**Course Topics**

[**Security Monitoring Fundamentals**](#_bookmark0)

1. [Getting Start](#_bookmark1)
   1. [SOC Components](#_bookmark2)
   2. [Security Monitoring Process](#_bookmark3)
   3. [Identification and Analysis](#_bookmark4)
   4. [Required Knowledge](#_bookmark5)
   5. [Security Monitoring Tasks](#_bookmark6)
2. [Security Monitoring Infrastructure](#_bookmark8)
   1. [Lab Architecture](#_bookmark9)
   2. [Windows Setup and Logging](#_bookmark10)
   3. [Linux Setup and Logging](#_bookmark11)
   4. [NSM Setup and Logging](#_bookmark12)
   5. [SIEM Setup](#_bookmark13)
3. Splunk as a Security Monitoring Solution
   1. Introduction to Splunk
   2. Splunk Architecture
   3. Log Parsing and Normalization
   4. Splunk SPL
   5. Data Models and Correlation Searches
   6. Alerts, Dashboards and Reports Endpoint Security Monitoring
4. Windows Components
   1. Windows Architecture
   2. Windows Event Log
   3. Windows Process
   4. COM Objects and .NET Framework
   5. [Windows Powershell](#_bookmark7)
   6. Windows Registry
   7. Windows APIs
   8. Windows Defender
5. Windows Auditing and Logging
   1. Sysmon Installation and Configuration
   2. Windows Audit Policies
   3. Windows Firewall Logging
   4. Powershell Logging
   5. Windows Defender Logging
6. Windows Threats Analysis
   1. Credential Abuse
   2. Binary Attacks
   3. Microsoft Office Attacks
   4. Windows Privilege Escalation
   5. Windows Persistence
   6. Windows Lateral Movement
   7. Antivirus Alerts and Evasion
   8. Active Directory Attacks
7. Linux Components
   1. Linux Architecture
   2. Linux Bash and Scripting
   3. Linux Auditd Service
   4. Linux Device and Drivers
   5. Linux Syscalls
   6. Linux Firewall and SELinux
8. [Linux Auditing and Logging](#_bookmark14)
   1. Auditd and Rules
   2. Kernel Auditing
   3. Files and Directory Auditing
   4. Firewall Auditing
   5. Suspicious Activity Auditing
9. Linux Threat Analysis
   1. Credential Abuse
   2. Linux Privilege Escalation
   3. Linux Persistence
   4. Linux Defense Evasion Network Security Monitoring
10. Network Service and Components
    1. Network Protocol Stack
    2. Mail Service
    3. DNS
    4. Web
    5. SMB
    6. RPC
    7. Netflow
    8. Syslog
11. NSM Components
    1. Suricata as a NIDS
    2. Firewall
    3. Flow Generator
    4. Full Packet Capture
12. NSM Auditing and Logging
    1. NIDS Configuration
    2. NIDS Rules Writing
    3. NIDS Alerting
    4. Firewall Rules and Logging
    5. Flow Logging
13. Network Threat Analysis
    1. Web Attacks
    2. SMTP Attacks
    3. SMB Attacks
    4. RPC Attacks
    5. DNS Attacks
    6. C2 Infrastructure
    7. Port Forwarding
    8. Tunneling
    9. Known Exploits Security Monitoring Functions
14. Security Monitoring Checklist
15. Alert Investigation
16. Alert Correlation
17. Monitoring Tips and Tricks
18. Threat Intelligence
19. Incident Reporting
20. Security Monitoring Automation

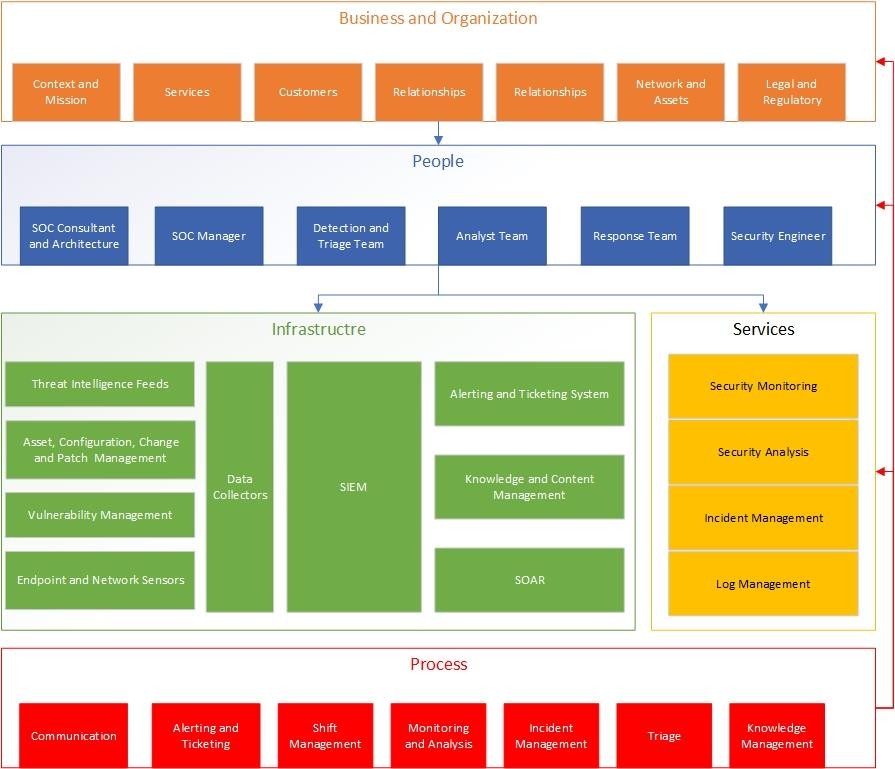
# Security Monitoring Fundamentals

## Getting Start 1.1.SOC Components

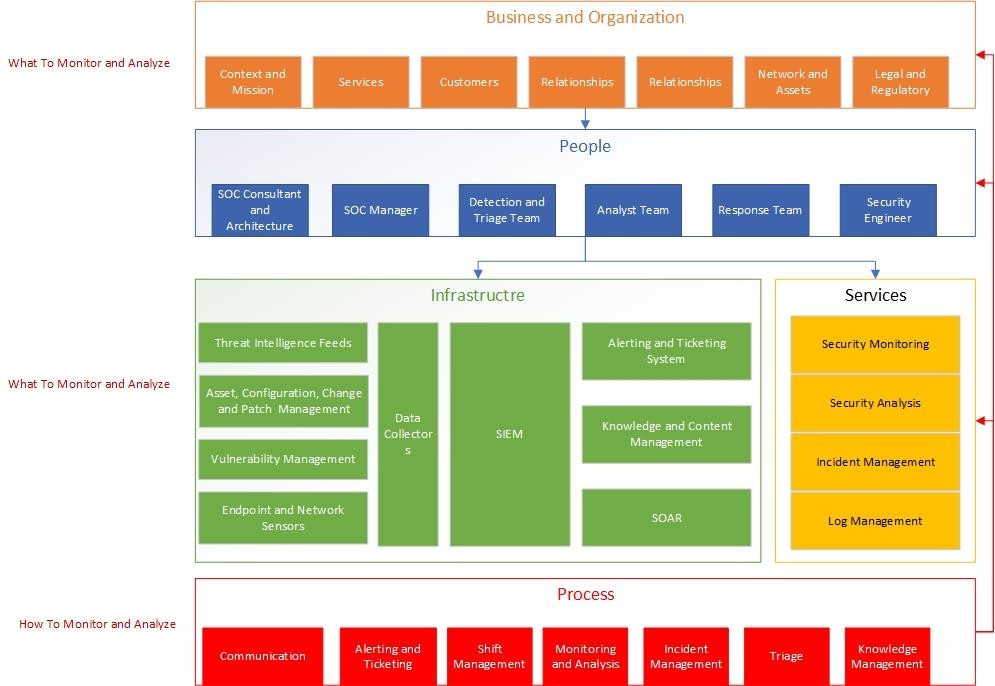
Two Important Questions Security Monitoring Team MUST Ask Yourself:

* + What to Monitor and Analyze
  + How to Monitor and Analyze

Before answer, Let’s review SOC Components:

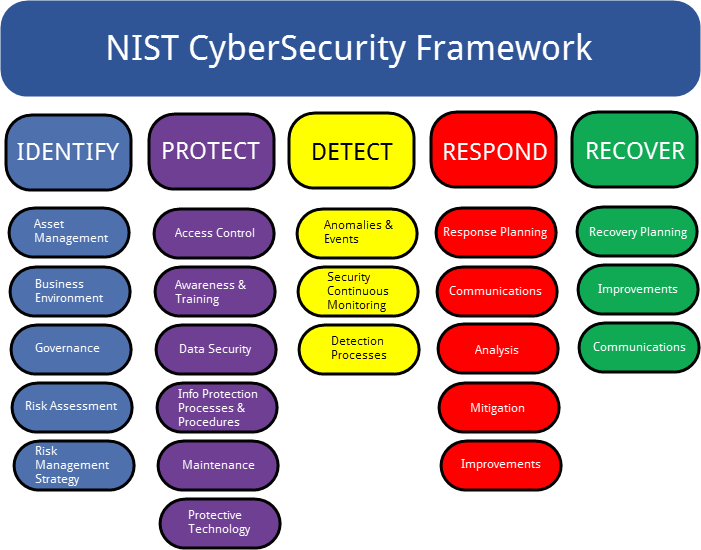


Based on the components:



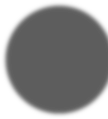
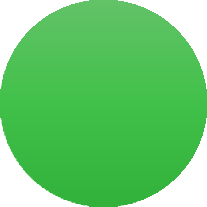
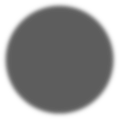
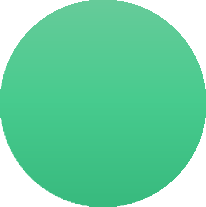
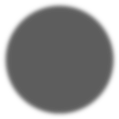
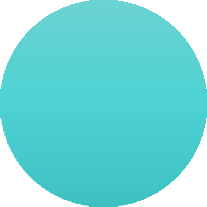
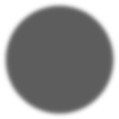
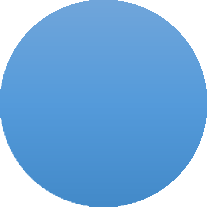
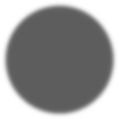
## Security Monitoring Process

### CSF Framework



*Security Monitoring Process*

## Identification and Analysis



Business

Environment

Asset

Management

Security Policy

Network

Topology

What to

Monitor

### Business Environment

* + - Business Context: Mission, Objectives and …
    - Legal and Regulatory Requirements
    - Business Relationships
    - Governance Structure

### Asset Management

* + - Integrate Monitoring Solutions with Asset Management Tools (ServiceDesk, Active Directory)
    - Create/Fill a list of Assets and Attributes Asset Name, Type, Functionality, Trusted Users, …

### Security policy

* + - Acceptable Use Policy
    - Access Control Policy
    - Configuration Management Policy
    - Asset Update Policy
    - Password Policy
    - Data Backup Policy
    - Data in transition Policy
    - , ….

### Network Communication Topology

* + - Communication between Zones
    - Communication Devices
    - , …

### Case Study:

Refer to Excel File

*Analyzing Results*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key Requirements and Results Drived from Identification Phase | | | | | |
| Trigge  r | High Level Concerns | Technical Mapping  to Monitoring | Sensor  s | Datatype | State |
| Identify Business | Monitor and Anlyze Untrusted Connections to/from External Networks | Connection from untrusted relationships Connection to/from Automation during work-off Times Connection to/from Untrusted Locations  … | NSM  Firewal l  ACL | Network Connectio n Logs ACL Logs | NSM  Logs Exist Firewall Logs Exists Lack of ACL  Logs |
| Monitor and Anlyze Unsecure  Network Connections |  |  |  |  |
| Monitor and Anlyze Unathorized  Access to Saving Accounts |  |  |  |  |
| Monitor and Anlyze Unathorized  Access to Transaction Database |  |  |  |  |
| Compliance Monitoring Based-on  BSA Security Controls |  |  |  |  |
| Identify Assets | Monitor and Analyze None  Updated Assets |  |  |  |  |
| Monitor and Analyze AV Status on  Assets |  |  |  |  |
| Monitor and Analyze Unauthorized Internet Access from Clients and  Others |  |  |  |  |
| Monitor and Analyze Clients to  Clients Communications |  |  |  |  |
| Monitor and Analyze WSUS Access  and Network Connections |  |  |  |  |
| Security Policies | Monitor and Analyze Access  Control Violations |  |  |  |  |
| Monitor and Analyze Configuration  Managements Violations |  |  |  |  |
| Monitor and Analyze Media  Protection Violations |  |  |  |  |
| Monitor and Analyze  Authentication Policy Violations |  |  |  |  |
| Netwok Architecture Diagram | Monitor and Analyze File Server To Database Connections |  | DBF | Database Connectio  n Logs | Lack of DBF |
| Monitor and Analyze  Communication Between Servers |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Monitor and Analyze Communication Between Data  Center and Client Zones |  |  |  |  |

