xylem
Let's Solve Water

Aug 13, 2024

Hardening the Internet of Things: Towards Designing Access Control for IoT Devices

RADHIKA UPADRASHTA

CSET'24

Agenda

Introduction

Internet of Things – General Architecture and Threats

IoT Gateway - Secure Design

Penetration Testing and Results

Conclusion

About the Authors

Xylem

A water technology company

Florida Institute of Technology

Professor of Cybersecurity

Attacks against IoT Systems

Insecam website provided access to IoT camera systems worldwide in 2014

- with default credentials or insecure remote services
- Thousands of cameras from 136 countries

Mirai Botnet

- Scanned for IoT devices running on stripped-down Linux OS in 2016
- Infected devices running with default credentials
- Gained access to 65000 devices in 20 hours

Why is it important?

Internet of Things (IoT) is growing rapidly

- Attacks against them too

Linux systems are increasingly used in embedded and IoT devices

- Linux-based malware is increasing too
- New attacks like “Living-off-the-land” technique

US White House Administration

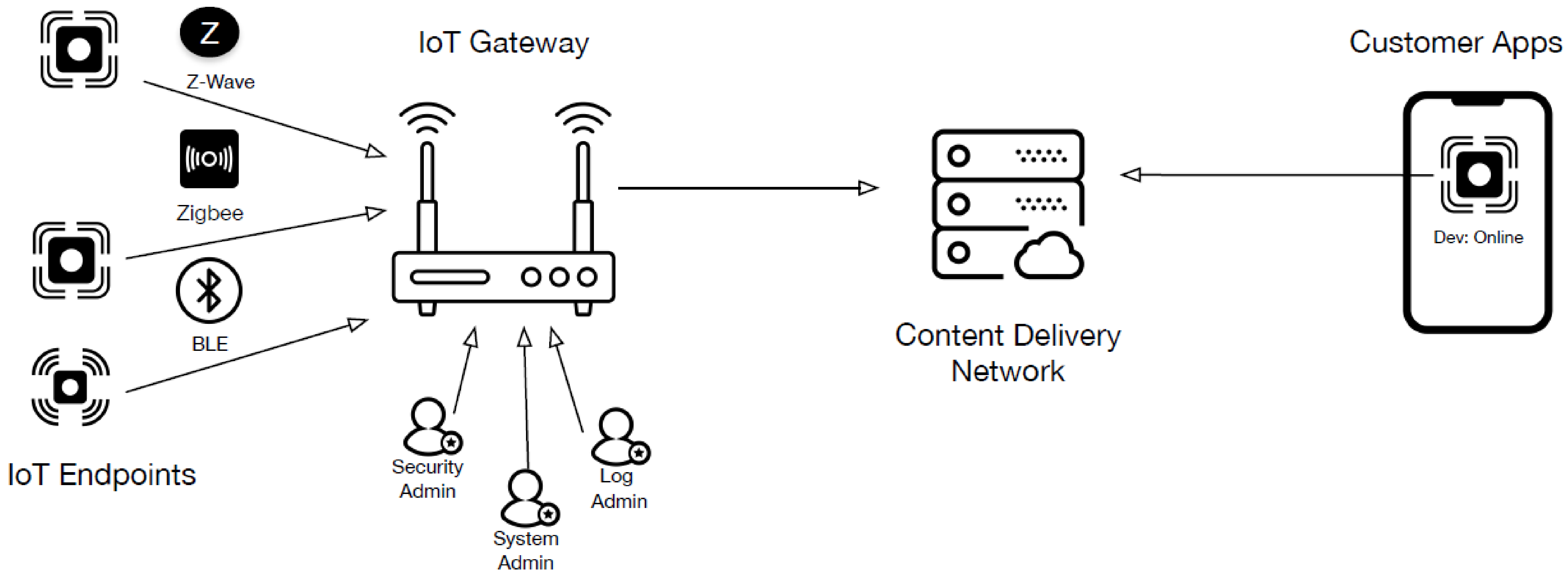
- July 2023 – US Cyber Trust Mark
- To educate and inform the customers about the security of IoT devices

How do we design a secure IoT device?

xylem
Let's Solve Water

General IoT Architecture

IoT System Architecture



Threat Model of an IoT Gateway

Threat	Security Control
Unauthorized access to sensitive data	Restrict access to data
Obtain unauthorized access to deny service	Restrict access to firewall, service configuration
Pivot to other devices	Restrict access to firewall and info about the ecosystem
Gain access to a user role by elevating privileges	Authorization checks to prevent elevation
Tamper with Logs	Restrict access to sensitive logs
Exploit vulnerable libraries on the operating system	Have a patch system, validate input

