SANDIA REPORT
SAND2022-15172
Printed September 2022

Sandia National Laboratories

Equipment Testing Environment (ETE) Process Specification

Andrew Hahn, Benjamin Karch, Robert Bruneau, Michael Rowland

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

ABSTRACT

This document is intended to be utilized with the Equipment Test Environment being developed to provide a standard process by which the ETE can be validated. The ETE is developed with the intent of establishing cyber intrusion, data collection and through automation provide objective goals that provide repeatability. This testing process is being developed to interface with the Technical Area V physical protection system.

The document will overview the testing structure, interfaces, device and network logging and data capture. Additionally, it will cover the testing procedure, criteria and constraints necessary to properly capture data and logs and record them for experimental data capture and analysis.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support received by the International Nuclear Security Engineering department for their support in setting up and coordinating the test environment for this evaluation. Specifically, thanks to Sondra Spence, James Bonilla, Remengton Pierce for their direct support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acronyms and Terms

Acronym/Term	Definition
ADS	Attack Development System
C3	Command and Control Channel
CAS	Central Alarm Station
CLI	Command Line Interface
CNC	Control Network Communicator
CVE	Common Vulnerability and Exposure
DUT	Device Under Test
ECS	Experiment Control System
CLI	Command Line Interface
CMS	Communication Management System
ETE	Equipment Test Environment
IP	Internet Protocol
JSON	Java Script Object Notation
LMS	Log Management System
NCM	Network Configuration Manager
NRS	Network Recording Script
PCAP	Packet Capture file
PPS	Physical Protection System
Protect Li	Firewall device that is configurable for specific applications
RC	Report Collector
SA&BS	Scripts Archive & Backup System
SPAN	Switched Port Analyzer
SIEM	Security Incident and Event Monitor
SO	Shared Object
SSC	System Startup Configurator
SSH	Secure Shell Protocol
SUT	System Under Test
USB Samurai	Remote Controlled USB injection tool
USB Rubber Ducky	USB Drive that consisting of attack payloads
UCA	User Command Agent
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
ZMQ	Zero MQ is an embeddable networking library for capturing traffic

1. PURPOSE

This testing procedure is intended to be utilized with the ETE specification in order to provide consistent initialization of the system for data capture, timing coordination, and provide consistent injection of observations.

2. TESTING STRUCTURE

2.1. Network Recording Structure

- Principle: All network packets to and from the system under test must be captured
- Practice: The network communications of the system under test must be recorded with the network taps and ProtectLi recording system.
- Execution:
    - The network must be recorded at the network adaptor of the system under test and downstream of the nearest network switch. If the nearest downstream switch is not accessible, introduce the unmanaged switch specified in the equipment list above.
    - If the switch has an available SPAN port, it must be routed to the ProtectLi for recording.

2.2. Log Recording Structure

- Principle: All available logs from the system under test should be collected and recorded.
- Practice: If the system under test generates logs they should be recorded. These are broken into two types of logs:
    - The larger group of all logs the system can record. These will be collected by the log collection script at the end of each test.
    - The logs that the system would report on the network to the SIEM. Logstash on the ProtectLi will record these.
- Execution:
    - The ProtectLi will be ready to receive the network logs with the recording script.
    - The Log Setup Script will start log recording and broadcasting on the device under test.
    - The Log Collection Script will collect all the logs from the system under test and deliver them to the Desktop to be collected on the external SSD for data collection.