## Required Knowledge

### Review Jobs (Indeed)

|  |  |  |
| --- | --- | --- |
| Company | Responsibilities | Required Knowledge |
| Englewood | * Conduct proactive monitoring, investigations, and mitigation of security events * Analyze security event data from EDR, SIEM, Dashboards, etc. * Spend time understanding the environment you're responsible for and engage with various teams to gain further knowledge of the environment(s) * Recognize potential, successful, and unsuccessful intrusion attempts and compromises through review and analysis of relevant event data * Research new and evolving threats with potential to impact the monitored environment | * Minimum 2 years experience in Information Systems or IT security-related functions * Knowledge of information security principles, concepts, practices * Knowledge of networks, firewalls, and operating systems * Ability to provide technical advice, guidance, and recommendations to management and other technical specialists on critical information technology security issues * Strong analytical skills and able to collate and interpret data from various sources * Experience with security incident detection and response |
| Cognizant Technology Solutions New York | * Monitors various log sources from tools and applications such as Endpoint Detection and Response (EDR) logs, Intrusion   Prevention/Detection Systems (IPS/IDS), firewall logs, Windows logs, Linux operating system logs, etc.   * Analyze, investigate, and respond to security events and incidents. * Escalate high or critical incidents or complex security alerts to Senior Security Analysts. * Track and update security incidents over the course of the incident lifecycle. * Work with SIEM engineering to fine- tune rules for false positive alerts. * Develop and suggest SIEM rules that help in detection of security incidents. * Prepare documents and reports as requested. | Minimum Qualifications:   * 0 or more years of Security Operations Center experience * Some IT exposure (Networking, Service Desk, self-learning, etc.) * Industry standard security certification (i.e., Security+ or other entry-level security certifications) * Strong verbal/written communication and interpersonal skills are required to document and communicate findings, escalate critical incidents, and interact with other members.   Preferred Qualifications   * SIEM software and EDR tool experience * Well versed in log analysis on various log sources from Next- |

|  |  |  |
| --- | --- | --- |
|  | * Attend meetings and training as required. * Participate in knowledge sharing sessions. * Recommend documentation improvements. | Gen firewalls, Domain Controllers, Linux operations systems, Anti-Virus logs, EDR/XDR, IPS/IDS, router and switch logs, etc.   * Experience in threat hunting, log integration, and incident case management. * 1-2 years of Security Operations Center experience. * 1-2 years of general IT support experience. * Any experience with networking |
| Korn Ferry | * Security Monitoring and Incident Response * Monitoring systems for signs anomalies, attacks, and unauthorized activities. * Investigate potential incidents and provide timely feedback. * Analyze events to identify trends, threats, and vulnerabilities. * Work to contain and remediate security incidents. * Threat Intelligence * Keep up to date with latest trends in cybersecurity threats, vulnerabilities, and best practices. * Security Infrastructure Management * Assist with the maintenance of existing security tools and technologies, such as SIEM, EDR and firewalls. * Contribute to the selection of new security tools. * Documentation and Reporting * Create and maintain detailed documentation of security processes and procedures. * Generate regular reports on security metrics, incidents, and trends for management review. * Collaboration and Communication * Work closely with other IT teams to identify and remediate security vulnerabilities. | * Hands on experience with security tools such as SIEM / EDR and vulnerability Management. * Proven experience in a security operations role. * In-depth knowledge of cybersecurity principles, threat landscapes, and attack vectors. * Experience working in a large, multinational, complex company. * Good knowledge of infrastructure concepts – such as Windows / Linux, DNS, AD and routing. * Knowledge and understanding of cloud computing concepts and service models. * Active learner with strong work ethic. * Proactive, flexible, responsive, and resourceful. * Ability to work both independently and collaboratively as a member of a small team. * Excellent organization and prioritization skills. * Ability to manage multiple projects and thrive in a fast- paced environment. * Strong attention to detail and analytical skills. * Strong communication and interpersonal skills. |

|  |  |  |
| --- | --- | --- |
|  |  | * Achieved a cybersecurity certification (e.g., CompTIA Security+, ISC2 SSCP, etc.) |

*Required Knowledge*

* + - Network Knowledge
      * Protocols and Services
      * Network Devices
      * Switching and Routing
    - Operating System
      * Windows Structure
      * Windows Event Logs
      * Windows Components (WMI, COM Objects, …)
      * Windows Audit Policies
      * Windows Defender
      * Windows Powershell
      * Linux Structure
      * Linux Kernel and Service Logs
      * Linux Components (Kernel Modules, Systemd, …)
      * Linux Auditd Service
      * Linux Firewall
      * Linux Bash Scripting
    - Sensors
      * NSM
      * Firewall
      * EDR
      * Syslog
      * Audit Logs
      * , …
    - Threat Knowledge
      * Kill-Chain
      * The most important techniques MITRE ATT&CK
      * Known Exploit Tools
      * Client Side Attack Knowledge
    - SIEM
      * SIEM Structure
      * SIEM Query
      * SIEM Report and Dashboard
      * Investigation with SIEM
    - Reporting

## Security Monitoring Tasks

### Monitoring the security operations center infrastructure

* a. Monitor the condition of the sensors and register a ticket if there is a failure in sending the log
* b. Monitoring the general process of sending logs in different time periods
* c. Checking the health status of the Splunk cluster through the MC component
* d. Checking the presence/absence of overlap in logs
* e. Review and follow up to improve the status of parsing and normalization of logs
* f. Checking the correct operation of important ES modules such as TI, Asset and Identity and...

### Monitoring, analysis and measurement of Notable Events issued by the ES app

* Analysis of Notable Events issued by checking the relevant raw log or the information contained in the Notable Event
* Detecting the criticality of the Notable Event and adding it to the Investigation panel if needed
* Elevation of frequent Notable Events by changing the relevant settings (with the coordination of the relevant manager or official)
* Suppressing incorrect Notable Events on a temporary basis (with the coordination of the relevant manager or official)

### Monitoring, analyzing and quantifying the output of USECASEs

* Monitoring predefined dashboards (ES dashboards, dashboards created in other apps)
* monitoring report output or scheduled or real-time alerts
* Adjusting the existing Report, Alert and Dashboard settings (Refresh Time, Schedule Time, Time Range, etc.)
* Compilation of use cases required by the organization with the cooperation of a higher level expert and the coordination of the relevant manager/responsible

### Monitor vulnerability news, security alerts and signs of threats

* Examining global vulnerability notification and warning sites such as NVD, US-CERT and...
* Checking daily security news from news sites such as TheHackerNews, NetworkWorld and...
* Check security alerts and news issued by Afta, Defense, Maher Center and Kashif
* Preparing/updating Lookup File from IOCs announced by Afta, Defense, Center Maher, Kashif and...

### Registration and tracking of security tickets

* Register a ticket if you see suspicious cases, checked along with the priority amount
* Track/do registered tickets and change ticket status
* Escalate critical tickets to a higher level expert or relevant manager/responsible

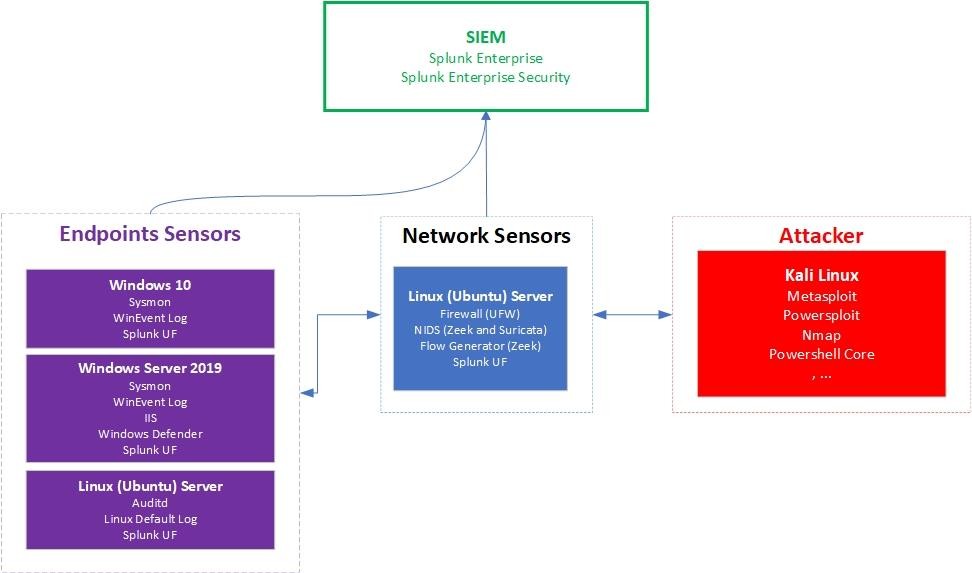
### Registering/changing/tracking the list of assets (servers, clients and users)

* Monitoring the latest changes recorded in the asset management system
* Matching the assets in the asset management system with the logs
* Enrich logs with information contained in the asset management system

### Preparation of reports and documents

* Preparation of shift report in the desired format of the organization
* Preparation of incident reports in the format considered by the organization
* Compilation of periodic performance reports (with topics including the organization's security status, identified incidents, technical activities performed, improvements made, etc.) in cooperation with other experts and relevant officials in the Security Operations Center

## Security Monitoring Infrastructure 2.1.Lab Architecture



### How to build your own lab

* + S1: ESXi or VMWare Workstation (With high resources)
  + S2: Using emulator (Eve-ng or pnetlab) + docker
  + S3: Using prepared lab (Attack Rang, Detection Lab (Deprecated))
    - Attack Range installation:
    - https://attack-range.readthedocs.io/en/latest/Attack\_Range\_Local.html
  + S4: Docker + Kubernetes

## SIEM Setup

### Splunk resource requirements (Single Instance):

* An x86 64-bit chip architecture
* 12 physical CPU cores, or 24 vCPU at 2 GHz or greater speed per core.
* 12 GB RAM.
* Hard disk depends on your data volume
* A 64-bit Linux or Windows distribution

*Ref: https://docs.splunk.com/Documentation/Splunk/9.2.0/Capacity/Referencehardware*

### Download and Install Splunk Enterprise

* Create a Splunk account
* Download Splunk Enterprise
* Install

*Tar zxvf <>.tgz -C /opt/*

*Alias splunk=”/opt/splunk/bin/splunk” Splunk start*

*Splunk enable boot-start*

* Create index
* Forwarding and Receiving config
* Data input (upd 514) config
* SSL enabled config
* Restart manually via shell

## Windows Setup and Logging

### Sysmon

* Download and place sysmon on drvie C
  + Sysmon64.exe -? Config <to get help about Sysmon config file>
  + Sysmon64.exe -u <If you want to uninstall the previous version>
  + Sysmon -h
* Create a sysmon config file
  + Sysmon Shell Tool
  + Online XML Parser
  + Notepad++
* Install Sysmon with the following command via powershell (Runas Admin)

*sysmon -accepteula -i c:\windows\config.xml*

*Get-Service | Where-Object Name -match "Sysmon" <To make sure Sysmon service installed>*

* Verfiy success log generation with the following command:

*Get-WinEvent -LogName “Microsoft-Windows-Sysmon/Operational” | Select-Object -first 10*

### Powershell Logging

* Module Logging

Logs execution details as it execute commands including any of variable initialization, command invocation and module loading (very high volume log) – Event Code = 4103

*Computer Configuration > Policies > Administrative Settings > Windows Components > Windows PowerShell > Turn on Module Logging*

*Channel: Microsoft-Windows-PowerShell/Operational*

* Script Block Logging

Record the processing of commands, script blocks, functions, and scripts whether invoked interactively, or through automation - Event Code = 4104

*Computer Configuration > Policies > Administrative Settings > Windows Components > Windows PowerShell > Turn on Powershell Script Block Logging*

*Channel: Microsoft-Windows-PowerShell/Operational*

* Transcript Logging

Transcript Logging provides a summary of what’s happening in a PowerShell session as if you were looking

over the shoulder of the person typing. It will provide the commands and the output of those commands

*Computer Configuration > Policies > Administrative Settings > Windows Components > Windows PowerShell > Turn on Powershell Transcription*

* After enable powershell logging, type:

*Gpupdate /force*

### Install and config SIEM agent

* Download the latest version of splunk UF
* Install the agent with the following commands

*Tar zxvf splunkforwarder\*.tgz -C <Target Directory>*

*$SplunkForwarder\_Home/bin/splunk start*

* Specify inderxer to forward

*/opt/splunkforwarder/bin/splunk add forward-server hostname.domain:9997*

* Test forwarder connection

*/opt/splunkforwarder/bin/splunk list forward-server*

* Add the following path to monitor

*[WinEventLog://Windows PowerShell/]  
disabled=0*

*#Monitor PowerShell transcript logs [monitor://C:\pstrans\\*\\*.txt] sourcetype = powershell:transcript index = powershell*

*disabled = 0 multiline\_event\_extra\_waittime = true time\_before\_close = 300*

*#Monitor PowerShell Windows Event Logs [WinEventLog://Microsoft-Windows-PowerShell/Operational] disabled = 0*

*renderXml = 1 index = powershell*

*source = XmlWinEventLog:Microsoft-Windows-PowerShell/Operational sourcetype = XmlWinEventLog*

*#Monitor PowerShell Windows Event Logs [WinEventLog://Microsoft-Windows-Sysmon/Operational] disabled = 0*

*renderXml = 1 index = powershell*

*source = XmlWinEventLog:Microsoft-Windows-Sysmon/Operational sourcetype = XmlWinEventLog*

## Linux Setup and Logging

### Set Timezone

*timedatectl set-timezone "Asia/Tehran"*

### Config network via netplan

*nano /etc/netplan/00-installer-config.yaml*

* Add/Change as the followings:

*network:*

*ethernets:*

*ens160:*

*dhcp4: no*

*addresses:*

*- 192.168.0.227/24*

*routes:*

*- to: default*

*via: 192.168.0.1*

*nameservers:*

*addresses: [8.8.8.8,8.8.8.4]*

*version: 2*

*renderer: networkd*

* Then:

*Netplan apply*

### Install and config SIEM agent

* Download the latest version of splunk UF
* Install the agent with the following commands

*Tar zxvf splunkforwarder\*.tgz -C <Target Directory>*

*$SplunkForwarder\_Home/bin/splunk start*

* Specify inderxer to forward

*/opt/splunkforwarder/bin/splunk add forward-server hostname.domain:9997*

* Test forwarder connection

*/opt/splunkforwarder/bin/splunk list forward-server*

* Add the following path to monitor

*/opt/splunkforwarder/bin/splunk add monitor <the following path> -index soc -sourcetype %app%*