Threat Model of an IoT Gateway

Threat	Security Control
Unauthorized access to sensitive data	Restrict access to data
Obtain unauthorized access to deny service	Restrict access to firewall, service configuration
Pivot to other devices	Restrict access to firewall and info about the ecosystem
Gain access to a user role by elevating privileges	Authorization checks to prevent elevation
Tamper with Logs	Restrict access to sensitive logs
Exploit vulnerable libraries on the operating system	Have a patch system, validate input

xylem
Let's Solve Water

Gateway Design

IoT Gateway Design

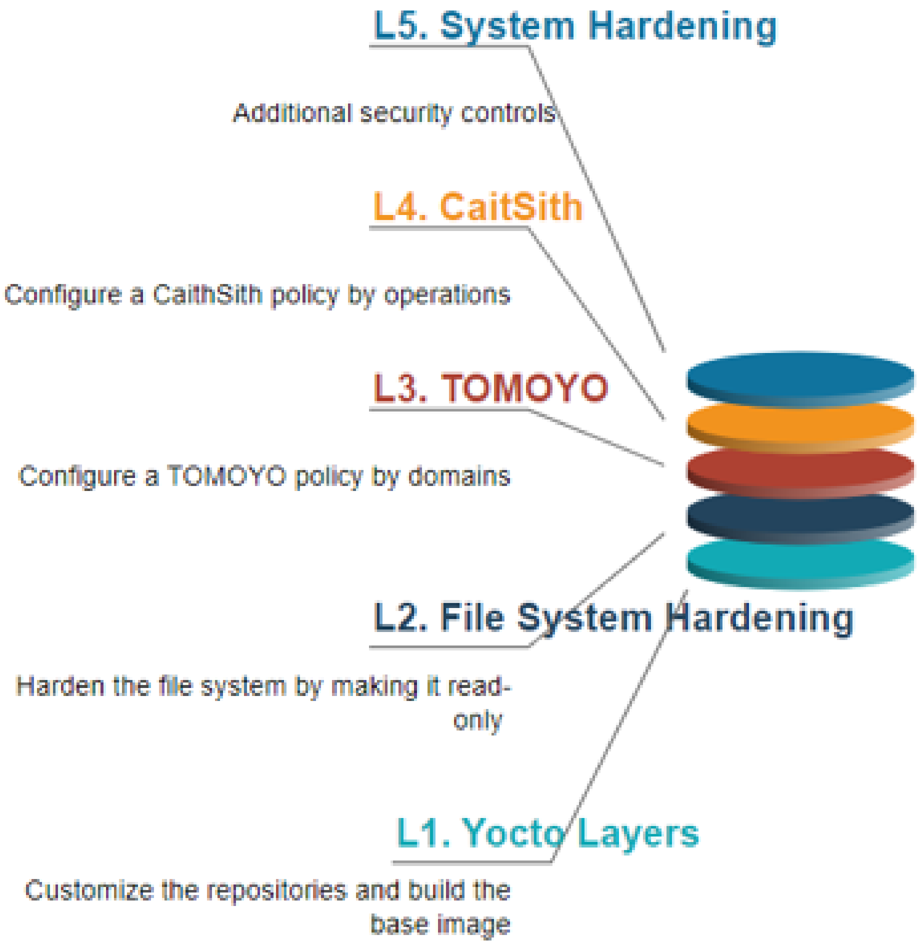
Embedded device

- ARMv5 processor, Under 500MHz, 128/256MB RAM
- Running Yocto Project-based Linux
- Using Two Linux Security Modules (LSMs) – TOMOYO and CaitSith
- Multiple user roles