2.3. Attack Development Structure

- Principle: Attacks that emulate network based and insider-based threats will be developed and executed on the equipment under test. The development, time, and execution of the attack must be recorded.
- Practice: The attacks will be broken into 2 major vectors, network and insider. For both vectors the recording system will be active for all the time from the start of first attempts and development of the attack to the final success or failure of the attack. For every milestone of attack development (i.e., when a function or effect is gained) the time and function or effect will be recorded in the logs on the ProtectLi and on Paper. Time will be reference by the ProtectLi's time clock.
- Execution:
    - Network:

        - The Attacker machine will be connected at the nearest downstream switch.
        - Recording will start just before the Attacker machine is connected to the network.
        - Each function gained or effect achieved must be recorded in the Logs on the ProtectLi and in the Lab Logbook. The Script/command/tool used to gain the function or generate the effect must be recorded with the time in the logbook.
        - Anytime a function gained, or effect achieved has a visual impact on the equipment under test, this must be recorded by camera and stored with the other log files. The filename of the picture must be descriptive of the effect, causation, and time.
        - At the end of test, the Recording Script will be safely ended, the Log collection script will be executed on the device under test if possible, and the attacker machines command history must be recorded to a ".txt" file.
        - All Logs and PCAPs will be transferred to the SSD's appropriate for the OS type.
    - Insider:
        - This will use the USB Samurai and USB Rubber Ducky to attack the equipment under test if the equipment has USB-A ports.
        - The Attacker Machine will be used as a development base for the attacks.
        - Recording will start just before the first time a USB attacking device is connected to the equipment under test.
        - Each function gained or effect achieved must be recorded in the Logs on the ProtectLi and in the Lab Logbook. The Script/command/tool used to gain the function or generate the effect must be recorded with the time in the logbook.
-

3. TESTING BOUNDARIES

The testing boundaries in will be defined by the sponsor, host or by operational necessities of the system under test (SUT). In this instance we are evaluating the Technical Area V PPS system and CAS. Some devices are utilized by security forces to conduct operations and must not be disrupted as a result of our testing. The team has taken care to isolate those and make sure that the team understand the out of bounds devices.

Some limitations include

- CAS server – this cannot be attacked directly wen testing the controller in the field distribution box.
- Access controls that define boundary to limited areas. We configured the network to utilize access controls that are representative of devices being utilized but are not currently active in the necessary access controls to the limited area being monitored by security forces.

MITRE report [1] effects that represent adversary goals include Deny, Disrupt, Deceive, Degrade, Destroy. These adversary goals are represented in our testing as follows

Deny:

- Delay Detection
  - Delay reporting of detection to CAS/Operators
  - Delay sensing observational data delivery to CAS/Operators (i.e., delay camera image transmission)

Disrupt

- Degrade Detection
  - Degrade sensing data integrity delivered to CAS/Operators
- Stop Detection
  - Prevent CAS/Operators from detecting sensor trip
  - Prevent CAS/Operators from observation of intrusion
- Change Access Rules
  - Change access controls to allow unauthorized access to areas, or resources.
  - Change access controls to disallow authorized access to areas, or resources.
- Disable Device
  - Stop device under test from performing its function

Deceive

- Distract Operators
  - False positive detections
  - Device malfunctions that require operator intervention

Degrade

- Compromise camera for video processing by attacking camera settings
- Compromise PTZ camera to reorient the focus area of the camera by re-directing the visual field

Destroy

- We recognize that some of the network attacks and device perturbations can result in a bricked device meaning it no longer performs its intended function however the red team has chosen to not do any testing that will intentionally damage devices for this exercise.

4. TESTING PROCEDURE

4.1. Framework (here's the tools)

- What tools we expect to be created
  - Custom programmed SSH instances with configurations preloaded
  - Elasticsearch configuration
  - Logging scripts / locations defined through log stash
  - Sysmon configuration
  - Packet parsing scripts
  - Kali Linux system configured with tools to be utilized
  - Python
  - Dirbuster
- What is the preparation needed for the system under test
  - First time connecting to testbed environment
  - Packet parsing scripts
  - Elasticsearch setup and configuration
  - Logging through log stash
  - Sysmon configuration
  - Netcat for exploit deployment / test launch
  - Appropriate taps for network connection and monitoring in addition to providing storage capacity for captured network packets
  - How do you refresh the system if you run a test?
  - What should you avoid to not brick anything?
    - On the PPS the team used TMUX for running instances and maintaining consistent operating run-time for specific scripts
    - Capture successful attacks and attack traffic with the LMS recorder
    - Files are being stored locally and stored for forensic analysis for both research and implementation purposes
  - What happens if you need a reset (remote reset)
  - Outlined dependencies of environment (TA-5 dependencies and boundaries)
    - TAV has operational devices that are beyond the scope of the test environment. The network configuration has been established to isolate out of bound devices.
    - What can't be hit that would damage systems outside of system under test?
    - What can't be hit that would damage systems outside of system under test?
  - If needed pause or suspend the test
  - If needed restart the test
- Method (High level description): The approach that we are going to take
  - The test approach will consist of setup per this specification and the ETE process specification. The specific SUT or device under test and expected results will be documented, test data capture results will be stored, analyzed. The test criteria may be altered depending on preliminary analysis and then re-run, with those