* Check sending log status in endpoint

*tcpdump -i ens160 port 9997 and host 192.168.0.247*

## NSM Setup and Logging

Add 3 interface to monitor, internal network and external network

* Config NSM to act as a router

*Sysctl net.ipv4.ip\_forward=1*

### Install suricata

*add-apt-repository ppa:oisf/suricata-stable apt install suricata*

*systemctl enable suricata.service*

* Initial Configuration

*nano /etc/suricata/suricata.yaml*

*# (Optional) Enabling Community Flow ID community-id: true*

*# Set default interface to monitor af-packet:*

*- interface: eth0*

*# Config live rule reloadin detect-engine:*

*- rule-reload: true*

* Update Suricata rulsets

*suricata-update list-sources suricata-update*

*suricata-update enable-source tgreen/hunting*

* Validating Suricata conf file

*suricata -T -c /etc/suricata/suricata.yaml -v*

### Install Zeek

*apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python3-dev swig zlib1g-dev*

* Download Zeek source from <https://download.zeek.org/zeek-6.0.3.tar.gz>and install as bellow:

*tar -xzf zeek-<verison>.tar.gz cd zeek*

*./configure Make*

*Make install*

* Add Zeek binary to PATH env to be used as a default linux binaries

*nano ~/.bashrc*

*export PATH=/usr/local/zeek/bin:$PATH source ~/.bashrc*

* Initial config

*cd /usr/local/zeek/etc nano node.cfg*

*#change the interface name*

* Validating Zeek config

*zeekctl check*

Once you get “zeek scripts are ok.” at the end you can deploy zeek, using below command.

zeekctl deploy zeekctl status

To view Zeek logs

*cd /usr/local/zeek/logs/current tail -f conn.log*

Path to monitor:

/*var/log/ufw/ufw\**

*/var/log/suricata/eve.json (sourcetype json)*

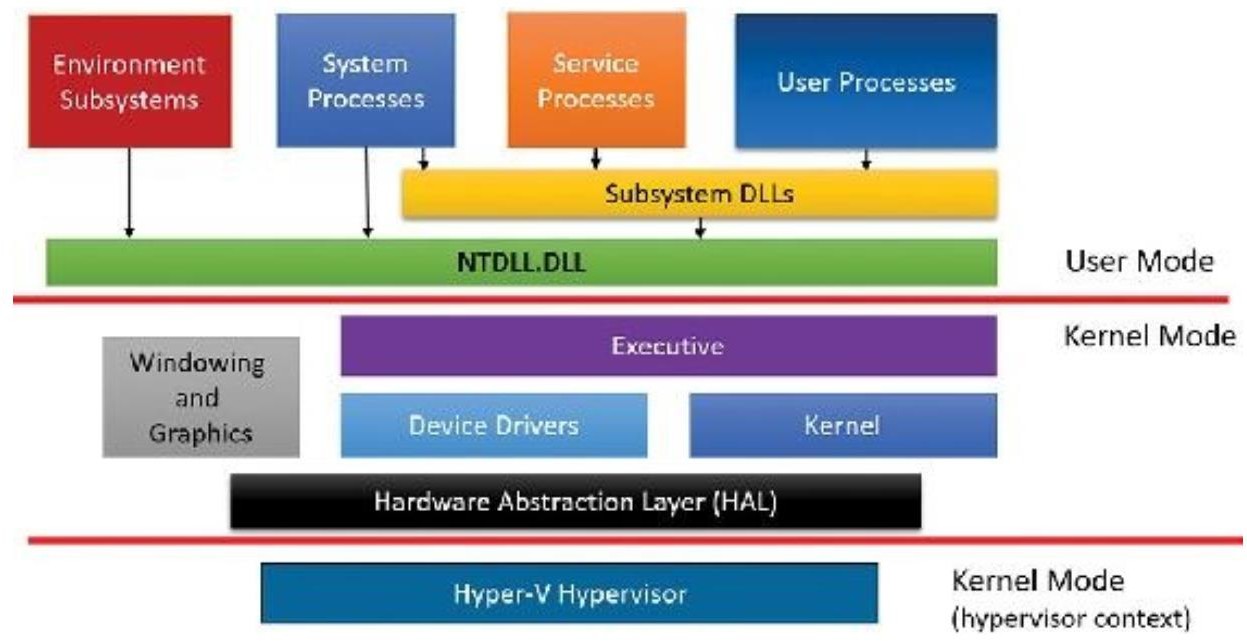
*/usr/local/zeek/logs/current (sourcetype bro:json)*

*/var/log/audit/audit.log (sourcetype linux\_audit)*

## Splunk as a monitoring Solution

# Endpoint Security Monitoring

## Windows Components 3.1.Windows Architecture



User Process (Applications):

* + Windows default apps (calc.exe, notepad.exe, explorer.exe, ..)
  + Third party apps (notepad++.exe, chrome.exe, …)

Get-Process -IncludeUserName #List all of processes

Get-Process -IncludeUserName | Where-Object { $\_.SI -eq 1 -and $\_.UserName -match "Ahangari" } # list all usermod processes

Process Explorer > Select-Column > Session, UserName

Service Process:

These are processes that host Windows services, such as the Task Scheduler and Print Spooler services. A Windows Service is an executable application that the operating system runs in the background. It does not require a logged-in user session to run. In Windows, the Service Control Manager (SCM) manages all Windows service processes

Get-Service

Sc.exe /query

Some services are hosted by svchost.exe (is a shared-service process that serves as a shell for loading services from DLL files) or run indepentely (.exe extension)

System Process:

These are fixed, or hardwired, processes, such as the logon process and the Session Manager, that are not Windows services. That is, they are not started by the Service Control Manager.

Get-Process -IncludeUserName | Where-Object { $\_.UserName -eq "NT AUTHORITY\SYSTEM" }

Environment Subsystem:

Components of the Microsoft Windows NT or Windows 2000 operating system that support the running of applications from different operating system architectures. Environmental subsystems provide the necessary “environment” in which these applications can run. They are an essential part of the Windows NT operating system that enables cross-platform support for applications written for different operating systems.

Windows NT and Windows 2000 include the following environmental subsystems:

* Win32 subsystem for running 32-bit Windows applications
* OS/2 subsystem for running OS/2 1.X character-based applications (does not support the OS/2 Presentation Manager GUI or Warp versions)
* POSIX subsystem for running POSIX.1-compliant applications

NTDLL.DLL

Short for NT Layer DLL, is a core component of the Windows operating system. It contains a collection of functions and system calls that are essential for the proper functioning of various processes and applications. These functions include memory management, exception handling, and input/output operations.

NTDLL is responsible for handling system-level operations and acts as an interface between the software and the hardware components of your computer. It is loaded into memory during the boot process and remains active throughout the entire system operation.

.\strings.exe -n 20 C:\Windows\System32\ntdll.dll | findstr.exe /i "Zw" # find function starts with zw – Native System APIs

Dependency Walker # To see imported and exported functions (API)

Executive:

The Windows executive contains the base OS services, such as memory management, process and thread management, security, I/O, networking, and inter-process communication.

The executable file is Ntoskernel.exe

Dependency Walker # To see dependencies

Windows kernel:

This consists of low-level OS functions, such as thread scheduling, interrupt and exception dispatching, and multiprocessor synchronization. It also provides a set of routines and basic objects that the rest of the executive uses to implement higher-level constructs.

Dependency Walker # To see dependencies

The executable file is Ntoskernel.exe

Device drivers:

This includes both hardware device drivers, which translate user I/O function calls into specific hardware device I/O requests, and non-hardware device drivers, such as file system and network drivers

Msinfo32.exe (System Information)

Hardware Abstraction Layer (HAL):

This is a layer of code that isolates the kernel, the device drivers, and the rest of the Windows executive from platform-specific hardware differences

Windowing and Graphics system:

This implements the graphical user interface (GUI) functions (better known as the Windows USER and GDI functions), such as dealing with windows, user interface controls, and drawing

Hypervisor Layer:

This is composed of a single component: the hypervisor itself. There are no drivers or other modules in this environment. That being said, the hypervisor is itself composed of multiple internal layers and

services, such as its own memory manager, virtual processor scheduler, interrupt and timer management, synchronization routines, partitions (virtual machine instances) management and inter-partition communication (IPC), and more.

## Windows Process

Although programs and processes appear similar on the surface, they are fundamentally different. A *program* is a static sequence of instructions, whereas a *process* is a container for a set of resources used when executing the instance of the program.

### Windows Process Contents:

At the highest level of abstraction, a Windows process comprises the following:

**A private virtual address space** This is a set of virtual memory addresses that the process can use. **An executable program** This defines initial code and data and is mapped into the process’s virtual address space.

**A list of open handles** These map to various system resources such as semaphores, synchronization objects, and files that are accessible to all threads in the process.

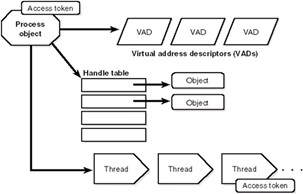
**A security context** This is an *access token* that identifies the user, security groups, privileges, attributes, claims, capabilities, User Account Control (UAC) virtualization state, session, and limited user account state associated with the process, as well as the AppContainer identifier and its related sandboxing information.

**A process ID** This is a unique identifier, which is internally part of an identifier called a *client ID*.

**At least one thread of execution** Although an “empty” process is possible, it is (mostly) not useful

Think of a process as a room and a thread as a person in the room. A program is a set of instructions for the person in the room to carry out. Looking at it in this fashion, it is easy to see that the process itself doesn't do any work, but the thread does. A thread lives in a process, and executes the instructions of the program.

Show processes (notepad.exe) with procexp64.exe



### Threads

A thread is an entity within a process that Windows schedules for execution. Without it, the process’s program can’t run. A Thread is a stream of sequential machine-code instructions that the processor executes. A thread includes the following essential components:

**The contents of a set of CPU registers representing the state of the processor**

**Two stacks—one for the thread to use while executing in kernel mode and one for executing in user mode**

**A private storage area called thread-local storage (TLS) for use by subsystems, run-time libraries, and DLLs**

**A unique identifier called a thread ID (part of an internal structure called a client ID; process IDs and thread IDs are generated out of the same namespace, so they never overlap) In addition, threads sometimes have their own security context, or token, which is often used by multithreaded server applications that impersonate the security context of the clients that they serve.**

The volatile registers, stacks, and private storage area are called the **thread’s context**. Because this information is different for each machine architecture that Windows runs on, this structure, by necessity, is architecture-specific. The Windows GetThreadContext function provides access to this architecture-specific information (called the CONTEXT block)

### Handles

A Handle is a logical association with a shared resource like a file, Window, memory location, etc. When a thread opens a file, it establishes a "handle" to the file. Handles are used to link to transitory or environmental resources outside the processes memory structure.

### Status

**Running:** For processes that don’t have any user interface, Running should be the normal case, although the threads may all be waiting for something, such as a kernel object being signaled or some I/O operation to complete.

**Suspended:** The other option for such processes is Suspended, and this happens **if all the threads in the process are in a suspended state.** This is unlikely to occur by the process itself, but can be achieved programmatically by calling the undocumented NtSuspendProcess native API on the process, typically through a tool. For processes that create a user interface, the Running status value means that the UI is responsive. In other words, the thread that created the window(s) is waiting for UI input (technically, the message queue associated with the thread). The Suspended state is possible just like in the non-UI case, but for Windows Apps (those hosting the Windows Runtime), Suspended normally occurs when the app loses its foreground status by being minimized by the user. Such processes are suspended after 5 seconds so that they don’t consume any CPU or networking resources, thus allowing the new foreground app to get all machine resources. This is especially important for battery-powered devices, such as tablets and phones.

**Note Responding:** The third possible value for Status is Not Responding. This can happen if a thread within the process that created the user interface has not checked its message queue for UI-related activity for at least 5 seconds.

Each process also points to its parent or creator process (which may be, but is not always, its creator process). If the parent no longer exists, this information is not updated. Therefore, it is possible for a process to refer to a nonexistent parent. This is not a problem, because nothing relies on this information being kept current. In the case of the Process Explorer tool, the start time of the parent process is taken into account to avoid attaching a child process based on a reused process ID. Why would a parent process not be the same as its creator? In certain cases, some processes that appear to be created by a certain user application might involve the help of a broker, or helper, process, which is responsible for calling the process creation API. In such cases, it would be confusing (and sometimes incorrect, if handle or address space inheritance is needed) to display the broker process as the creator, and a “re-parenting” is done.

### Access Token and Security Descriptor

### …

## Windows APIs

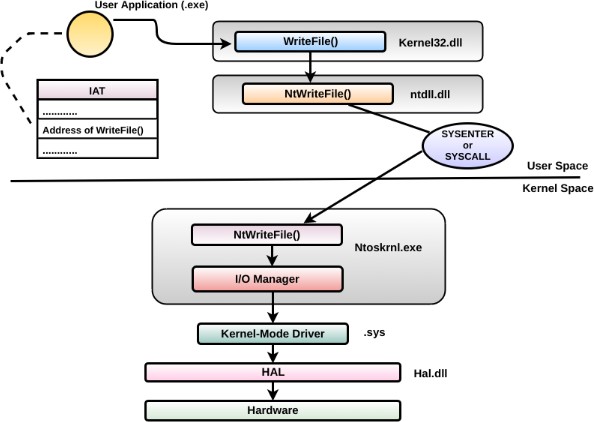
### What is Windows API

The Windows API serves as a gateway to unlock the vast potential of Windows development.

It provides developers with a set of functions, interfaces, and protocols that enable them to interact with the Windows operating system and its underlying resources.