No external security solutions

Secure by design

System Hardening Layers



L5. System Hardening

Additional security controls

L4. CaitSith

Configure a CaithSith policy by operations

L3. TOMOYO

Configure a TOMOYO policy by domains

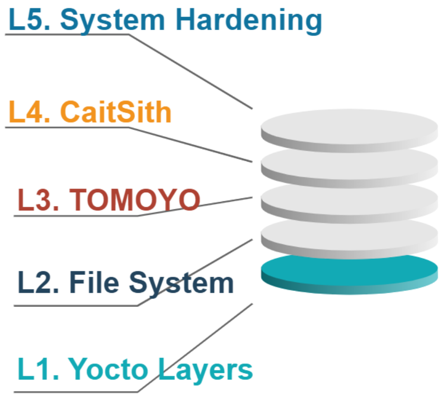
L2. File System Hardening

Harden the file system by making it read-only

L1. Yocto Layers

Customize the repositories and build the base image

L1: Yocto Project



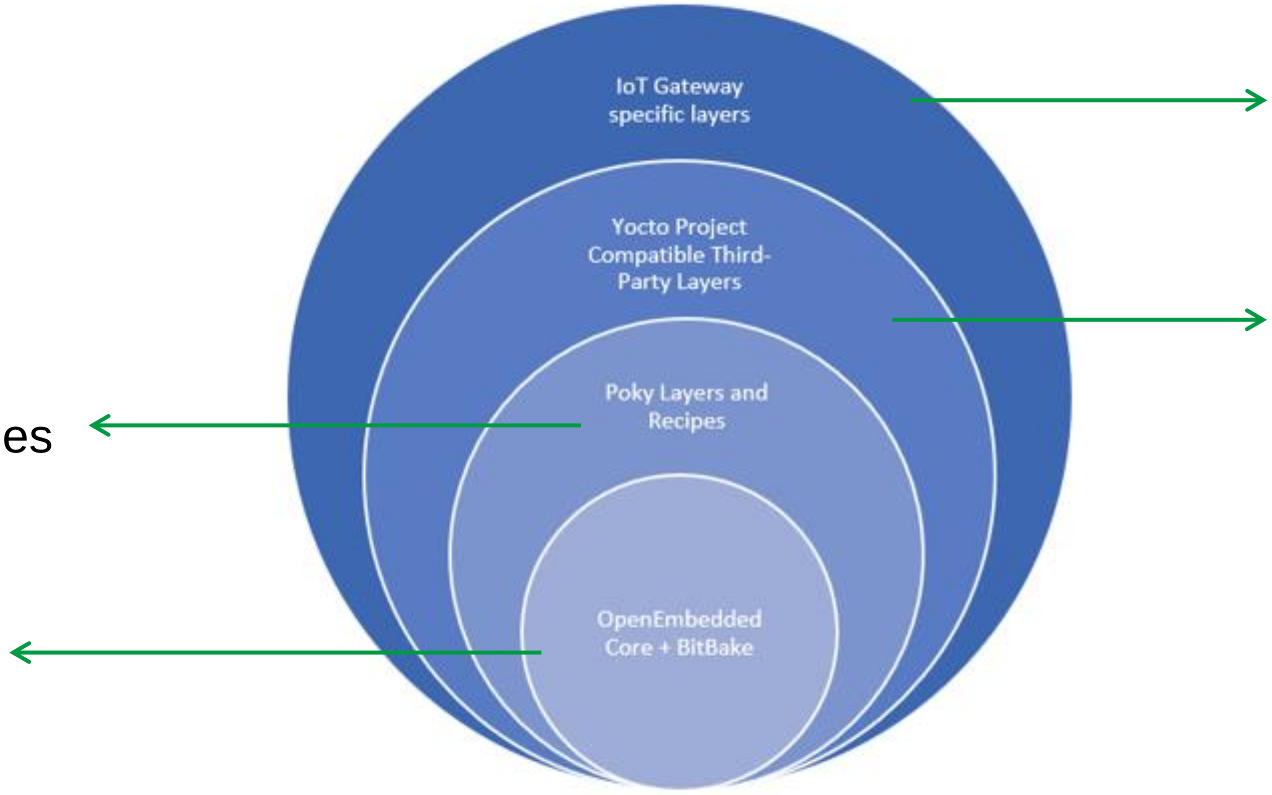
Open-Source Project to create customized Linux distribution system for several hardware architectures

Grew from the OpenEmbedded Project

Widely used in Embedded and IoT devices

Provides a set of tools to customize and build the Linux environment

Yocto Project Components


IoT Gateway specific layers
Yocto Project Compatible Third-Party Layers
Poky Layers and Recipes
OpenEmbedded Core + BitBake

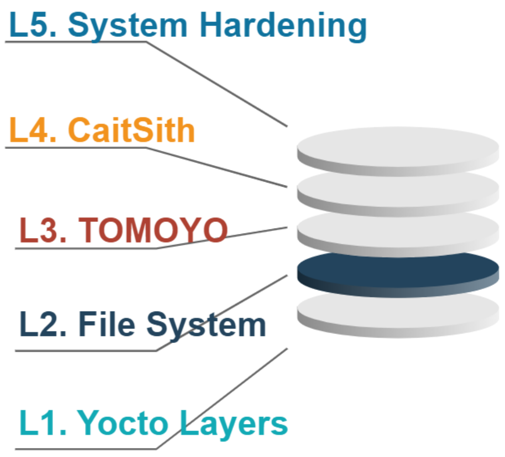
Gateway-specific instruction files

Layers of third-party libraries for various services

Build instruction files (recipes) + collection of recipes (layers)

Core build system + Build engine

L2: Customizing the File System



Removed unnecessary libraries

- No X11 packages since there is no GUI in embedded devices
- Debian: general purpose distribution with all features
- Yocto: Customizing reduces the size and the attack surface

Additional customization

- File system is made read-only
  - Done by setting the *read-only-rootfs* property in the Yocto recipe/build config file
  - Prevents modifying the system binaries
  - Explicitly configure where to write

Linux Security Modules (LSM)

LSM framework provides extensions for security checks

LSM: code compiled directly into Linux Kernel to implement access control

Major LSMs in the official kernel: AppArmor, SELinux, Smack, and TOMOYO

Only one major LSM can be enabled (as of Linux Kernel 4.19)

Linux “Capabilities” module is always enabled in the distro

Selected at

build time using *CONFIG_DEFAULT_SECURITY* argument

boot time using “*security*=“ kernel argument

LSMs in our IoT Gateway

TOMOYO as the major LSM compiled into the kernel

CaitSith as the external LSM

- can run with another major LSM
- loaded last and comes last in the order of execution

Why not SELinux?