        - results also being captured. Data result captures will include number of attempts and any necessary changes to successfully test to know ranges of results.
    - Establish a unique event at the start of the test sequence to sync time stamps between system under test network and establish NTP server sync at the beginning for the NTP and SUT synchronization. This will help establish common time for analysis.

4.2. Initial Evaluation of Device Under Test

    - Record the details of the device under test in the Logbook.
    - Determine the services, processes, and programs that are critical to its function.
    - Search for CVE’s related to the device under test, its services, processes, programs, and operating system
        - If available, determine which CVE’s will be most quickly implemented with the tools at hand.
        - Develop an ordered list of CVE’s that can gain functions or produce effects desired.
- Setup Recording System
    - System must record all traffic from the device under test.
        - All traffic downstream and upstream of the switch where the attacker computer will be plugged in.
    - If applicable, install the log recording and capture tools on the device under test with the Log Reporting Setup Script
    - Verify that the system is recording packets and logs correctly from the system under test.
- Verify Operation of System Under Test
    - Validate that the system under test is functioning as expected in the SUT specification.

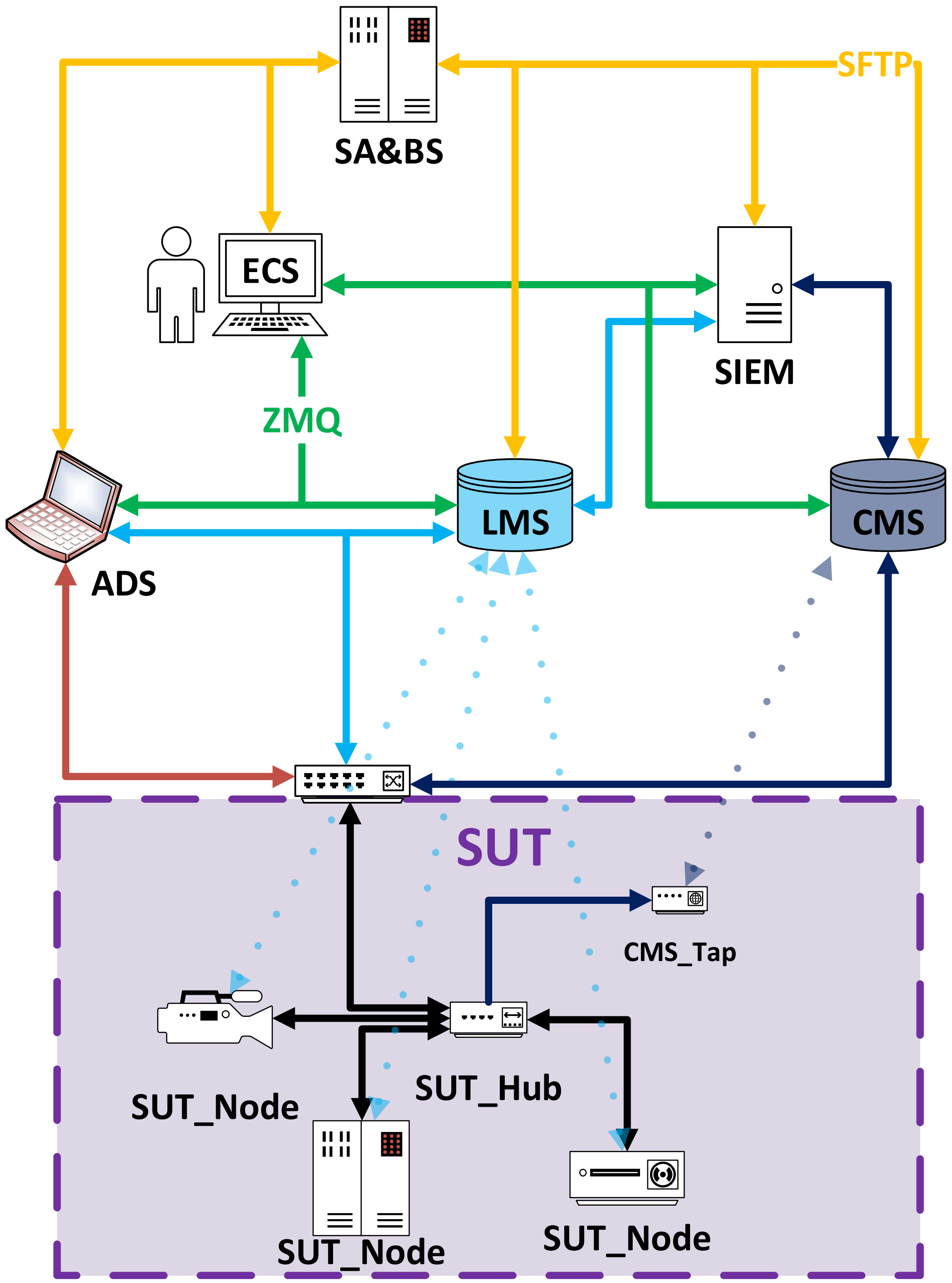

Figure 1. ETE TEST DIAGRAM

5. ETE SCRIPTS AND SOFTWARE

5.1. ECS

The ECS is the control center for the ETE, it must monitor and command the ETE subsystems. To accomplish this the ETE will have a module in each subsystem to accept commands and return feedback. This will be done using a standard JSON formatted communication using the ZMQ communication library. The system will also keep the entire ETE on a single clock time to ensure that all logs, files and PCAPs have a uniform time record.