The API acts as an intermediary, allowing applications to access system-level functionality and perform tasks such as file manipulation, network communication, and GUI development.

### API Call Flow



*API Call with Powershell*

Creating a message box is probably one of the most straight forward examples as the API call requires very little input. Make sure to check out the [pinvoke](http://www.pinvoke.net/default.aspx/user32/MessageBox.html) entry for MessageBox to get a head-start on the structure definition and the [MSDN](https://msdn.microsoft.com/en-us/library/windows/desktop/ms645505(v%3Dvs.85).aspx) entry to get a better understanding of the structure parameters.

The C++ function structure from MSDN can be seen below.

int WINAPI MessageBox(

\_In\_opt\_ HWND hWnd,

\_In\_opt\_ LPCTSTR lpText,

\_In\_opt\_ LPCTSTR lpCaption,

\_In\_ UINT uType

);

This easily translates to c#, it is almost a literal copy/paste of the example on pinvoke.

Add-Type -TypeDefinition @" using System;

using System.Diagnostics;

using System.Runtime.InteropServices;

public static class User32

{

[DllImport("user32.dll", CharSet=CharSet.Auto)] public static extern bool MessageBox(

IntPtr hWnd, /// Parent window handle String text, /// Text message to display String caption, /// Window caption

int options); /// MessageBox type

}

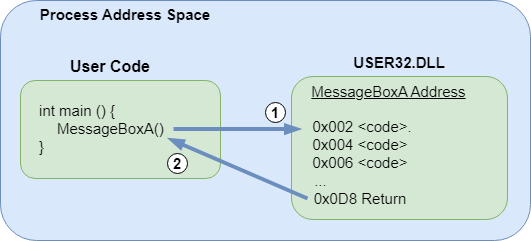
"@

[User32]::MessageBox(0,"Text","Caption",0) |Out-Null Executing the code above pops the expected message box.

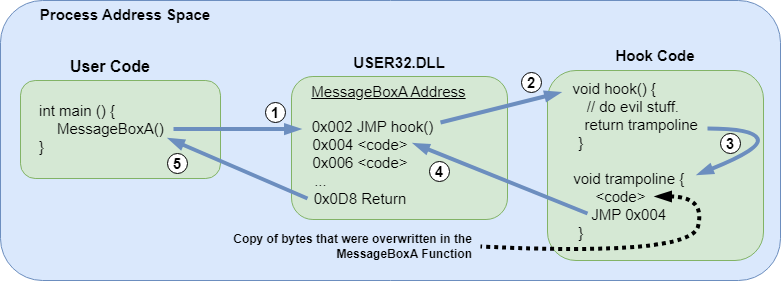
### API Hooking

API hooking is a technique by which we can instrument and modify the behavior and flow of API calls. API hooking can be done using various methods on Windows. Techniques include memory break point and . DEP and JMP instruction insertion.

**Normal Operation Flow (No Hook)**



**Hooked Operation Flow**



How to Monitor API Calls

<https://www.lares.com/blog/hunting-in-the-sysmon-call-trace/>

Sysinternal – Procmon64

API Monitor Tool – Spy

### The most interesting APIs to Monitoring

#### API Call Convention

Windows API Calls which are involved with text manipulation are often appended with **‘A’ or ‘W’**. The

‘A’ is used to identify functions which work with ANSI strings as input(s) and output(s). The ‘W’ is used to identify functions which work with Unicode strings as input(s) and output(s). By the way, Windows also has native support for Unicode strings, because they’re widely supported and work well with several languages and character sets.

Similarly, the **‘Ex’** at the end of several API calls standards for ‘Extended’ - as in an extension of

functionality along with a more detailed interface. For example, the import ‘CreateFile’ works with 4 parameters, whereas the ‘CreateFileEx’ call works with 17 parameters, which much more detail as you can imagine based on the increase in parameters count.

Here’s a list of the most intriguing API calls

#### Registry API Calls

The following pair of API calls can be used to manage the registry, modify the values, and spawn other keys as well. This behavior is particularly useful to identify persistence via the registry. Other than that, almost every configuration and setting is stored in the Registry. You’re likely to find the registry keys for services or other paths used by the malware if you monitor these calls.

|  |  |
| --- | --- |
| RegSetValueEx | Set the data-type and value of a certain registry key |
| RegCreateKeyEx | Create a registry key |
| RegEnumKeyEx | Extract the data-type and value of a specific registry key |
| RegQueryValueEx | Get the data-type and value of a specific registry key |
| RegOpenKeyEx | Open a specific key |
| RegCloseKey | Close the open handle to the specified registry key |
| RegEnumValue | Enumerate through the values from a given registry key path |

#### Threads and Processes

Monioring processes and threads for unknown activity is also important. Often, malware spawn processes and threads to carry out several tasks without being noticed.

|  |  |  |  |
| --- | --- | --- | --- |
|  | CreateProcess | Create a new process along with threads |  |
|  | GetCurrentProcess | Retrieve a pseudo-handle for a process |  |
|  | GetCurrentProessId | Retrieve the key process identifier |  |
|  | CreateThread | Create a thread within a process |  |
|  | SetThreadPriority/GetThreadPriority | Set and retrieve the priority for the thread within the context of a process |  |
|  | GetProcessTimes | Retrieves the timing information about the process |  |
|  | ExitProcess | Exit a process and the threads it spawned |  |

#### System Information

API calls can also extract system information, which is of much use to a malware. This information particularly identifies a system. The actor can even exfiltrate this data out by means of C2 communication.

|  |  |
| --- | --- |
| GetFileVersionInfo | Retrieves the information about a particular file |
| GetFileVersionInfoSize | Checks whether retrieval of information is possible and returs size, if true |
| GetSystemMetrics | Returns the informationn about a particular metric or configuration. Several values are available to the user, along with the option to pass it as a parameter |
| GetSystemInfo/GetNativeSystemInfo | Retrieves system information |
| IsDebuggerPresent | Identifies a user-mode debugger on the excecutable |
| QueryPerformanceCounter | Get the current value of the performance monitor |

#### Files

File-based API calls accumulate to a large count. But, just going through the list, I can pick a few apart which are extremely sketchy (especially for an executable that pretends to be a PDF viewer).

|  |  |
| --- | --- |
| DeleteFile | Delete an existing file |
| GetFileType | Retrieve the type of the file |
| MoveFile | Move the file from one location to another |
| GetFileAttributes | Retrieve the attributes of the specified file |
| CopyFile | Copy the file from one location to another |
| FindFirstFileEx | Search a file with information |
| GetFileSize | Retrieves the size of the specified file |
| ReadFile | Read data from the I/O device or a file |

#### Keyboard, Mouses, and Input Devices

Quite obviously, its API calls which are used to send input data to the processor and so on. These API calls are also used by malware (especially keyloggers) with the intent to steal data from a computer and dispatch it away. Here’s a list of API calls found in this sample:

|  |  |
| --- | --- |
| EnableWindow | Ability to enable or disable keyboard or mouse input to the said Window |
| GetAsyncKeyState | Identify the state of the keys (pressed or released) |

#### Cryptography

There’s also evidence of some cryptographic API calls being loaded with the executable. These can

indeed encrypt the data and hide it from the user’s eyes before sending it off to the HTTP server (yes, we’ll go over the networking API calls next). Here’s a list of those API calls:

|  |  |
| --- | --- |
| CryptDeriveKey | Generates cryptographic session keys |
| CryptEncrypt | Encrypts data |
| CryptDecrypt | Decrypts data which is encrypted by CryptEncrypt |
| CryptCreateHash | Obtain handle to the hash object for the encryption function (passed off as a parameter). Also initiates the hashing of a stream of data |
| CryptHashData | Adds the data to a specified hash object |

#### Networking

Lastly, the Windows Sockets library imports several functions and API calls which are key to spawn connections and used by the malware for C2 communication.

|  |  |
| --- | --- |
| HttpQueryInfo | Retrieve headers related to HTTP requests |
| HttpSendRequestEx | Send request to a web server |
| HttpEndRequest | Break communication with a web server |
| HttpOpenRequest | Spawn an HTTP request handle |

|  |  |
| --- | --- |
| InternetConnectA | Open an FTP or HTTP session for a website |
| InternetGetConnectedState | Retrieves the connection state for the system |
| InternetSetOptionA | Change Internet options |
| InternetWriteFile | Write data to an Internet file whose handle has already been acquired |
| InternetCrackUrlA | Divide the URL into parts |
| InternetSetStatusCallbackA | Functions made during a state change or progress during an operation |

## Windows Event Log

### writing a log fileWindows Event Log Generation Process

Note: Attacker can stop Event-Logging service How to: by suspending Event Log threads

Demo: Process Hacker > Services – Event Log > Go to Process > SVCHost.exe – Suspend Below is the code for the technique that at a high level works like this:

* + - Open a handle to Service Control Manager with OpenSCManagerA
    - Open a handle to EventLog service with OpenServiceA
    - Retrieve svchost.exe (hosting EventLog) process ID with QueryServiceStatusEx
    - Open a handle to the svchost.exe process (from step 3)
    - Get a list of loaded modules loaded by svchost.exe EnumProcessModules
    - Loop through the list of svchost loaded modules, retrieved in step 5, find their names with GetModuleBaseName and find the base address of the module wevtsvc.dll - this is the module containing EventLog service inner-workings
    - Get wevtsvc.dll module info with GetModuleInformation. It will return a structure with module's start address and its image size - we will need these details later, when degerming if EventLog service thread's fall into wevtsvc.dll module's memory space
    - Enumerate all the threads inside svchost.exe with Thread32First and Thread32Next
    - For each thread from step 8, retrieve the thread's start address with NtQueryInformationThread
    - For each thread from step 8, check if the thread's start address belongs to the wevtsvc.dll memory space inside svchost.exe
    - If thread's start address is inside the wevtsvc.dll memory space, this is our victim thread and we suspend it with SuspendThread
    - EventLog service is now disabled

### Windows Event Log Elements

* + - Log name/key (Channel) - The key refers to each logging component's classification, indicating the log's name to which events from these components will be written.
      * The system event logs will include events logged by system-level components such as the Windows Update Client.
      * The application event logs are slightly different; these include events related to different services as well as applications that are installed or being installed on the Windows machine. If an event log is recorded when an application fails while running or during set- up, it should be tied to the application key.
      * Security event logs typically include audit records of successful and failed login attempts.
    - Level - Is the event being logged strictly for informational purposes, or does it indicate a critical error? The event level will tell you the severity of the event being recorded. Event levels include critical, error, warning, information and verbose.
    - Date/time - This refers to the date and time when the event was recorded.
    - Source - This is the name of the component that triggers the event log. In many cases, it will be the name of the application or process that writes the event log. For example, suppose the event is related to the failure of a database application on the machine. In that case, the event source may be the name of the database application that experienced the failure.
    - Event ID - The event ID is meant to serve as an identifier for a distinct logged event.
    - Task (Event) category - The task category serves as additional information to assist with debugging an application or system issue. The developers of a particular application can define the categories to help provide context for a particular event.
    - User - This can refer to the user logged in to a particular Windows machine at the time the event was recorded. For example, when installing an application, the username for the administrator logged into the machine will likely be reflected in the event log for the installation event.
    - Computer - The name of the machine that logged the event.

### Windows Event Log Event Data

Example: Event Code 4688 – Process Creation

|  |  |  |
| --- | --- | --- |
| A new process has been created.  Creator |  | Subject: |
| Security Account Account  Logon ID: 0x3E7 |  | ID: SYSTEM  Name: RFSH$ Domain: LAB |
| Target Security Account Account  Logon ID: 0x2C9D82 |  | Subject: ID: LAB\rsmith Name: rsmith Domain: LAB |
| Process New | Process | Information: ID: 0x2e0e4 |

New Process Name: C:\Windows\System32\RuntimeBroker.exe Token Elevation Type: %%1938 Mandatory Label: Mandatory Label\Medium Mandatory Level Creator Process ID: 0x268

Creator Process Name: C:\Windows\System32\svchost.exe Process Command Line:

Creator Subject:

The user and logon session that started the program.

Security ID: The SID of the account.

Account Name: The account logon name.

Account Domain: The domain or - in the case of local accounts - computer name.

Logon ID: A semi-unique (unique between reboots) number that identifies the logon session. Logon ID allows you to correlate backwards to the logon event (4624) as well as with other events logged during the same logon session.

Target Subject:

Added in Win2016/10. These fields only apply when the process is started under a different user account. By default, a new process runs under the same account and logon session as the creator process. One of the examples below shows the SYSTEM account starting RuntimeBroker.exe as a different user.

Security ID: The SID of the account.

Account Name: The account logon name.

Account Domain: The domain or - in the case of local accounts - computer name.

Logon ID: A semi-unique (unique between reboots) number that identifies the logon session. Logon ID allows you to correlate backwards to the logon event (4624) as well as with other events logged during the same logon session.

Process Information:

New Process ID: A semi-unique (unique between reboots) number that identifies the process. Process ID allows you to correlate other events logged during the same process. To determine when the program ended look for a subsequent event 4689 with the same Process ID.

New Process Name: The full path of the executable

Token Elevation Type: This is useful for detecting when users running under User Account Control consent to running a program with admin authority - look for Type 2.

Token Elevation Type: Token elevation is about User Account Control

%%1936 - Type 1 is a full token with no privileges removed or groups disabled. A full token is only used if User Account Control is disabled or if the user is the built-in Administrator account or a service account.

%%1937 - Type 2 is an elevated token with no privileges removed or groups disabled. An elevated token is used when User Account Control is enabled and the user chooses to start the program using Run as administrator. An elevated token is also used when an

application is configured to always require administrative privilege or to always require maximum privilege, and the user is a member of the Administrators group.

%%1938 - Type 3 is the normal value when UAC is enabled and a user simply starts a program from the Start Menu. It's a limited token with administrative privileges removed and administrative groups disabled. The limited token is used when User Account Control is enabled, the application does not require administrative privilege, and the user does not choose to start the program using Run as administrator.