- **Performance**
- SELinux stores the policy in the inode’s extended attributes
- Granularity of the policies slows down embedded devices that are resource-constrained

L3: TOMOYO LSM

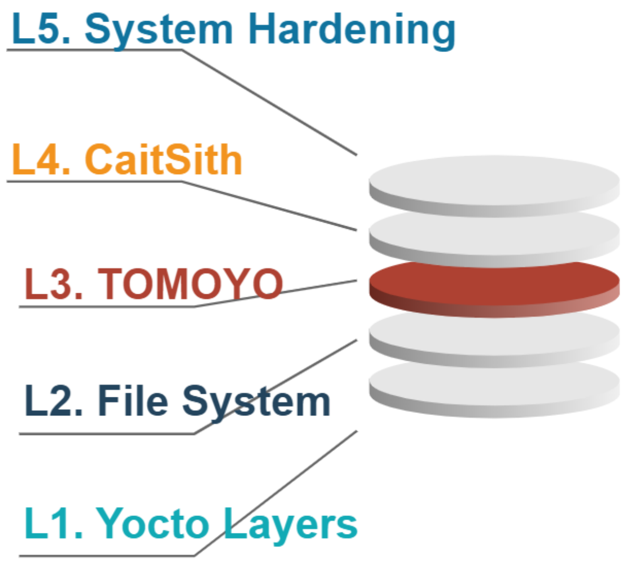
Sponsored by NTT Data Corporation

Enforces Mandatory Access Control by focusing on the behavior of the system

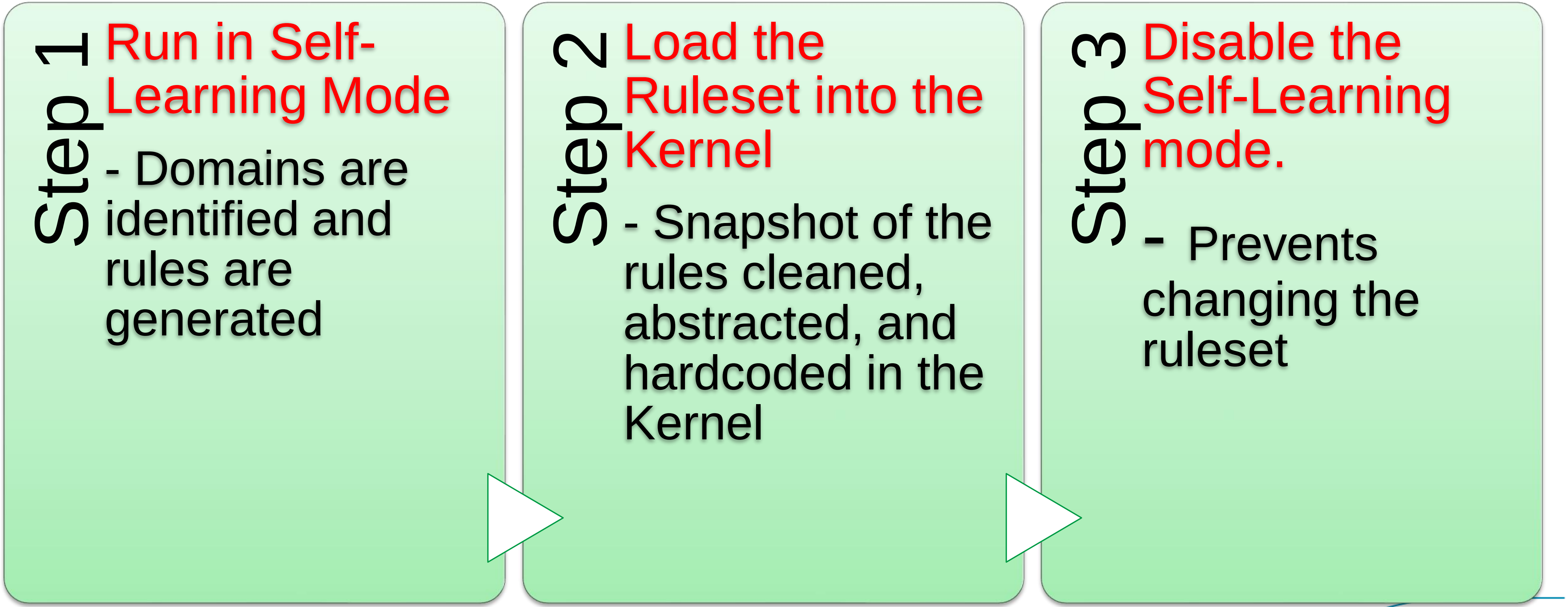
Domain: the process execution tree based on the sequence of execution

Domain for “*ls*” command

```
<kernel> /bin/sh /bin/ls
```



TOMOYO LSM Setup



Step 1

Run in Self-Learning Mode

- Domains are identified and rules are generated

Step 2

Load the Ruleset into the Kernel

- Snapshot of the rules cleaned, abstracted, and hardcoded in the Kernel

Step 3

Disable the Self-Learning mode.

- Prevents changing the ruleset

TOMOYO Features

TOMOYO Feature	Desired Security Control
fine-grained control to restrict elevating privileges to effective UID=0	Principle of Least Privilege
Enforce Role-Based Access Control by dividing privileges into custom groups	Authorization
Prevent tampering of */dev* filesystem by checking the attributes	Integrity
Restrict services through ACLs	Authorization and Confidentiality
Create firewall per application (Implicit deny per domain)	Access Control