As the primary interface for users of the ETE, the ECS will also need to provide a human interface. This will provide the users conditions and feedback from the ETE, allow the issue of commands, and provide a note and observation input that will be timestamped in an experiment log.

5.1.1. ECS Responsibilities

- Central Timekeeper
- Issue commands to ETE subsystems
- Gather condition and operational data on ETE subsystems
- Provide user interface
- Record user observations and notes with timestamps

5.1.2. ECS Dependencies

- ECS will be dependent on the CMS network to allow its commands and feedback to be networked and routed to it.

5.2. Command and Control Interface

Below each function described will exist as a thread in the C&CI script. These will share information with each other via a single shared memory object.

- System Startup Configurator (SSC)
    - This will read a JSON formatted startup file with all the initial conditions of the ETE system configuration. This will allow the ETE to know the network addresses of each system and broadcast out UDP to inform the other ETE systems of its address to listen to for commands. It will also include the current system time.
    - Dependencies: JSON reader, UDP, Timekeeper, ZMQ
    - Method:
        - Startup of ECS sets the SSC to begin looking for a JSON.
            - An input argument is script start can redirect to a JSON config that is no in the default location.
            - If no JSON, stop system, print error
        - Read and parse JSON
        - Initialize Shared Object and input IP matched with subsystem

      - Inject special script commands that the User Command Interpreter will use for special command calls (e.g., macro for recording start) into the Shared Object.
      - Inject configs for each sub-system
    - Start ZMQ server for subsystems to report to
      - This server will be handed off to the RC after good start reported.
    - Use UDP to broadcast to ETE subsystems the IP of the ECS and the current time.
      - Broadcast until each sub-system reports back via ZMQ
      - Print when subsystem reports back
    - Report start success to all ECS systems.