Mandatory Label: (new to Win10) In addition to each objects Discretionary Access Control List (permissions on a file) Windows also enforces Mandatory Integrity Control (MIC) over object access attempts which compares the object's integrity label to the the integrity level on the process trying to access the object. This field documents the integrity of the process which is determined from the user integrity level and the file integrity level of the EXE. The actual values observed so far include

Mandatory Label\Low Mandatory Level

Mandatory Label\Medium Mandatory Level

Mandatory Label\System Mandatory Level

|  |  |  |  |
| --- | --- | --- | --- |
| SID | RID | RID label | Meaning |
| S-1-16- 0 | 0x00000000 | SECURITY\_MANDATORY\_UNTRUSTED\_RID | Untrusted. |
| S-1-16- 4096 | 0x00001000 | SECURITY\_MANDATORY\_LOW\_RID | Low integrity. |
| S-1-16- 8192 | 0x00002000 | SECURITY\_MANDATORY\_MEDIUM\_RID | Medium integrity. |
| S-1-16- 8448 | 0x00002100 | SECURITY\_MANDATORY\_MEDIUM\_PLUS\_RID | Medium high integrity. |
| S-1-16- 12288 | 0X00003000 | SECURITY\_MANDATORY\_HIGH\_RID | High integrity. |
| S-1-16- 16384 | 0x00004000 | SECURITY\_MANDATORY\_SYSTEM\_RID | System integrity. |
| S-1-16- 20480 | 0x00005000 | SECURITY\_MANDATORY\_PROTECTED\_PROCESS\_RID | Protected process. |

Creator Process ID: Identifies the processes that started this process. Look for a preceding event 4688 with a New Process ID that matches this Creator Process process ID - or if on Win10 or later look at the next field to get EXE name of the parent process.

Creator Process Name: (new to Win10) This useful field documents the name of the program that started this new process.

Process Command Line: (new to Win2012R2) If enabled this field documents the command line arguments (including any passwords) passed into the EXE when the process was started. See “Administrative Templates\System\Audit Process Creation\Include command line in process creation events” in group policy.

Example: Event Code 4624 – An account was successfully logged on.

Subject:

Security ID: SYSTEM Account Name: DESKTOP-LLHJ389$

Account Domain: WORKGROUP Logon ID: 0x3E7

Logon Information:

Logon Type: 7

Restricted Admin Mode: - Virtual Account: No

Elevated Token: No

Impersonation Level: Impersonation New Logon:

Security ID: AzureAD\RandyFranklinSmith Account Name: [rsmith@montereytechgroup.com](mailto:rsmith@montereytechgroup.com)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Account |  | Domain: | AzureAD | |
| Logon |  | ID: | 0xFD5113F | |
| Linked |  | Logon | ID: | 0xFD5112A | |
| Network | | Account | Name: | | - |
| Network | | Account | Domain: | | - |

|  |  |  |
| --- | --- | --- |
| Process | Logon GUID: | {00000000-0000-0000-0000-000000000000}  Information: |
|  | Process  Process Name: | ID: 0x30c  C:\Windows\System32\lsass.exe |
| Network | Workstation | Information:  Name: DESKTOP-LLHJ389 |

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Network | Address: | - |
|  | Source | Port: | - |

Detailed Authentication Information: Logon Process: Negotiate Authentication Package: Negotiate

Transited Services: - Package Name (NTLM only): -

Key Length: 0

Subject:

Identifies the account that requested the logon - NOT the user who just logged on. Subject is usually Null or one of the Service principals and not usually useful information. See New Logon for who just logged on to the system.

* + Security ID
  + Account Name
  + Account Domain
  + Logon ID



Logon Information:

* + Logon Type: See below

This is a valuable piece of information as it tells you HOW the user just logged on:

|  |  |
| --- | --- |
| Logon Type | Description |
| 2 | Interactive (logon at keyboard and screen of system) |
| 3 | Network (i.e. connection to shared folder on this computer from elsewhere on network) |
| 4 | Batch (i.e. scheduled task) |
| 5 | Service (Service startup) |
| 7 | Unlock (i.e. unattended workstation with password protected screen saver) |
| 8 | Network Cleartext (Logon with credentials sent in the clear text. Most often indicates a logon to IIS with "basic authentication") [See this article for more information.](http://www.windowsitpro.com/Articles/Print.cfm?ArticleID=45214) |

|  |  |
| --- | --- |
| 9 | New Credentials such as with RunAs or mapping a network drive with alternate credentials. This logon type does not seem to show up in any events. If you want to track users attempting to logon with alternate credentials see [4648](http://www.ultimatewindowssecurity.com/wiki/SecurityLogEventID4648.ashx). MS says "A caller cloned its current token and specified new credentials for outbound connections. The new logon session has the same local identity, but uses different credentials for other network connections." |
| 10 | Remote Interactive (Terminal Services, Remote Desktop or Remote Assistance) |
| 11 | Cached Interactive (logon with cached domain credentials such as when logging on to a laptop when away from the network) |

Remaining logon information fields are new to Windows 10/2016

* + Restricted Admin Mode: Normally "-"."Yes" for incoming Remote Desktop Connections where the client specified /restrictedAdmin on the command line. Restricted admin mode is an important way to limit the spread of admin credentials in ways they can be harvested by malware using pass-the-hash and related techniques. You should only see with for logon type 10. When you remote desktop into a server with /restrictedAdmin you get full authority on that server but it doesn't carry with you if you access other systems from within that RDP session. This field allows you to detect RDP sessions that fail to use restricted admin mode.
  + Virtual Account: Normally "No". This will be Yes in the case of services configured to logon with a "Virtual Account". Virtual Accounts only come up in Service logon types (type 5), when Windows starts a logon session in connection with a service starting up. You can configure services to run as a virtual account which is what Microsoft calls a "managed local account". They're "domain" is "NT Service" as in an instance of MS SQL Server named Supercharger running as NT SERVICE\MSSQL$SUPERCHARGER.
  + Elevated Token: Yes or No. It will be Yes if the user is a member of Administrators - kind of... The "kind of" applies to interactive logons, when you are an admin and you have User Account Control (UAC) enabled. Then when you logon you actually get 2 logon sessions. One without the Administrators SID and related privileges in your security token and another session with all that authority. Everything you do happens under the unprivileged logon session until you attempt to run something requiring admin authority. After you approve the UAC dialog box, Windows runs that one operation under the other logon sesson. So in the log you will see 2 of these events, one where this field is Yes and other No. The 2 logon sessions are connected by the Linked Logon ID described below.

Impersonation Level: (Win2012 and later)

|  |  |
| --- | --- |
| Anonymous | Anonymous COM impersonation level that hides the identity of the caller. Calls to WMI may fail with this impersonation level. |
| Default | Default impersonation. |

|  |  |
| --- | --- |
| Delegate | Delegate-level COM impersonation level that allows objects to permit other objects to use the credentials of the caller. This level, which will work with WMI calls but may constitute an unnecessary security risk, is supported only under Windows 2000. |
| Identify | Identify-level COM impersonation level that allows objects to query the credentials of the caller. Calls to WMI may fail with this impersonation level. |
| Impersonate | Impersonate-level COM impersonation level that allows objects to use the credentials of the caller. This is the recommended impersonation level for WMI calls. |

New Logon:

The user who just logged on is identified by the Account Name and Account Domain. You can determine whether the account is local or domain by comparing the Account Domain to the computer name. If they match, the account is a local account on that system, otherwise a domain account.

* + Security ID: the SID of the account
  + Account Name: Logon name of the account
  + Account Domain: Domain name of the account in either the DNS name (can be upper or lowercase) or pre-Win2k NETBIOS domain name. In the case of special subjects (well known security principals) like SYSTEM, LOCAL SERVICE, NETWORK SERVICE, ANONYMOUS LOGON this field will be "NT AUTHORITY". It can also be "NT Service" as in the case of virtual accounts for services. See above. Finally, if the account is a local account, this field will be the name of the computer.
  + Logon ID: a semi-unique (unique between reboots) number that identifies the logon session just initiated. Any events logged subsequently during this logon session will report the same Logon ID through to the logoff event [4647](http://www.ultimatewindowssecurity.com/wiki/SecurityLogEventID4647.ashx) or [4634.](http://www.ultimatewindowssecurity.com/wiki/SecurityLogEventID4634.ashx)
  + Linked Login ID: (Win2016/10) This is relevant to User Account Control and interactive logons. When an admin logs on interactively to a system with UAC enabled, Windows actually creates 2 logon sessions - one with and one without privilege. This is called a split token and this fields links the 2 sessions to each other. See Elevated Token above.
  + Network Account Name: (Win2016/10) This appears to always be "-". It seems connected to LogonUser() with LOGON32\_LOGON\_NEW\_CREDENTIALS
  + Network Account Domain: (Win2016/10) see above
  + Logon GUID: Supposedly you should be able to correlate logon events on this computer with corresponding authentication events on the domain controller using this GUID. Such as linking [4624](http://www.ultimatewindowssecurity.com/wiki/SecurityLogEventID4624.ashx) on the member computer to [4769](http://www.ultimatewindowssecurity.com/wiki/SecurityLogEventID4769.ashx) on the DC. But the GUIDs do not match between logon events on member computers and the authentication events on the domain controller.

Process Information:

* + Process ID is the process ID specified when the executable started as logged in [4688](http://www.ultimatewindowssecurity.com/wiki/SecurityLogEventID4688.ashx).
  + Process Name: identifies the program executable that processed the logon. This is one of the trusted logon processes identified by [4611](http://www.ultimatewindowssecurity.com/wiki/SecurityLogEventID4611.ashx).

Network Information:

This section identifies WHERE the user was when he logged on. Of course if logon is initiated from the same computer this information will either be blank or reflect the same local computers.

* + Workstation Name: the computer name of the computer where the user is physically present in most cases unless this logon was initiated by a server application acting on behalf of the user. Workstation may also not be filled in for some Kerberos logons since the Kerberos protocol doesn't really care about the computer account in the case of user logons and therefore lacks any field for carrying workstation name in the ticket request message.
  + Source Network Address: the IP address of the computer where the user is physically present in most cases unless this logon was initiated by a server application acting on behalf of the user. If this logon is initiated locally the IP address will sometimes be 127.0.0.1 instead of the local computer's actual IP address. This field is also blank sometimes because Microsoft says "Not every code path in Windows Server 2003 is instrumented for IP address, so it's not always filled out."
  + Source Port: identifies the source TCP port of the logon request which seems useless since with most protocols source ports are random.

Detailed Authentication Information:

* + Logon Process: indicates a logon initiated by User Account Control
  + Authentication Package:
  + Transited Services: This has to do with server applications that need to accept some other type of authentication from the client and then transition to Kerberos for accessing other resources on behalf of the client. Transmitted services are populated if the logon was a result of a S4U (Service For User) logon process. S4U is a Microsoft extension to the Kerberos Protocol to allow an application service to obtain a Kerberos service ticket on behalf of a user – most commonly done by a front-end website to access an internal resource on behalf of a user.
  + Package name: If this logon was authenticated via the NTLM protocol (instead of Kerberos for instance) this field tells you which version of NTLM was used. See security option "Network security: LAN Manager authentication level". This field only populated if Authentication Package

= NTLM. Possible values: “NTLM V1”, “NTLM V2”, “LM”

* + Key Length: Length of key protecting the "secure channel". See security option "Domain Member: Require strong (Windows 2000 or later) session key". If value is 0 this would indicate security option "Domain Member: Digitally encrypt secure channel data (when possible)" failed. MS says the length of NTLM Session Security key. Typically it has 128 bit or 56 bit length. This parameter is always 0 if “Authentication Package” = “Kerberos”, because it is not applicable for Kerberos protocol. This field will also have “0” value if Kerberos was negotiated using Negotiate authentication package.

## COM Objects and .NET Framework

### COM Objects

COM (Component Object Model) is a platform-independent, distributed, object-oriented system for creating binary software components that can interact with each other.

Windows’ API always used both the language and structure of the C programming language. The C language was ideal due to its access to low-level system resources as well as its widespread adoption among software developers at the time. However, as Windows grew in size and complexity, it became difficult to manage everything in a C-style language. To offset the complications, Microsoft released the Component Object Model (COM).COM provided a versatility that was compatible with numerous programming languages as a code wrapper. Code wrappers can reduce complexity of code without sacrificing the utility of the codebase within them. Users could write C, C++, Visual Basic, and other forms of compiled code that could run in Windows. COM also made interprocess communication more streamlined for the average Windows user’s productivity applications.

For example, embedded Excel spreadsheets in Word documents could be updated from Excel, and those changes would be reflected in the embedded object in the Word document. This interconnectivity of programs is a common user experience of interprocess communication. As Windows-based computers became networked, the COM was upgraded resulting in the Distributed

Component Object Model (DCOM). This addressed new issues between COM objects including memory issues, and formatting issues when data is passed between objects running on two different networked machines. In simple terms, DCOM allows us to open a Word document on a network share as if it were stored on our machine. DCOM also enables us to do the same with executables stored on a file server.

To see com objects via powershell:

Get-CimInstance Win32\_DCOMApplication Abusing DCOM for Lateral Movement:

1. $a =

[System.Activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application.1", "10.0.0.2"))

2. $a.Document.ActiveView.ExecuteShellCommand("cmd",$null,"/c hostname > c:\fromdcom.txt","7")

### .NET Framework

Microsoft began a significant development platform modernization in the early 2000s with the .NET Framework.

.NET introduced the new programming languages C#61 and Visual Basic.NET,62 which provide wrappers for the Windows API as well as COM objects within the operating system. The framework also incorporated secure coding mechanisms, and can differentiate between locally-developed code and Internet-downloaded code. It can also determine whether code execution requires elevated privileges and verify if these privileges have been granted to the application.