TOMOYO Policy Example - *sudo*

Sudo from a local login shell

<kernel> /bin/sh /usr/bin/sudo

☑

ALLOW LOCAL SUDO

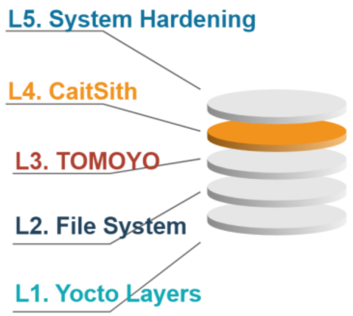
Sudo from a remote login shell

<kernel> /usr/sbin/sshd /bin/sh /usr/bin/sudo

🚫

DENY REMOTE SUDO

L4: CaitSith LSM



Sponsored by NTT Data Corporation

Derived from TOMOYO but the policy syntax is different

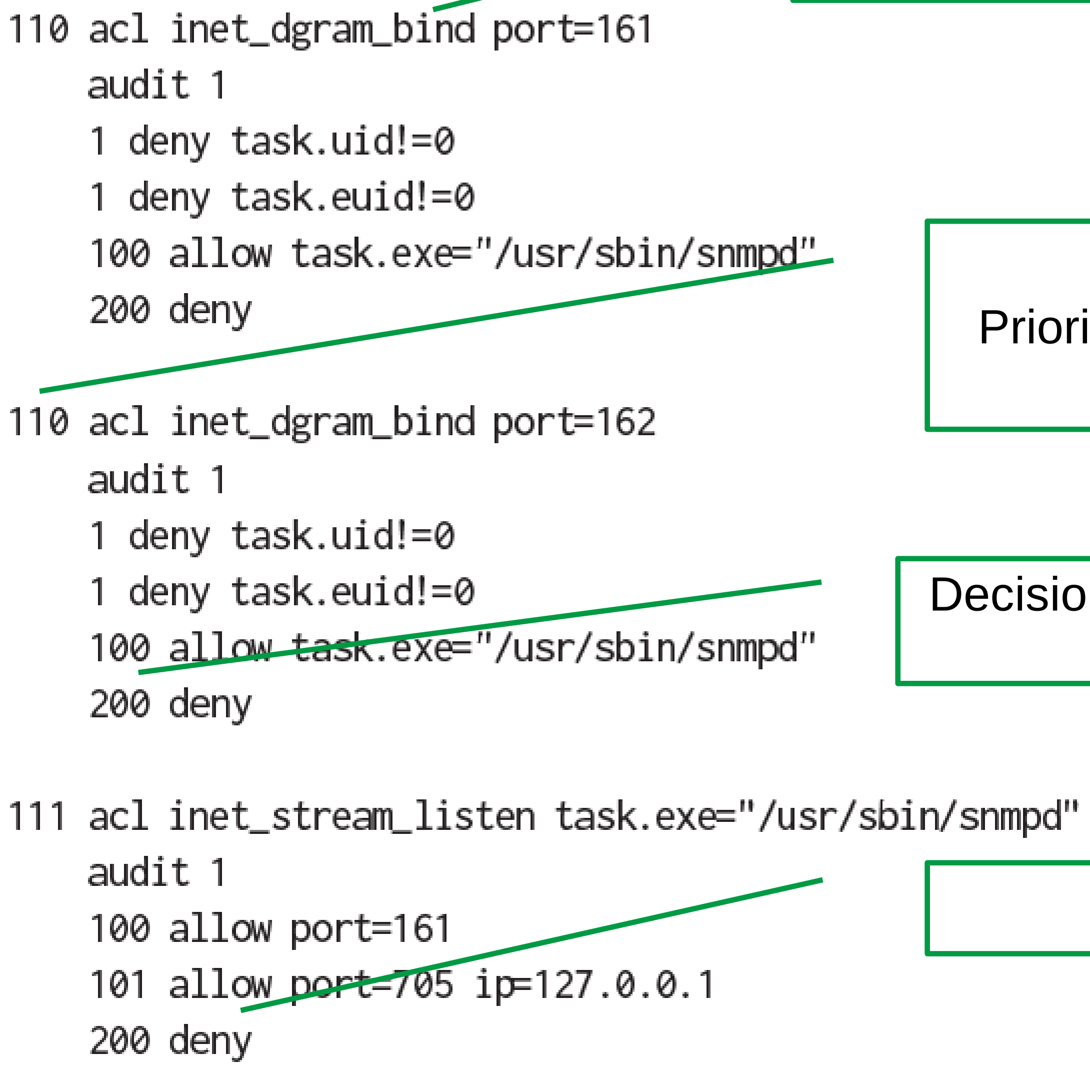
TOMOYO and CaitSith complement each other

TOMOYO Rules	CaitSith Rules
What a domain can do	Who can access files and programs at the Kernel level
Acts on the subject	Restrict access on the object

snmpd Example

TOMOYO	CaitSith
Run *snmpd* only when started as a child to init-manager which is started by Kernel	• Limit access to port 161 or 162 to only *snmpd* • Limit *snmpd* only to be able to open port 161 or 162

CaitSith Rule Example



```
110 acl inet_dgram_bind port=161
    audit 1
    1 deny task.uid!=0
    1 deny task.euid!=0
    100 allow task.exe="/usr/sbin/snmpd"
    200 deny

110 acl inet_dgram_bind port=162
    audit 1
    1 deny task.uid!=0
    1 deny task.euid!=0
    100 allow task.exe="/usr/sbin/snmpd"
    200 deny

111 acl inet_stream_listen task.exe="/usr/sbin/snmpd"
    audit 1
    100 allow port=161
    101 allow port=705 ip=127.0.0.1
    200 deny
```