5.3. Control Network Communicator (CNC)

- This system handles the communications of the ECS. It writes out JSON commands to the network via ZMQ and unsured their receipt. Every command will include the current time.
- Dependencies: ZMQ, JSON, TCP/IP, Timekeeper, User Command Interpreter
- Method
  - Wait for SSC to report good start.
  - Bind ZMQ command socket to each subsystem
    - Each subsystem will have its own bound socket
  - Query the Shared Object for ETE configurations that need to be sent to out to each sub-system
    - The SSC will have set these by reading the JSON config for the ECS.
  - Send ETE configuration to each sub-system that requires it in JSON formatted message.
    - Confirm configuration receipt and validation of good config from these sub-systems
  - Command loop:
    - The CNC will wait for commands for the subsystems by watching the queues on the Shared Object
      - The SO will have queues for commands that must be issued to each subsystem
    - When a queue has a command the CNC will package that command in the correct JSON format, add a timestamp, and send it to the subsystem

        - Once sent the CNC will listen for a command success return from the subsystem before moving to the next queued command.
    - At the UCA's command the CNC will send a shutdown command to all subsystems to safely shut down the ETE. The CNC will wait to hear successful command receipts before safely shutting down the ECS.

5.4. Timekeeper

- This system keeps the time on the ECS as up to date as possible, locked to an epoch of the ECS and users' choice. It is callable for a time and at intervals of 100ms it will record the time to the Shared Object. It will also be resettable.

NOTE: not all testing will require a separate timekeeper system, this will need to be determined by the test team depending on the SUT. The team will need to assess if the data capture equipment times are adequate for post analysis following completion of the testing.

- Dependencies: System Time
- Method
    - The Timekeeper will start with the ECS and record the system time at start
    - System Current Time – System start time = Current ECS time
    - Every round 100ms record to Shared Object
    - Call for time returns current ECS Time to 1ms resolution
    - Call for reset updates System start time to System current time

5.5. User Command Agent (UCA)

- This is a command window or CLI that allows the user to input commands in a specific format that the UCA can interpret. It will also generate a log of all the commands with timestamps to file.
- Dependencies: Timekeeper, SSC, CNC
- Method
    - The UCA waits for the SSC to send the good start indication
    - The UCA sets up its command dictionary based on the configuration from the SSC stored in the Shared Object
    - Command Loop:
        - Wait for user input string
            - Pressing enter or a new line initiates command interpretation loop
        - Clean up input string (separate into an array, space delimitation, reduce all to lower case)
        - Check first array word against dictionary of acceptable commands

        - No match -> return error, go back to CLI
        - Match -> proceed to next processing step based on the dictionary
    - Command Dictionary:
        - (Full dictionary and arguments will be determined by each subsystem final development)
        - Rec_start – follow SSC config to start up the ETE recording
        - Rec_stop – follow SSC config to stop the ETE recording
        - Time_reset – resets ECS time
        - Shutdown – shuts the ETE down safely
        - Note – Makes a log of whatever comes after the 'Note'
        - LMS_inject – Commands the LMS to inject the LMS agent into the SUT
            - Argument: OS type
        - LMS_scrape – Commands the LMS to gather the logs from the SUT
        - LMS_clear – Commands LMS to clear out logs
        - LMS_stop – Commands LMS to stop the log agent on the SUT
        - LMS_shutdown – safely shuts down the LMS
    - Put properly formatted command in the queue of the CNC for the appropriate system.
    - Record the command to the log file with a timestamp

5.6. Report Collector (RC)

- The report collector gathers the return information from the ETE subsystems and records their messages to a log. The SSC hands off the reporting ZMQ server to the RC to continue monitoring. The Feedback Agent relies on the RC for ETE data reports.
- Dependencies: SSC, ZMQ, Timekeeper
- Method
    - Waits for SSC to hand off report ZMQ server
    - Check if Forwarding agent has a hold on logging to file
    - Loop:
        - Listen to ZMQ server for responses
        - Receive a response

        - Parse into a string for logging, add a time stamp taken from the Timekeeper
        - Log to file and the Feedback queue in the Shared Object
    - Listen for shutdown command to safely close ZMQ server
    - Listen for command to end logging to file for log forwarding agent to pack file.