These features are a part of the Common Language Runtime (CLR). The CLR is a virtual machine64 incorporated within Windows that executes code in a different way than traditionally-compiled languages. Code in C# or Visual Basic.NET is translated into bytecode, 65 or instructions that can be understood by the CLR. The bytecode inside the CLR is inspected and compiled as needed within the virtual machine, rather than compiled all at once. Like Java and other compiled languages that make use of a virtual machine, this allows easier and better protected memory management for applications that are running. Applications are restricted to memory that has been only allocated for them and are unable to read from memory of other processes, including privileged processes. This also means that our user- level software does not have to be rewritten for different operating systems that support the .NET framework. In an enterprise environment with different endpoint devices for end users, any endpoint can run a .NET application without the need for a rebuilt application or a new codebase. The only requirement is that the endpoint support the CLR.

In 2016, a free and open-source successor of the .NET Framework was released to the public. Known as

.NET Core,66 this evolution of the framework makes .NET available to other operating systems in the marketplace. Applications written in C# and other supported languages can be compiled and executed in Linux as well as macOS without the use of a compatibility layer. The .NET Core was updated extensively, growing from version 1.0 to 3.1 in 2019 alone. With the release of .NET 5 (without the Core designation), the original .NET Framework and .NET Core have been combined and is now known simply as .NET without any other nomenclature

## Windows Powershell

### Introduction Architecture

*Features and use cases Important Commands*

|  |  |
| --- | --- |
| 4.3.Get-Help | 4.4.Get-Help Get-Service  **Get-Help Get-\***  **Get-Help Get-Service -examples** |
| 4.5.Get-Process | 4.6.Get-Process notepad.exe  Get-Process -Id 46205  Get-Process notepad.exe -IncludeUsername (returns the owner of process) Get-Process notepad.exe -FileVersionInfo  Get-Process powershell -module |
| 4.7.Format- Table | 4.8.Get-Date | format-table  Get-Date | format-table -Property Month, Day, Hour |
| 4.9.Format-List | 4.10. Get-Date | Format-List Get-Process | Format-List \* (returns all details) |
| 4.11. Get-  Service | 4.12. Get-Service “wmi\*”  Get-Service -Displayname "\*network\*"  “WMI\*” | Get-Service |
| 4.13. Where- Object | 4.14. |
| 4.15. Get-  WinEvent | 4.16. |
| 4.17. Get-  CimInstance | 4.18. |
| 4.19. Invoke- WebRequest | 4.20. |
| 4.21. Get-  WmiObject | 4.22. |
| 4.23. Import- Module | 4.24. |

## 4.25.

* 1. Windows Registry

Windows maintains service and application configurations in the Windows Registry. The Windows Registry is a hierarchical database that stores critical information for the operating system and for applications that choose to use it. The registry stores settings, options, and other miscellaneous information in a hierarchical tree structure of hives, keys, and values.

Keys are the defining data structure for the registry. A key can contain single values related to a necessary

configuration or additional keys possibly containing more values or keys. The values are comprised of three fields: name, type, and data. The name is a meaningful description of the value, such as Load\_On\_Startup, which indicates that an application is to be started when the system boots. The type represents the format of the value such as REG\_SZ for a string or REG\_DWORD for an unsigned integer. The data is the actual value conforming to the type.

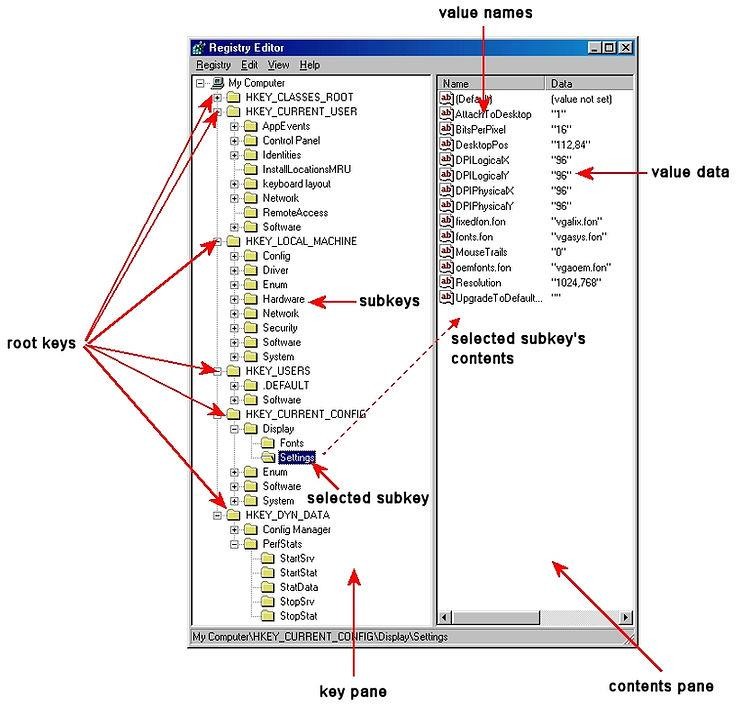
### Hives

Inside the registry, there are root folders. These root folders are referred to as hives. There are five (5) registry hives.

* HKEY\_USERS: contains all the loaded user profiles
* HKEYCURRENT\_USER: profile of the currently logged-on user
* HKEYCLASSES\_ROOT: configuration information on the application used to open files
* HKEYCURRENT\_CONFIG: hardware profile of the system at startup
* HKEYLOCAL\_MACHINE: configuration information including hardware and software settings

### Registry Structure

The registry is structured very similarly to the Windows directory/subdirectory structure. You have the five root keys or hives and then subkeys. In some cases, you have sub-subkeys. These subkeys then have descriptions and values that are displayed in the contents pane. Very often, the values are simply 0 or 1, meaning on or off, but also can contain more complex information usually displayed in hexadecimal.



### Information in the Registry with Forensic Value

As a forensic investigator, the registry can prove to be a treasure trove of information on who, what, where, and when something took place on a system that can directly link the perpetrator to the actions being called into question.

Information that can be found in the registry includes:

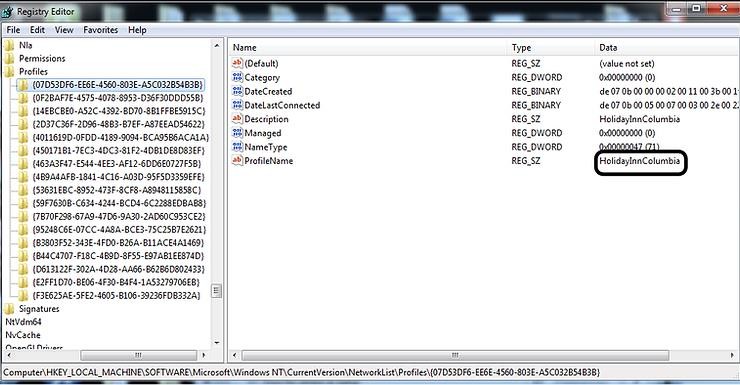
* Users and the time they last used the system
* Most recently used software
* Any devices mounted to the system including unique identifiers of flash drives, hard drives, phones, tablets, etc.
* When the system connected to a specific wireless access point
* What and when files were accessed
* A list any searches done on the system
* And much, much more Wireless Evidence in the Registry

Many hackers crack a local wireless access point and use it for their intrusions. In this way, if the IP address is traced, it will lead back to the neighbor's or other wireless AP and not them.

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles

There, you will find a list of GUIDs of wireless access points the machine has been connected to. When you click on one, it reveals information including the SSID name and the date last connected in hexadecimal. So, although Mr. Borrell initially denied his involvement with this hack, this evidence was conclusive and he eventually plead guilty.

You can see in this screenshot below showing the perpetrator had connected to the "HolidayInnColumbia" SSID in November 2014.



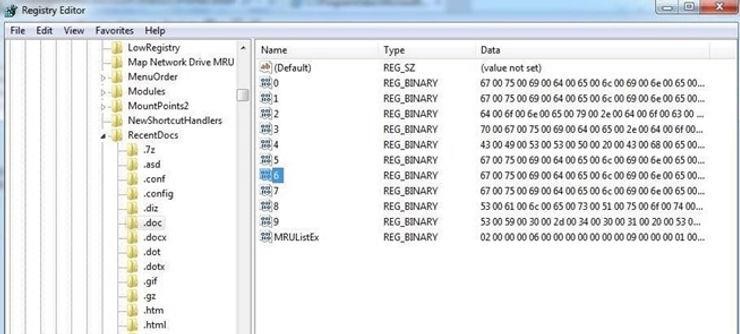
The RecentDocs Key

The Windows registry tracks so much information about the user's activities. In most cases, these registry keys are designed to make Windows run more efficiently and smoothly. As a forensic investigator, these keys are like a road map of the activities of the user or attacker.

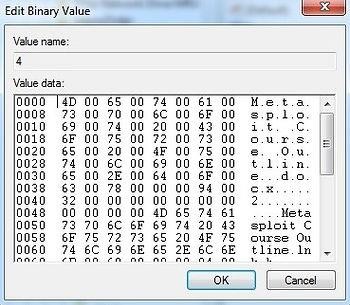
One of those keys is the "RecentDocs" key. It tracks the most recent documents used or opened on the system by file extension. It can be found at:

HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs

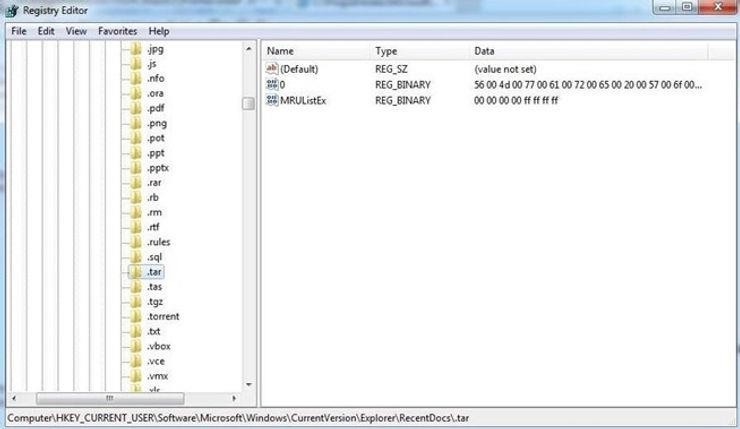
So, for instance, the most recently used Word documents would be found under .doc or the .docx extension depending upon the version of Word they were created in (each key can hold up to the last 10 documents). If we go to the .docx extension, we see the last 10 Word documents listed under this key.



When we click on one of those keys, it reveals information about the document as seen below. We can view the document data in both hex, to the left, and ASCII, to the right. In this case, it show that this document was a Metasploit course outline.



In some cases, an attacker will upload a .tar file, so that is a good place to look for breach evidence. In general, you won't see a .tar file extension on a Windows machine, so the presence of an entry here would be something that needs further investigation. Check the files in the .tar key and see what they might reveal about the attack or attacker.

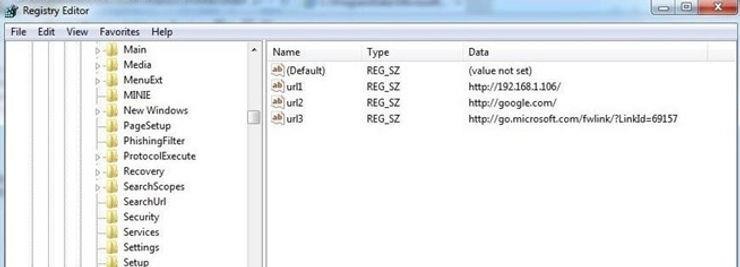


In civil or policy violation investigations, evidence might be found in the various graphic file extensions such as .jpg, .gif, or .png.

TypedURLs Key

When the user types a URL in Internet Explorer, this value is stored in the registry at: HKEY\_CURRENT\_USER\Software\Microsoft\Internet Explorer\TypedURLs

When we open that key in the registry, it lists the last URLs that the user visited with IE. This could reveal the source of malicious malware that was used in the breach, or in civil or policy violation types of investigations, may reveal what the user was looking for/at.



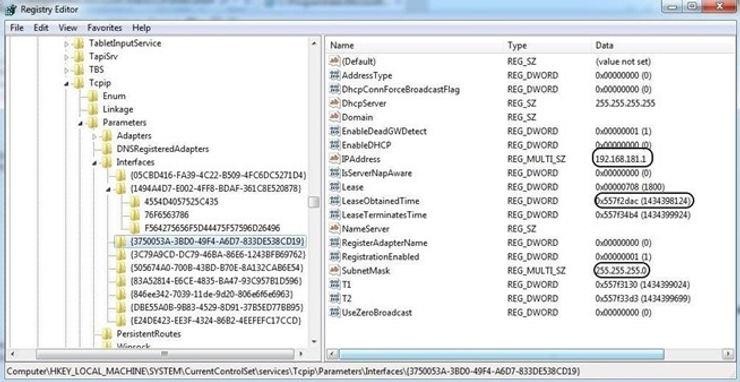
The values will run from urI1 (the most recent) to urI25 (the oldest).

IP Addresses

The registry also tracks the IP addresses of the user interfaces. Note that there may be numerous interfaces and this registry key tracks each interface's IP address and related information.

