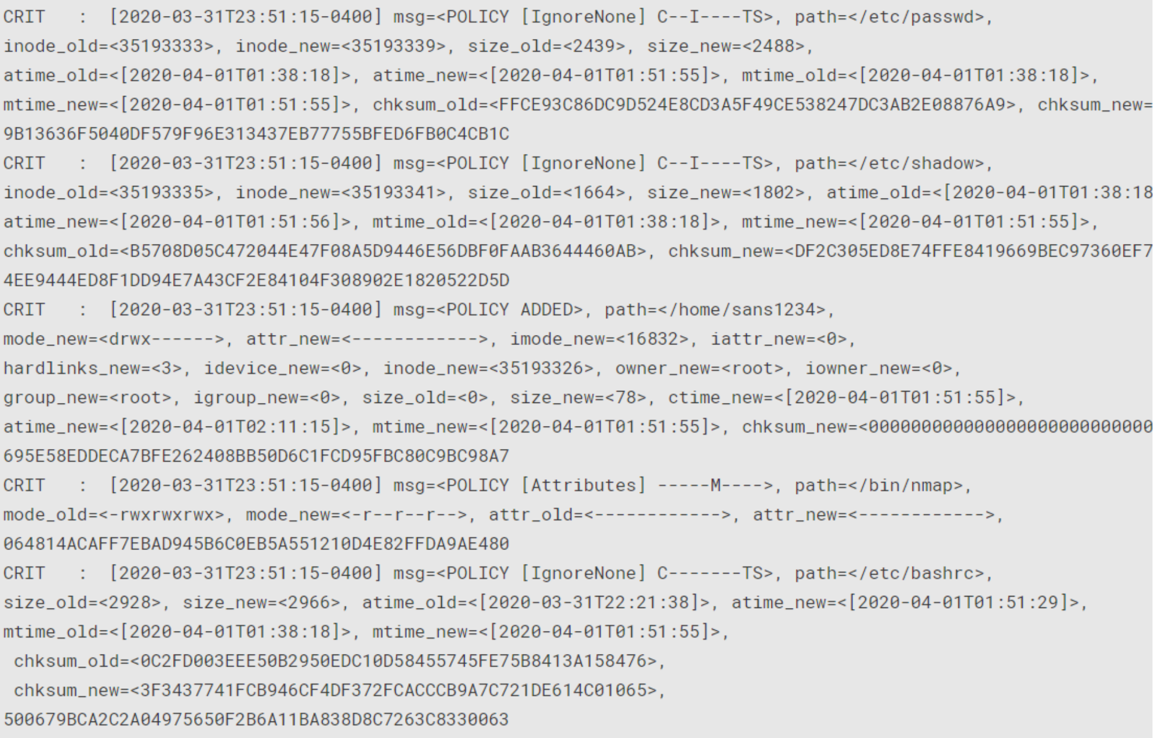
Note: This script must be made executable (`chmod u+x filename`) before it can be run.