5.7. Feedback Agent

- The Feedback Agent is responsible for providing feedback to users. It will display messages received via the Report Collector, the statuses of systems as reported by the CNC, and error messages from the ECS. It will be text based at first, but should become somewhat graphical, though still function over SSH.
- Dependencies: CNC, RC, SSC, UCA
- Method
    - Wait for SSC to send good start indication
    - RC Loop:
        - Listen to RC queue for messages
        - Print messages and remove from queue
    - UCA Loop:
        - Listen for UCA request for ETE status
        - Request connection status from CNC
        - Print connection status's for ETE

5.8. Log Forwarding Agent

5.8.1. Responsibilities

- The log forwarding agent collects the logs of the ECS and sends them to the SA&BS for archival. This system will also clear the logs from the ECS at request.

5.8.2. Dependencies: UCA, RC, SSH

- Method:
    - Listen for log forward or shut down request from UCA
    - End logging systems process, safely close log file.
    - Rename file based on ETE test name in Shared Object and the date and time
    - Send to SA&BS and confirm receipt of file
    - Restart RC logging

5.9. CMS

The Communication Management System has the responsibilities of governing and recording the network of the ETE. It first must establish the network and control its routing and connections. It will then control the recording of the SUT and necessary sub-systems. The CMS will need to ensure that it always maintains the connection to the ECS to maintain control.

5.9.1. Responsibilities

- Control VLAN creation and management on ETE
- Control communications between ETE and SUT components
- Record full PCAPs of traffic to and from the SUT
- Manage the size of PCAPs
- Capture network logs from the captured traffic
- Forward logs and PCAPs to the SA&BS

5.9.2. Dependencies

The CMS will be dependent on the ECS for commands, and it will have hardware dependencies from the network taps and the network switch it will control. The taps and switch also imply a dependency on the software used to control these devices.

5.10. Command and Control Channel (C3)

The C3 will be the interface between the ECS and the CMS. It is a module standardized and shared across the ETE subsystems. It will listen to the ECS for UDP and ZMQ to configure and control the other systems in the CMS. It will use a configuration library for its specific control needs over the CMS.

5.10.1.1. Responsibilities

- Set an initial starting config for the CMS network
- Listen for the ECS UDP
- Connect to the ECS ZMQ channels
- Send commands to CMS systems
- Report back to ECS

5.10.1.2. Function

- Setup System
  - This will start up the other CMS Systems and put the network in a condition to allow it to listen for the ECS UDP. The Setup System will then configure the connection to the ECS via ZMQ.
  - Dependencies: ZMQ, UDP, NCM
  - Method:

        - On start up the system will initialize the NCM and NRS
        - Wait for NCM start and send command to configure network on one VLAN to listen for UDP.
            - Commands to NCM and NRS are transmitted through pipes or shared memory.
        - Wait for UDP transmission
        - Configure ZMQ to ECS
            - Both communication directions will be established.
        - Send "System Ready" signal to ECS
        - Wait for receipt of configuration JSON from ECS
        - Validate the JSON config and report successful receipt to ECS
        - Set system time based on ECS time
        - Parse configuration file and send to NCM and NRS
        - Pass ZMQ to C3 Client

5.10.2. C3 Client

- The C3 Client handles the communications to and from the ECS after the setup system is finished.
- Dependencies: ZMQ, NCM, NRS, ECS
- Method
    - Wait for setup to pass the ZMQ connection
    - Loop:
        - Listen for ZMQ command from ECS
        - Parse command and forward to appropriate CMS systems
        - Report back messages from subs systems to ECS

5.10.3. Network Configuration Manager (NCM)

The NCM controls the network switch that is at the core of the CMS. It must find the switch and begin the control pathway, which is dependent on the manufacture of switch used, then maintain control of the switch for commands from the C3.