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\

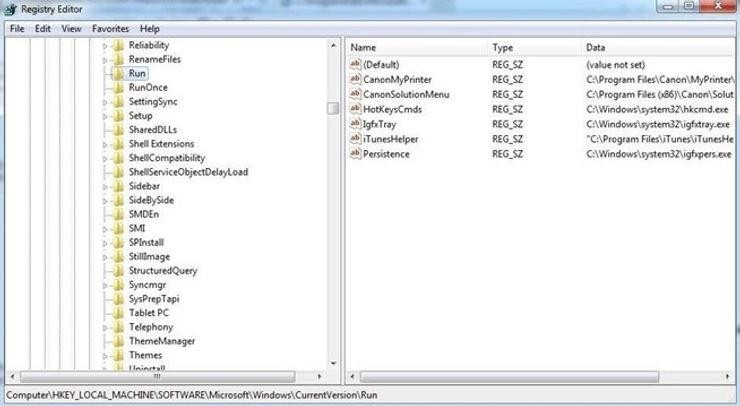
As we can see below, we can find the IP address assigned to the interface, the subnet mask, and the time when the DHCP server leased the IP. In this way, we can tell whether the suspect was using that particular IP at the time of the intrusion or crime.



Start Up Locations in the Registry

As a forensic investigator, we often need to find what applications or services were set to start when the system starts. Malware is often set to start each time the system restarts to keep the attacker connected. This information can be located in the registry in literally tens of locations. We will look at a just a few of the most commonly set keys.

Probably the most used location is: HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run



Any software/locations designated in these subkeys will start every time the system starts. Rootkits and other malicious software can often be found here and they will start each time the system starts.

RunOnce Startup

If the hacker just wanted the software to run once at start up, the subkey may be set here. HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce

Start Up Services

The key below lists all the services that set to start at system startup. If the key is set to 2, the service starts automatically; if it is set to 3, the service must be started manually; and if the key is set to 4, the service is disabled.

HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services Start Legacy Applications

When legacy 16-bit applications are run, the program listed is run at: HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\WOW Start When a Particular User Logs On

In the following key, the values are run when the specific user logs in. HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run Storage Artifacts in the Registry

Often, the suspect will use a Flash drive or hard drive for their malicious activities and then remove them so as not to leave any evidence. The skilled forensic investigator, though, can still find traces of evidence of those storage devices within the registry, if they know where to look.

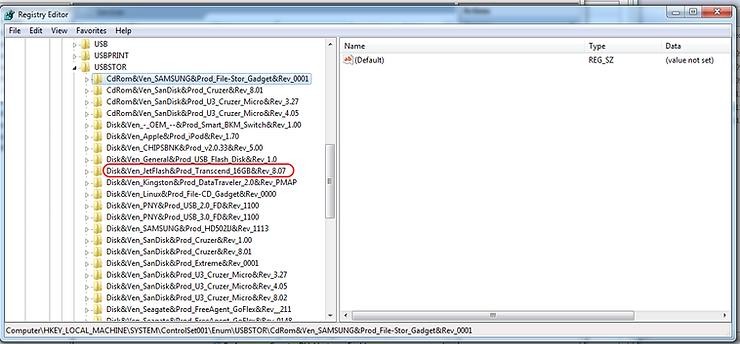
The registry on a Windows system varies a bit from version to version. A skilled, professional digital forensic investigator needs to be able to work with nearly all versions of Windows and other operating systems. Since Windows 7 is still the most widely used operating system, by far, I will be demonstrating on it. Keep in mind, though, that this will vary slightly between versions.

USB Storage Devices

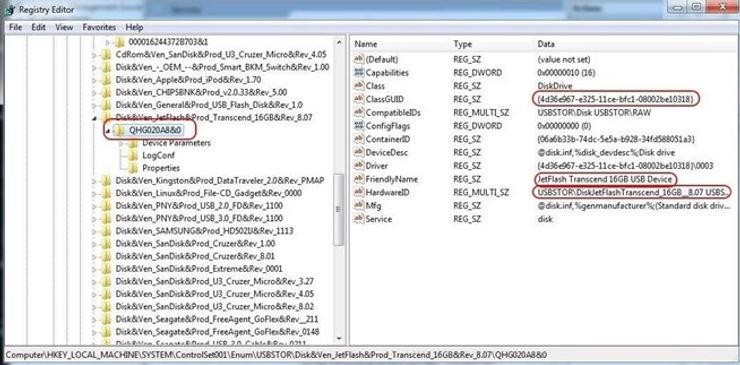
Imagine a case where we suspect that someone installed a keylogger or removed confidential information with a USB drive. How would we find evidence that a USB storage device was inserted and used? To find evidence of USB storage devices, we want to look at the following key.

HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Enum\USBSTOR\

In this key, we will find evidence of any USB storage device that has ever been connected to this system. Expand USBSTOR to see a listing of every USB storage device ever connected to this system.



In the screenshot above, I have circled one suspicious looking USB device. When we expand it, it reveals a unique identifier for that device. By clicking on this identifier, we can find much more information about the device.



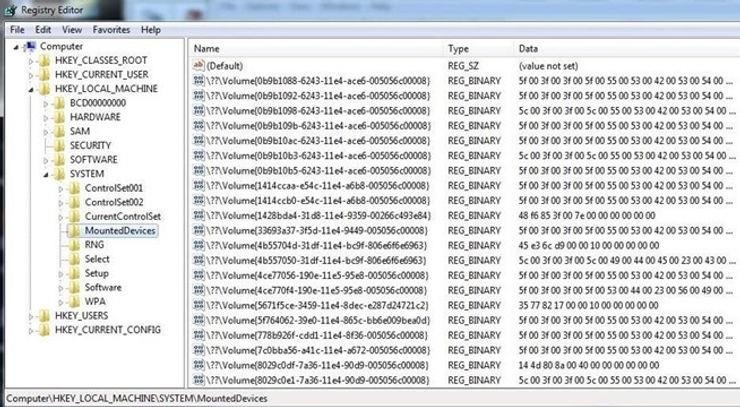
As you can see in the screenshot above, when we click on the USB storage identifier, it reveals in the right- hand window the Global Unique Identifier (GUID), the friendly name, and the hardware ID, among other things. This may be exactly the evidence we need to tie the suspect to their activity on this system!

Mounted Devices

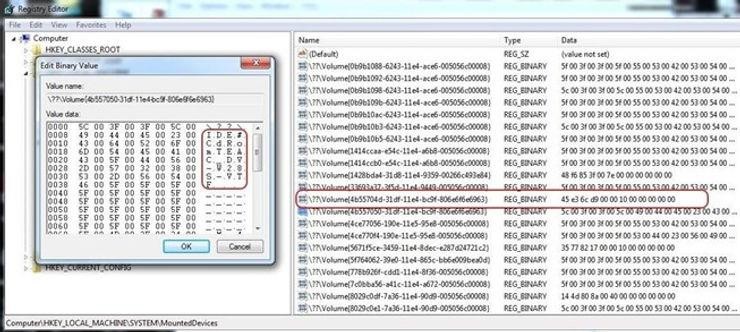
If the suspect used any hardware device that must be mounted to either read or write data (CD-ROM, DVD, hard drive, flash drive, etc.), the registry will record the mounted device. This information is stored at:

HKEY\_LOCAL\_MACHINE\System\MountedDevices

As you can see below, when we click on this key, it provides us a long list of every device ever mounted on that machine.



If we need further information on any of those mounted devices, we can simply click on it, and it will open a small app that will enable us to read the data in ASCII. As you can see, this device was an IDE CD-ROM manufactured by Teac.



If there is not a TEAC CD\_ROM on the system, the forensic investigator now knows that they need to find this piece of hardware to find further evidence of the crime.

The registry is a depository of volumes of information on what happened on a Windows system, and by learning our way around it, we can reconstruct the elements of a crime that it was used for.

## Windows Defender

1. Windows Auditing and Logging

Enable the most important audit policies

Ref: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best- practices/audit-policy-recommendations

Account Logon

*“Local Security Policy” >> Security Settings >> Advanced Audit Policy Configuration >> Account Logon*

*Audit Credential Validation: Success and Failure*

Account Management

*“Local Security Policy” >> Security Settings >> Advanced Audit Policy Configuration >> Account*

*Management*

*Audit Computer Account Management: Success*

*Audit Other Account Management Events: Success and Failure*

*Audit Security Group Management: Success and Failure*

*Audit User Account Management: Success and Failure*

Detailed Tracking

*“Local Security Policy” >> Security Settings >> Advanced Audit Policy Configuration >> Detailed Tracking*

*Audit Plug and Play Events: Success*

*Audit Process Creation: Success*

DS Access

*“Local Security Policy” >> Security Settings >> Advanced Audit Policy Configuration >> DS Access*

*Audit Directory Service Access: Success and Failure*

*Audit Directory Service Changes: Success and Failure*

Logon/Logoff

*“Local Security Policy” >> Security Settings >> Advanced Audit Policy Configuration >> Logon/Logoff*

*Audit Account Lockout: Success and Failure Audit Group Membership: Success Audit Logoff: Success Audit Logon: Success and Failure Audit Special Logon: Success*

Object Access

*“Local Security Policy” >> Security Settings >> Advanced Audit Policy Configuration >> Object Access*

*Audit Removable Storage: Success and Failure*

Policy Change

*“Local Security Policy” >> Security Settings >> Advanced Audit Policy Configuration >> Policy Change*

*Audit Policy Change: Success and Failure*

*Audit Authentication Policy Change: Success*

*Audit Authorization Policy Change: Success*

Privilege Use

*“Local Security Policy” >> Security Settings >> Advanced Audit Policy Configuration >> Privilege Use*

*Audit Sensitive Privilege Use: Success and Failure*

System

*“Local Security Policy” >> Security Settings >> Advanced Audit Policy Configuration >> System*

*Audit IPsec Driver: Success and Failure*

*Audit Other System Events: Success and Failure*

*Audit Security State Change: Success*

*Audit Security System Extension: Success and Failure*

*Audit System Integrity: Success and Failure*

Process Command Line

*“Administrative Template” >> System >> Audit Process Creation*

## Linux Auditing and Logging

* Config firewall (ufw)

*Ufw status*

*nano /etc/default/ufw*

*##Change default policies (if you want)*

* OR, enter the following commands

*ufw default allow incoming*

*ufw default allow outgoing*

*ufw default allow forward*

* To open specific port:

*ufw allow 4422*

* OR

*ufw allow 4422/tcp*

* To deny specific IP:

*ufw deny from 192.168.100.20*

* Enable ufw

*Ufw enable*

*Ufw status*

*Ufw status verbose*

* To list user added rules:

*Ufw show added*

* To enable ufw logging:

*Ufw logging medium*

* Path to log file

*Cat /var/log/ufw\**

Linux Auditd

auditd or Linux Audit Daemon is a user-space component of the Linux Auditing System, responsible for collecting and writing audit log file records to the disk. It is, however, not responsible for viewing the logs, which can be done through **ausearch** or **aureport** utilities

2 types of rules that we’re interested in configuring

* File watches — can watch read, write, execute or attribute changes
* Syscalls — record syscalls sent to the kernel by the application The most important linux audit points:
* Self-auditing (audit the auditing system)
* Kernel parameters (sysctl)
* Kernel module loading and unloading (modprob,insmod,rmmod)
* Mount operations (only attributable)
* Change swap
* Time
* Cron configuration & scheduled jobs
* User, group, password databases
* Sudoer file change
* change group identifiers
* Login configuration and information
* Changes to hostname
* Remote Shell Use
* IP configuration
* System startup scripts
* Power state
* Process ID change (switch account)
* , …

Audit rule writing

* -w : add watch file
* -p: action on file (w,r,a,x)
* -k add lable to identify related logs
* -S: the name of systemcall to audit
* -a: identify systemcall status and taks
* -F: …

Install and config auditd

*Apt install auditd -y*

*> /etc/audit/audit.rules*

*Nano /etc/audit/audit.rules*

Add the audit.rules from Florian Roth (need to customize)

Ref: https://github.com/Neo23x0/auditd/blob/master/audit.rules

*# Buffer Size*

*## Feel free to increase this if the machine panic's*

*-b 8192*

*# Failure Mode*

*## Possible values: 0 (silent), 1 (printk, print a failure message), 2 (panic, halt the system)*

*-f 1*

*# Ignore errors*

*## e.g. caused by users or files not found in the local environment*

*-i*

*# Self Auditing*

*## Audit the audit logs*

*### Successful and unsuccessful attempts to read information from the audit records*

*-w /var/log/audit/ -p wra -k auditlog*

*-w /var/audit/ -p wra -k auditlog*

*## Auditd configuration*

*### Modifications to audit configuration that occur while the audit collection functions are operating*

*-w /etc/audit/ -p wa -k auditconfig*

*-w /etc/libaudit.conf -p wa -k auditconfig*

*-w /etc/audisp/ -p wa -k audispconfig*

*## Monitor for use of audit management tools*

*-w /sbin/auditctl -p x -k audittools*

*-w /sbin/auditd -p x -k audittools*

*-w /usr/sbin/auditd -p x -k audittools*

*-w /usr/sbin/augenrules -p x -k audittools*

*## Access to all audit trails*

*-a always,exit -F path=/usr/sbin/ausearch -F perm=x -k audittools*

*-a always,exit -F path=/usr/sbin/aureport -F perm=x -k audittools*

*-a always,exit -F path=/usr/sbin/aulast -F perm=x -k audittools*

*-a always,exit -F path=/usr/sbin/aulastlogin -F perm=x -k audittools*

*-a always,exit -F path=/usr/sbin/auvirt -F perm=x -k audittools*

*## More information on how to filter events*

*### https://access.redhat.com/solutions/2482221*

*# Rules*

*## Kernel parameters*

*-w /etc/sysctl.conf -p wa -k sysctl*

*-w /etc/sysctl.d -p wa -k sysctl*

*## Kernel module loading and unloading*

*-a always,exit -F perm=x -F auid!=-1 -F path=/sbin/insmod -k modules*

*-a always,exit -F perm=x -F auid!=-1 -F path=/sbin/modprobe -k modules*

*-a always,exit -F perm=x -F auid!=-1 -F path=/sbin/rmmod -k modules*

*-a always,exit -F arch=b64 -S finit\_module -S init\_module -S delete\_module -F auid!=-1 -k modules*