11. Appendix D – OSSEC Persistence Alert



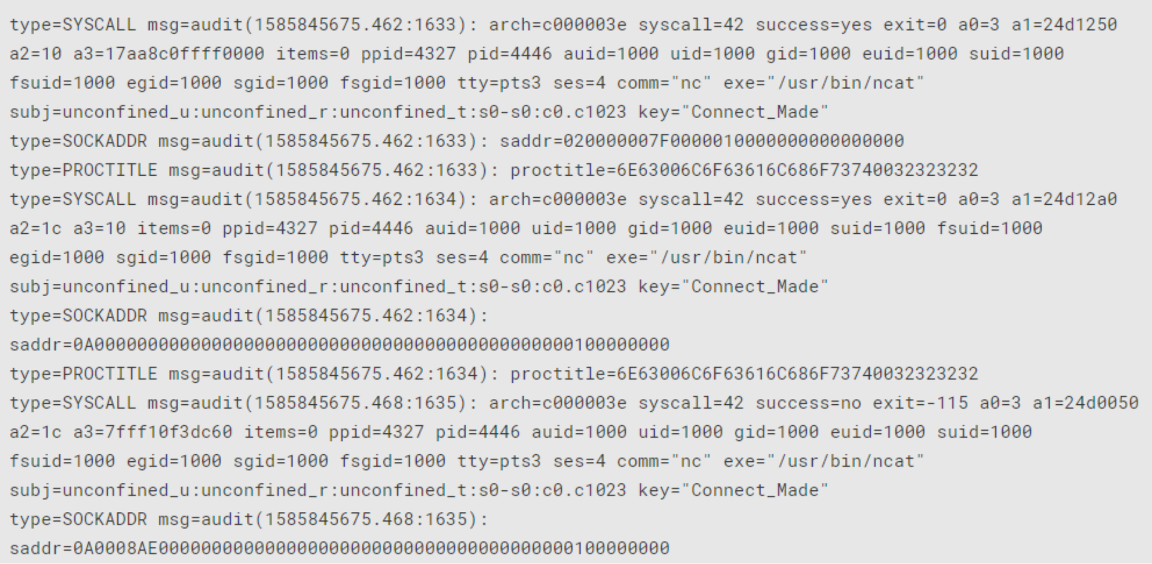
```
** Alert 1585758825.392347: mail  - ossec,syscheck,
2020 Apr 01 12:33:45 localhost->syscheck
Rule: 551 (level 7) -> 'Integrity checksum changed again (2nd time).'
Integrity checksum changed for: '/etc/passwd-'
Size changed from '2477' to '2518'
Old md5sum was: 'd8568e624b2a3bb21aaaef18c3bc7c3b'
New md5sum is : 'c2416c5798d4be534cc34faf0246a29b'
Old sha1sum was: '2af01cb918ee1878f74c9760e297331d0ce7209d'
New sha1sum is : '37ca501f509957299ef0189af9bc3695e44d72d3'
What changed:
48a49
> sans1234:x:0:0::/home/sans1234:/bin/bash

** Alert 1585750118.13508: mail  - ossec,syscheck,
2020 Apr 01 10:08:38 localhost->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/etc/shadow'
Size changed from '1349' to '1606'
Old md5sum was: '1228f8a7f53845acccd230341c7ddf9b'
New md5sum is : '31dff4c6d5beb741ea947a331a8b9a71'
Old sha1sum was: '3dfaeca8453f406f54908d9d43918179679b7f03'
New sha1sum is : '8a1fb30f2f7de43a4c54829d318dec2b171440f8'
What changed:
47a48,49
> sans1234:$6$YZwHn/EPMVG$UBtj6r6fwAyJn.iux/1KpA8COA6ma1:18353:0:99999:7:::

** Alert 1585759157.396754: mail  - ossec,syscheck,
2020 Apr 01 12:39:17 localhost->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/bin/nmap'
Permissions changed from 'rwxrwxrw-' to 'rwxrwx-wx'
202

** Alert 1585750162.16294: mail  - ossec,syscheck,
2020 Apr 01 10:09:22 localhost->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/etc/bashrc'
Size changed from '2853' to '2927'
Old md5sum was: '3f48a33cc1fce59ff2df86429151c0e0'
New md5sum is : 'c55dc0ef1ce539a669d0238865435cee'
Old sha1sum was: 'd19d076ff6f56bc4b16e645c290b913434620ab8'
New sha1sum is : '0526c14b870b4ff99cade910fb423847b041d7be'
What changed:
92a93,94
> This is a fake persistance mechanism
```

12. Appendix E – Samhain Persistence Alert



```
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY [IgnoreNone] C--I----TS>, path=</etc/passwd>,
inode_old=<35193333>, inode_new=<35193339>, size_old=<2439>, size_new=<2488>,
atime_old=<[2020-04-01T01:38:18]>, atime_new=<[2020-04-01T01:51:55]>, mtime_old=<[2020-04-01T01:38:18]>,
mtime_new=<[2020-04-01T01:51:55]>, chksum_old=<FFCE93C86DC9D524E8CD3A5F49CE538247DC3AB2E08876A9>, chksum_new=
9B13636F5040DF579F96E313437EB77755BFED6FB0C4CB1C
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY [IgnoreNone] C--I----TS>, path=</etc/shadow>,
inode_old=<35193335>, inode_new=<35193341>, size_old=<1664>, size_new=<1802>, atime_old=<[2020-04-01T01:38:18
atime_new=<[2020-04-01T01:51:56]>, mtime_old=<[2020-04-01T01:38:18]>, mtime_new=<[2020-04-01T01:51:55]>,
chksum_old=<B5708D05C472044E47F08A5D9446E56DBF0FAAB3644460AB>, chksum_new=<DF2C305ED8E74FFE8419669BEC97360EF7
4EE9444ED8F1DD94E7A43CF2E84104F308902E1820522D5D