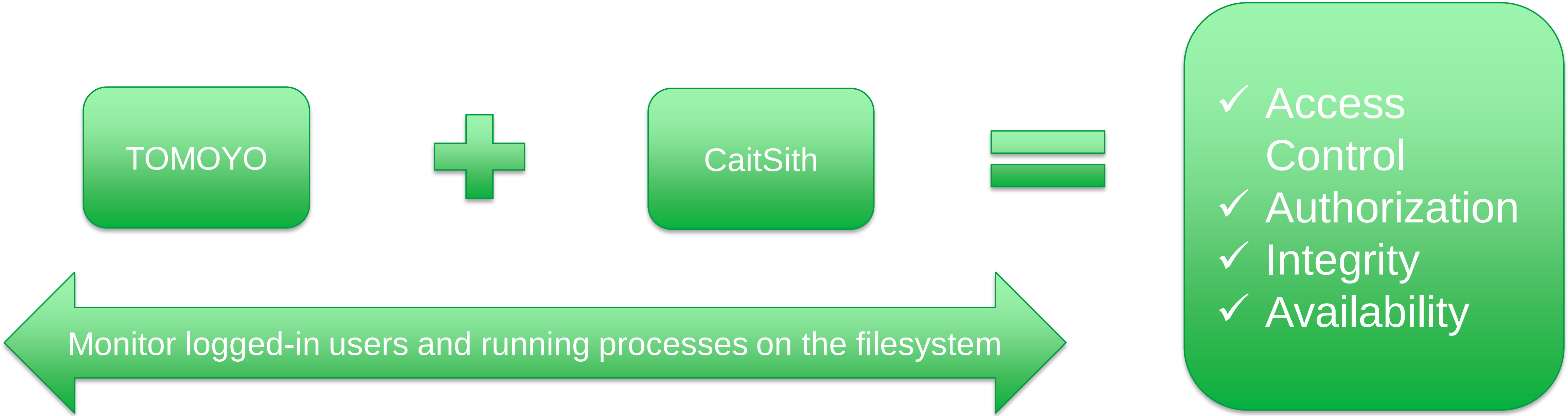
Name of the action

Priority of the ACL when there are multiple ACLs

Decision priority when there are multiple decisions within an ACL

Explicit allow or deny

TOMOYO and CaitSith

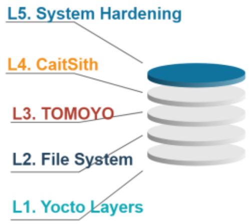


TOMOYO + CaitSith =

- ✓ Access Control
- ✓ Authorization
- ✓ Integrity
- ✓ Availability

Monitor logged-in users and running processes on the filesystem

L5: Additional Security Controls



Noexec, nodev, nosetuid options for *tmpfs* and *log* partitions

tmpfs kept small to prevent download of software

No root login or *sudo* access. Login through SSH certs by a trusted CA

Dynamic user creation after SSH login using SSH certs, removed on logout

All config files readable by system services only

Pentest!

Pentest Scope

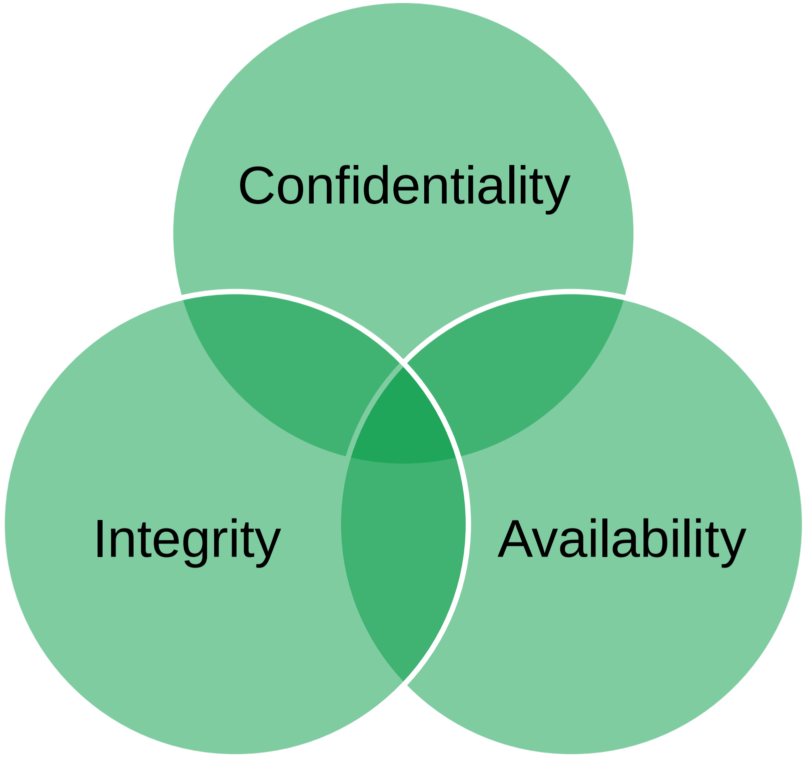
Only the IoT gateway was in scope.