*## Modprobe configuration*

*-w /etc/modprobe.conf -p wa -k modprobe*

*-w /etc/modprobe.d -p wa -k modprobe*

*## KExec usage (all actions)*

*-a always,exit -F arch=b64 -S kexec\_load -k KEXEC*

*## Special files*

*-a always,exit -F arch=b64 -S mknod -S mknodat -k specialfiles*

*## Mount operations (only attributable)*

*-a always,exit -F arch=b64 -S mount -S umount2 -F auid!=-1 -k mount*

*### NFS mount*

*-a always,exit -F path=/sbin/mount.nfs -F perm=x -F auid>=500 -F auid!=4294967295 -k T1078\_Valid\_Accounts*

*-a always,exit -F path=/usr/sbin/mount.nfs -F perm=x -F auid>=500 -F auid!=4294967295 -k T1078\_Valid\_Accounts*

*## Change swap (only attributable)*

*-a always,exit -F arch=b64 -S swapon -S swapoff -F auid!=-1 -k swap*

*## Time*

*-a always,exit -F arch=b64 -F uid!=ntp -S adjtimex -S settimeofday -S clock\_settime -k time*

*### Local time zone*

*-w /etc/localtime -p wa -k localtime*

*## Stunnel*

*-w /usr/sbin/stunnel -p x -k stunnel*

*-w /usr/bin/stunnel -p x -k stunnel*

*## Cron configuration & scheduled jobs*

*-w /etc/cron.allow -p wa -k cron*

*-w /etc/cron.deny -p wa -k cron*

*-w /etc/cron.d/ -p wa -k cron*

*-w /etc/cron.daily/ -p wa -k cron*

*-w /etc/cron.hourly/ -p wa -k cron*

*-w /etc/cron.monthly/ -p wa -k cron*

*-w /etc/cron.weekly/ -p wa -k cron*

*-w /etc/crontab -p wa -k cron*

*-w /var/spool/cron/ -p wa -k cron*

*## User, group, password databases*

*-w /etc/group -p wa -k etcgroup*

*-w /etc/passwd -p wa -k etcpasswd*

*-w /etc/gshadow -k etcgroup*

*-w /etc/shadow -k etcpasswd*

*-w /etc/security/opasswd -k opasswd*

*## Sudoers file changes*

*-w /etc/sudoers -p wa -k actions*

*-w /etc/sudoers.d/ -p wa -k actions*

*## Passwd*

*-w /usr/bin/passwd -p x -k passwd\_modification*

*## Tools to change group identifiers*

*-w /usr/sbin/groupadd -p x -k group\_modification*

*-w /usr/sbin/groupmod -p x -k group\_modification*

*-w /usr/sbin/addgroup -p x -k group\_modification*

*-w /usr/sbin/useradd -p x -k user\_modification*

*-w /usr/sbin/userdel -p x -k user\_modification*

*-w /usr/sbin/usermod -p x -k user\_modification*

*-w /usr/sbin/adduser -p x -k user\_modification*

*## Login configuration and information*

*-w /etc/login.defs -p wa -k login*

*-w /etc/securetty -p wa -k login*

*-w /var/log/faillog -p wa -k login*

*-w /var/log/lastlog -p wa -k login*

*-w /var/log/tallylog -p wa -k login*

*## Network Environment*

*### Changes to hostname*

*-a always,exit -F arch=b64 -S sethostname -S setdomainname -k network\_modifications*

*### Detect Remote Shell Use*

*-a always,exit -F arch=b64 -F exe=/bin/bash -F success=1 -S connect -k "remote\_shell"*

*-a always,exit -F arch=b64 -F exe=/usr/bin/bash -F success=1 -S connect -k "remote\_shell"*

*### Successful IPv4 Connections*

*-a always,exit -F arch=b64 -S connect -F a2=16 -F success=1 -F key=network\_connect\_4*

*### Changes to other files*

*-w /etc/hosts -p wa -k network\_modifications*

*-w /etc/sysconfig/network -p wa -k network\_modifications*

*-w /etc/sysconfig/network-scripts -p w -k network\_modifications*

*-w /etc/network/ -p wa -k network*

*-a always,exit -F dir=/etc/NetworkManager/ -F perm=wa -k network\_modifications*

*## System startup scripts*

*-w /etc/inittab -p wa -k init*

*-w /etc/init.d/ -p wa -k init*

*-w /etc/init/ -p wa -k init*

*## Library search paths*

*-w /etc/ld.so.conf -p wa -k libpath*

*-w /etc/ld.so.conf.d -p wa -k libpath*

*## Systemwide library preloads (LD\_PRELOAD)*

*-w /etc/ld.so.preload -p wa -k systemwide\_preloads*

*## SELinux events that modify the system's Mandatory Access Controls (MAC)*

*-w /etc/selinux/ -p wa -k mac\_policy*

*## Critical elements access failures*

*-a always,exit -F arch=b64 -S open -F dir=/etc -F success=0 -k unauthedfileaccess*

*-a always,exit -F arch=b64 -S open -F dir=/bin -F success=0 -k unauthedfileaccess*

*-a always,exit -F arch=b64 -S open -F dir=/sbin -F success=0 -k unauthedfileaccess*

*-a always,exit -F arch=b64 -S open -F dir=/usr/bin -F success=0 -k unauthedfileaccess*

*-a always,exit -F arch=b64 -S open -F dir=/usr/sbin -F success=0 -k unauthedfileaccess*

*-a always,exit -F arch=b64 -S open -F dir=/var -F success=0 -k unauthedfileaccess*

*-a always,exit -F arch=b64 -S open -F dir=/home -F success=0 -k unauthedfileaccess*

*-a always,exit -F arch=b64 -S open -F dir=/srv -F success=0 -k unauthedfileaccess*

*## Process ID change (switching accounts) applications*

*-w /bin/su -p x -k priv\_esc*

*-w /usr/bin/sudo -p x -k priv\_esc*

*## Power state*

*-w /sbin/shutdown -p x -k power*

*-w /sbin/poweroff -p x -k power*

*-w /sbin/reboot -p x -k power*

*-w /sbin/halt -p x -k power*

*## Session initiation information*

*-w /var/run/utmp -p wa -k session*

*-w /var/log/btmp -p wa -k session*

*-w /var/log/wtmp -p wa -k session*

*## Discretionary Access Control (DAC) modifications*

*-a always,exit -F arch=b64 -S chmod -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S chown -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S fchmod -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S fchmodat -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S fchown -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S fchownat -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S fremovexattr -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S fsetxattr -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S lchown -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S lremovexattr -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S lsetxattr -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S removexattr -F auid>=1000 -F auid!=-1 -k perm\_mod*

*-a always,exit -F arch=b64 -S setxattr -F auid>=1000 -F auid!=-1 -k perm\_mod*

*# Special Rules*

*## Reconnaissance*

*-w /usr/bin/whoami -p x -k recon*

*-w /usr/bin/id -p x -k recon*

*-w /bin/hostname -p x -k recon*

*-w /bin/uname -p x -k recon*

*-w /etc/issue -p r -k recon*

*-w /etc/hostname -p r -k recon*

*## Suspicious activity*

*-w /usr/bin/wget -p x -k susp\_activity*

*-w /usr/bin/curl -p x -k susp\_activity*

*-w /usr/bin/base64 -p x -k susp\_activity*

*-w /bin/nc -p x -k susp\_activity*

*-w /bin/netcat -p x -k susp\_activity*

*-w /usr/bin/ncat -p x -k susp\_activity*

*-w /usr/bin/ss -p x -k susp\_activity*

*-w /usr/bin/netstat -p x -k susp\_activity*

*-w /usr/bin/ssh -p x -k susp\_activity*

*-w /usr/bin/scp -p x -k susp\_activity*

*-w /usr/bin/sftp -p x -k susp\_activity*

*-w /usr/bin/ftp -p x -k susp\_activity*

*-w /usr/bin/socat -p x -k susp\_activity*

*-w /usr/bin/wireshark -p x -k susp\_activity*

*-w /usr/bin/tshark -p x -k susp\_activity*

*-w /usr/bin/rawshark -p x -k susp\_activity*

*-w /usr/bin/rdesktop -p x -k susp\_activity*

*-w /usr/local/bin/rdesktop -p x -k susp\_activity*

*-w /usr/bin/wlfreerdp -p x -k susp\_activity*

*-w /usr/bin/xfreerdp -p x -k susp\_activity*

*-w /usr/local/bin/xfreerdp -p x -k susp\_activity*

*-w /usr/bin/nmap -p x -k susp\_activity*

*## T1002 Data Compressed*

*-w /usr/bin/zip -p x -k Data\_Compressed*

*-w /usr/bin/gzip -p x -k Data\_Compressed*

*-w /usr/bin/tar -p x -k Data\_Compressed*

*-w /usr/bin/bzip2 -p x -k Data\_Compressed*

*-w /usr/bin/lzip -p x -k Data\_Compressed*

*-w /usr/local/bin/lzip -p x -k Data\_Compressed*

*-w /usr/bin/lz4 -p x -k Data\_Compressed*

*-w /usr/local/bin/lz4 -p x -k Data\_Compressed*

*-w /usr/bin/lzop -p x -k Data\_Compressed*

*-w /usr/local/bin/lzop -p x -k Data\_Compressed*

*-w /usr/bin/plzip -p x -k Data\_Compressed*

*-w /usr/local/bin/plzip -p x -k Data\_Compressed*

*-w /usr/bin/pbzip2 -p x -k Data\_Compressed*

*-w /usr/local/bin/pbzip2 -p x -k Data\_Compressed*

*-w /usr/bin/lbzip2 -p x -k Data\_Compressed*

*-w /usr/local/bin/lbzip2 -p x -k Data\_Compressed*

*-w /usr/bin/pixz -p x -k Data\_Compressed*

*-w /usr/local/bin/pixz -p x -k Data\_Compressed*

*-w /usr/bin/pigz -p x -k Data\_Compressed*

*-w /usr/local/bin/pigz -p x -k Data\_Compressed*

*-w /usr/bin/unpigz -p x -k Data\_Compressed*

*-w /usr/local/bin/unpigz -p x -k Data\_Compressed*

*-w /usr/bin/zstd -p x -k Data\_Compressed*

*-w /usr/local/bin/zstd -p x -k Data\_Compressed*

*## Added to catch netcat on Ubuntu*

*-w /bin/nc.openbsd -p x -k susp\_activity*

*-w /bin/nc.traditional -p x -k susp\_activity*

*## Sbin suspicious activity*

*-w /sbin/iptables -p x -k sbin\_susp*

*-w /sbin/ip6tables -p x -k sbin\_susp*

*-w /sbin/ifconfig -p x -k sbin\_susp*

*-w /usr/sbin/arptables -p x -k sbin\_susp*

*-w /usr/sbin/ebtables -p x -k sbin\_susp*

*-w /sbin/xtables-nft-multi -p x -k sbin\_susp*

*-w /usr/sbin/nft -p x -k sbin\_susp*

*-w /usr/sbin/tcpdump -p x -k sbin\_susp*

*-w /usr/sbin/traceroute -p x -k sbin\_susp*

*-w /usr/sbin/ufw -p x -k sbin\_susp*

*## Injection*

*### These rules watch for code injection by the ptrace facility.*

*### This could indicate someone trying to do something bad or just debugging*

*-a always,exit -F arch=b64 -S ptrace -F a0=0x4 -k code\_injection*

*-a always,exit -F arch=b64 -S ptrace -F a0=0x5 -k data\_injection*

*-a always,exit -F arch=b64 -S ptrace -F a0=0x6 -k register\_injection*

*-a always,exit -F arch=b64 -S ptrace -k tracing*

*## Anonymous File Creation*

*### These rules watch the use of memfd\_create*

*### "memfd\_create" creates anonymous file and returns a file descriptor to access it*

*### When combined with "fexecve" can be used to stealthily run binaries in memory without touching disk*

*-a always,exit -F arch=b64 -S memfd\_create -F key=anon\_file\_create*

*## Privilege Abuse*

*### The purpose of this rule is to detect when an admin may be abusing power by looking in user's home dir.*

*-a always,exit -F dir=/home -F auid=0 -F auid>=1000 -F auid!=-1 -C auid!=obj\_uid -k power\_abuse*

*# Socket Creations*

*# will catch both IPv4 and IPv6*

*-a always,exit -F arch=b32 -S socket -F a0=2 -k network\_socket\_created*

*-a always,exit -F arch=b64 -S socket -F a0=2 -k network\_socket\_created*

*-a always,exit -F arch=b32 -S socket -F a0=10 -k network\_socket\_created*

*-a always,exit -F arch=b64 -S socket -F a0=10 -k network\_socket\_created*

*# Software Management*

*# DPKG / APT-GET (Debian/Ubuntu)*

*-w /usr/bin/dpkg -p x -k software\_mgmt*

*-w /usr/bin/apt -p x -k software\_mgmt*

*-w /usr/bin/apt-add-repository -p x -k software\_mgmt*

*-w /usr/bin/apt-get -p x -k software\_mgmt*

*-w /usr/bin/aptitude -p x -k software\_mgmt*

*-w /usr/bin/wajig -p x -k software\_mgmt*

*-w /usr/bin/snap -p x -k software\_mgmt*

*# PIP(3) (Python installs)*

*-w /usr/bin/pip -p x -k third\_party\_software\_mgmt*

*-w /usr/local/bin/pip -p x -k third\_party\_software\_mgmt*

*-w /usr/bin/pip3 -p x -k third\_party\_software\_mgmt*

*-w /usr/local/bin/pip3 -p x -k third\_party\_software\_mgmt*

*-w /usr/bin/pipx -p x -k third\_party\_software\_mgmt*

*-w /usr/local/bin/pipx -p x -k third\_party\_software\_mgmt*