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY ADDED>, path=</home/sans1234>,
mode_new=<drwx------>, attr_new=<------------>, imode_new=<16832>, iattr_new=<0>,
hardlinks_new=<3>, idevice_new=<0>, inode_new=<35193326>, owner_new=<root>, iowner_new=<0>,
group_new=<root>, igroup_new=<0>, size_old=<0>, size_new=<78>, ctime_new=<[2020-04-01T01:51:55]>,
atime_new=<[2020-04-01T02:11:15]>, mtime_new=<[2020-04-01T01:51:55]>, chksum_new=<000000000000000000000000000
695E58EDDECA7BFE262408BB50D6C1FCD95FBC80C9BC98A7
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY [Attributes] -----M---->, path=</bin/nmap>,
mode_old=<-rwxrwxrwx>, mode_new=<-r--r--r-->, attr_old=<------------>, attr_new=<------------>,
064814ACAFF7EBAD945B6C0EB5A551210D4E82FFDA9AE480
CRIT   :  [2020-03-31T23:51:15-0400] msg=<POLICY [IgnoreNone] C-------TS>, path=</etc/bashrc>,
size_old=<2928>, size_new=<2966>, atime_old=<[2020-03-31T22:21:38]>, atime_new=<[2020-04-01T01:51:29]>,
mtime_old=<[2020-04-01T01:38:18]>, mtime_new=<[2020-04-01T01:51:55]>,
 chksum_old=<0C2FD003EEE50B2950EDC10D58455745FE75B8413A158476>,
 chksum_new=<3F3437741FCB946CF4DF372FCACCCB9A7C721DE614C01065>,
500679BCA2C2A04975650F2B6A11BA838D8C7263C8330063
```

13. Appendix F – Auditd Data Exfiltration Alert



```
type=SYSCALL msg=audit(1585845675.462:1633): arch=c000003e syscall=42 success=yes exit=0 a0=3 a1=24d1250
a2=10 a3=17aa8c0ffff0000 items=0 ppid=4327 pid=4446 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000
fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts3 ses=4 comm="nc" exe="/usr/bin/ncat"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="Connect_Made"
type=SOCKADDR msg=audit(1585845675.462:1633): saddr=020000007F0000010000000000000000
type=PROCTITLE msg=audit(1585845675.462:1633): proctitle=6E63006C6F63616C686F73740032323232
type=SYSCALL msg=audit(1585845675.462:1634): arch=c000003e syscall=42 success=yes exit=0 a0=3 a1=24d12a0
a2=1c a3=10 items=0 ppid=4327 pid=4446 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000
egid=1000 sgid=1000 fsgid=1000 tty=pts3 ses=4 comm="nc" exe="/usr/bin/ncat"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="Connect_Made"
type=SOCKADDR msg=audit(1585845675.462:1634):
saddr=0A00000000000000000000000000000000000000000000000100000000
type=PROCTITLE msg=audit(1585845675.462:1634): proctitle=6E63006C6F63616C686F73740032323232
type=SYSCALL msg=audit(1585845675.468:1635): arch=c000003e syscall=42 success=no exit=-115 a0=3 a1=24d0050
a2=1c a3=7fff10f3dc60 items=0 ppid=4327 pid=4446 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000
fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts3 ses=4 comm="nc" exe="/usr/bin/ncat"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="Connect_Made"
type=SOCKADDR msg=audit(1585845675.468:1635):
saddr=0A0008AE000000000000000000000000000000000000000100000000
```