Two user accounts

- ➢ Fully-privileged (all roles)
- ➢ Unprivileged (no roles)

To test:

- ✓ Confidentiality of sensitive info
- ✓ Integrity of files and services
- ✓ Availability of services



Testbed Changes

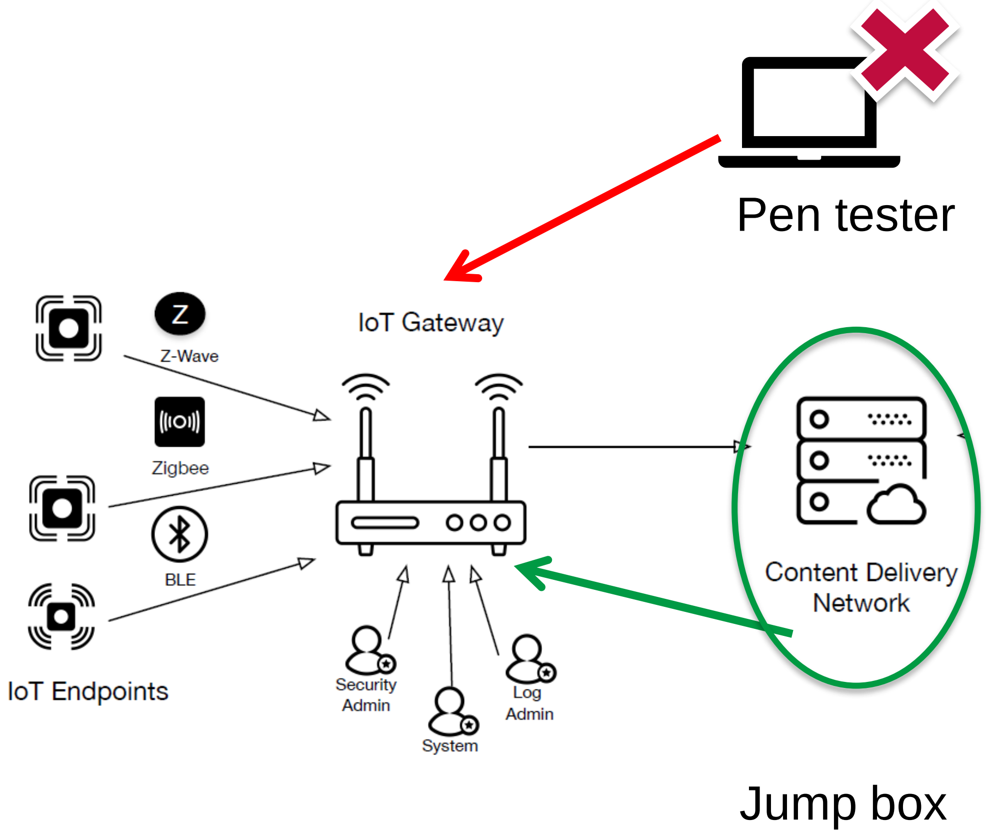
When the pentest started, the testbed could not be found.

- No ping
- No nmap

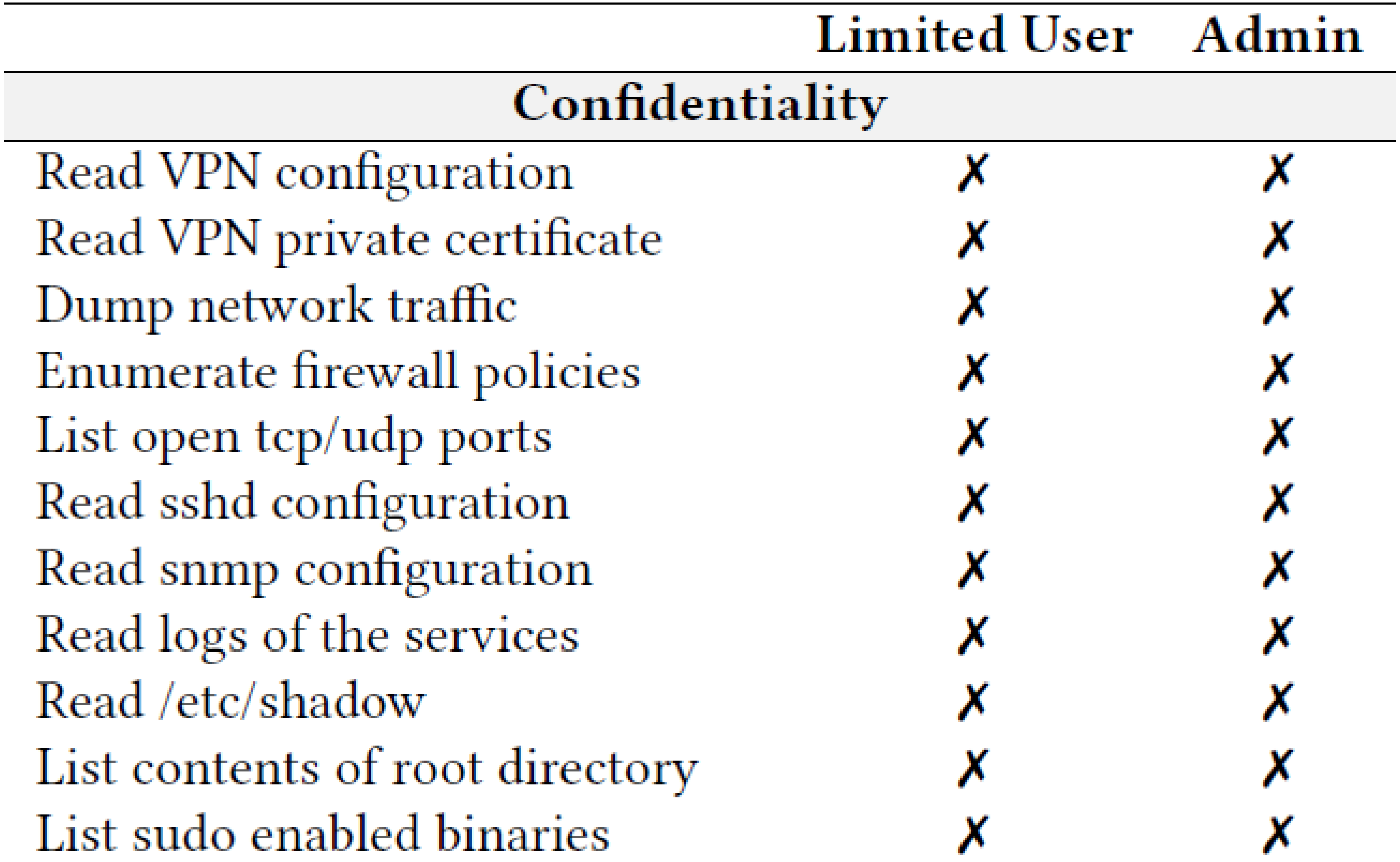
CAUSE: The gateway was blocking all connections.