5.10.3.1. Responsibilities

- Connect to switch control interface
- Manage connections and VLANs
- Report status to C3

5.10.3.2. Function

- Switch Setup
  - This finds and sets up the control pathway of the network switch.
  - Dependencies: TBD
  - Method
    - Scan ethernet ports for switch
    - Start control pathway
- Switch Control
  - After the control pathway is established, this script is called by the C3 to perform actions on the switch configuration. This is an interpreter for C3 configuration commands.
  - Dependencies: TDB, C3
  - Method
    - When called assert request
    - Process arguments into valid form for switch
    - Send command to switch
    - Ensure command is processed
    - Report command return from switch to C3

5.10.4. Network Recording Script (NRS)

This script is responsible for starting up the network-based recordings for the test. This includes starting PCAPs on all the network interfaces and Logstash to record to file all the received logs from the system under test.

5.10.4.1. Responsibilities

- Record network traffic on selected ports.
- Manage PCAP size and duration
- Capture network logs from SUT
- Organize, name, pack and send logs and PCAPs to SA&BS

5.10.4.2. Function

Detect all connected ethernet ports. Start TCPDump PCAP captures on each active interface. (Make these separate captures).

PCAPs need a naming scheme to make them distinct. Ethernet ports should form the prefix and a user defined string should form the remainder of the file names.

The script will have to maintain the PCAP recording over a long period of time and therefore will need to ensure that individual PCAP files do not become too large. The script should at intervals stop the PCAP and start and new PCAP with a new file. The interval should be 1 hour. The best solution would be a file size monitoring solution that measures the size of the PCAP file and above a file size cap with stop and restart the PCAP to a new file.

Start Logstash with a user defined IP address. Logstash should be receiving logs and recording them to a file. The name scheme of this file should follow similar rules as the PCAPs. The file format should be JSON.

The script should have a way for the user to end the recording safely. This would shutdown the PCAP recordings and Logstash. It should then collect the files in a zip folder that is deposited on the Desktop. The recording files should then be cleaned up.

5.11. LMS

The LMS receives control commands from the ECS on the ZMQ channel concerning log collection of individual devices / nodes in the SUT. For now, support exists only for nodes running Windows. The command received by the LMS from the ECS follows the standard ECS command format. An example of this is shown below:

```
{
“command” : “LMS_configure_target”
“parameters” : {
        “target_ip” : “10.0.0.26”
        “ssh_key” : “<ssh_key>”
        “username” : “admin”
        “log_type” : “Sysmon”
        }
}
```

5.11.1. Responsibilities

- Set up logging and log forwarding on individual machines in SUT
- Start and stop logging and log forwarding services on SUT machines
- Store logs received during experimentation
- Forward logs to SIEM
- Destroy all logs on SUT machines

5.11.2. Dependencies

It is also assumed that those nodes are running the OpenSSH service and that the LMS contains an SSH key, to allow for remote control of the node. Instructions for generating SSH keys and enabling the SSH service can be found at [https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_keymanagement].

Winlogbeat is used on the nodes in the SUT to collect and forward Sysmon logs to the LMS. The LMS runs an instance of logstash, which enables the LMS to receive those forwarded logs over UDP. Commands that the LMS must handle relate to its own logstash service starting and stopping as well as setting up, starting, and stopping logging and log forwarding services used on all applicable nodes in the SUT.

5.11.3. *Functions*

5.11.3.1. LMS_listen

This command indicates that the LMS should begin logstash to listen for logs reported from devices in the SUT. This is accomplished by running the "systemctl start logstash" command. Configuration of logstash is necessary only once, so there are no other commands necessary.

5.11.3.2. LMS_stop

This indicates that the LMS should shut down logstash and close the associated port. The LMS accomplishes this by issuing the "systemctl stop logstash" command.

5.11.3.3. LMS_configure_target

This indicates that a new node has been added to the SUT, where logging is not yet configured. The node must be loaded with the necessary tools, Sysmon and winlogbeat. These tools must also be configured to report logs to the proper endpoint (the LMS). This is accomplished by executing typical commands, but from a remote session and autonomously by the LMS. Pseudocode for this process is shown below:

```
scp /logtools/sysmon.exe target_user@target_ip:C:/logtools/sysmon.exe

scp /logtools/winlogbeat/* target_user@target_ip:C:/logtools/winlogbeat/ #includes scripts,
configuration file, executable, etc


ssh target_user@target_ip "
        C:/logtools/sysmon.exe -i -accepteula -h md5,sha256,imphash -l -n;
        powershell -ExecutionPolicy Bypass;
        powershell ./install-service-winlogbeat.ps1;
"
```

The configuration file included with winlogbeat files will have the following line edited to ensure logs are reported to the LMS:

```
output.logstash:
        hosts: ["LMS IP address"]
```

5.11.3.4. LMS_start_target

This command simply indicates to the LMS that an experiment is beginning, so the target machine should begin reporting logs to the LMS. This is accomplished by issuing the "powershell start-service winlogbeat" command over SSH to the target.

5.11.3.5. LMS_stop_target

This command indicates that the target machine should immediately halt its reporting of logs to the LMS. This is accomplished by issuing the "powershell stop-service winlogbeat" command over SSH to the target.

5.11.3.6. LMS_forward

This command will include a parameter to indicate the target of the forwarding. This can be used by the LMS to forward all recorded logs to the SIEM for analysis or to the SA&BS system for archival. The LMS will compress all data stored by elasticsearch using a command like "tar -zcvf logs_august_26_2022.tar.gz /var/lib/elasticsearch/data/". The compressed file can then be passed to the appropriate destination machine using the scp commad.

5.11.3.7. LMS_reset_target

This command instructs the LMS to clear and reset all logs on the target machine. The LMS will use the SSH tunnel to remotely execute the "powershell Clear-EventLog -LogName *" command. This removes all logs from the Windows machine so that a new experiment can be run without old logs in the system.

5.12. ADS

The ADS is the platform for the attacks, it will be the basis of operations for the emulated adversary. This will house and provide a platform for the toolkits used to develop the attack. Kali Linux was selected as it contains all the toolkits for penetration testing and adversary emulation that were desired. This system will be as isolated as possible from the ECS, LMS, CMS, and SA&BS during a test to prevent cyber effects disrupting the recording of data.

5.13. SIEM

The SIEM provides knowledge-based and behavior-based detection and alerting. This system will primarily provide post-processing support that can be in-line with ETE tests for real time attack feedback. The SIEM have 3 main use cases: as a post-processor for logs and PCAPs, development platform for SIEM alert and rule tuning for attacks, and as a real time alert engine for attack development.

The SIEM will be based on opensource available software (e.g., ELK) to allow greater development and integration flexibility. It will be a modular set of VMs or containers to fit the requirements of ETE experiments. It will be isolated logically from the SUT, being fed logs and PCAPs from the LMS and CMS respectively.

5.14. Scripts, Archive and Backup System (SA&BS)

The SA&BS will provide the ETE a space to store data and backups. It must have some organization system to store the data transmitted to it. This will be the central repository for all the scripts of the ETE, documentation on the SUT and ETE, data from tests,

REFERENCES

[1] https://www.mitre.org/sites/default/files/2021-11/prs-18-1174-ngci-cyber-threat-modeling.pdf

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Ben Karch	08851	brkarch@sandia.gov
Mike Rowland	08851	mtrowla@sandia.gov
Robert Bruneau	08851	rjbrune@sandia.gov
Andrew Hahn	08851	ashahn@sandia.gov
Technical Library	1911	sanddocs@sandia.gov

This page left blank