CHANGE: The pentest was done by using the CDN as the jumpbox.

SSH certs generated for users on the CDN.

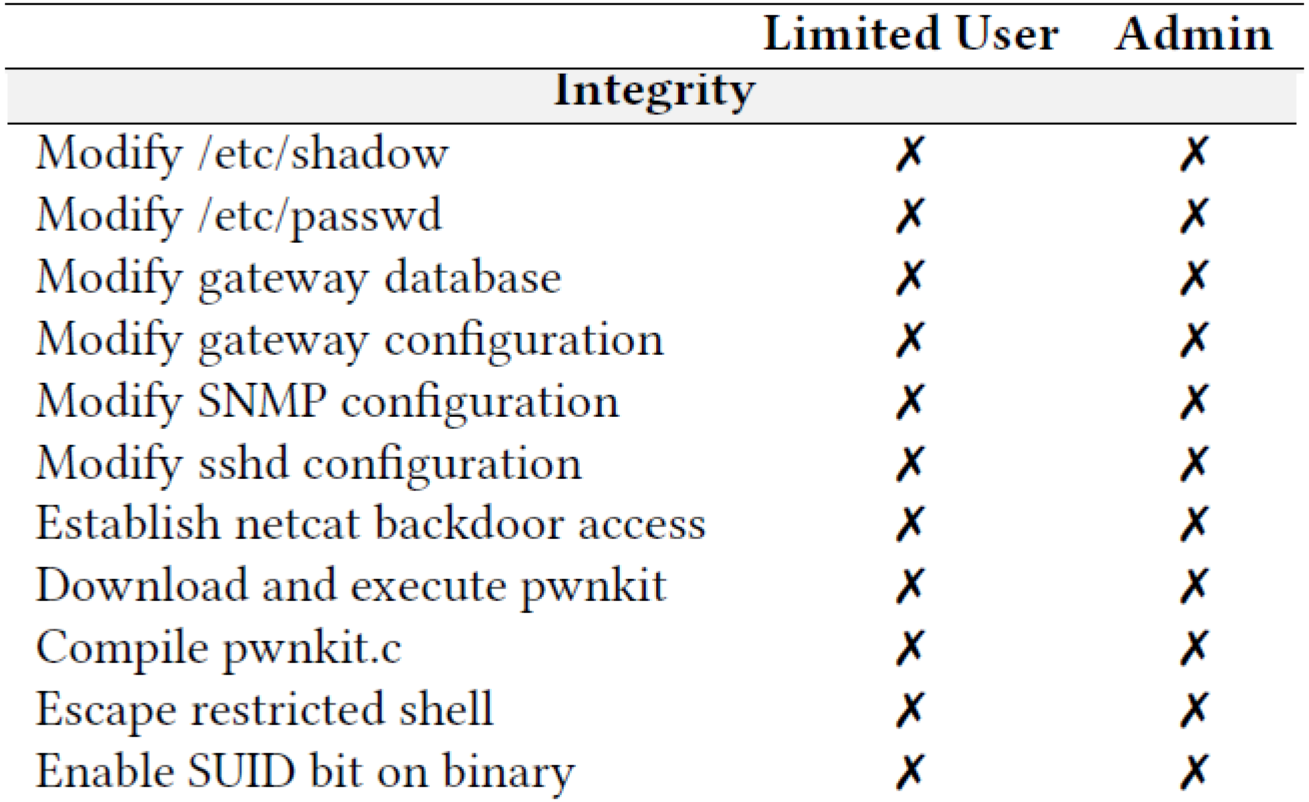


Pentest Results - Confidentiality



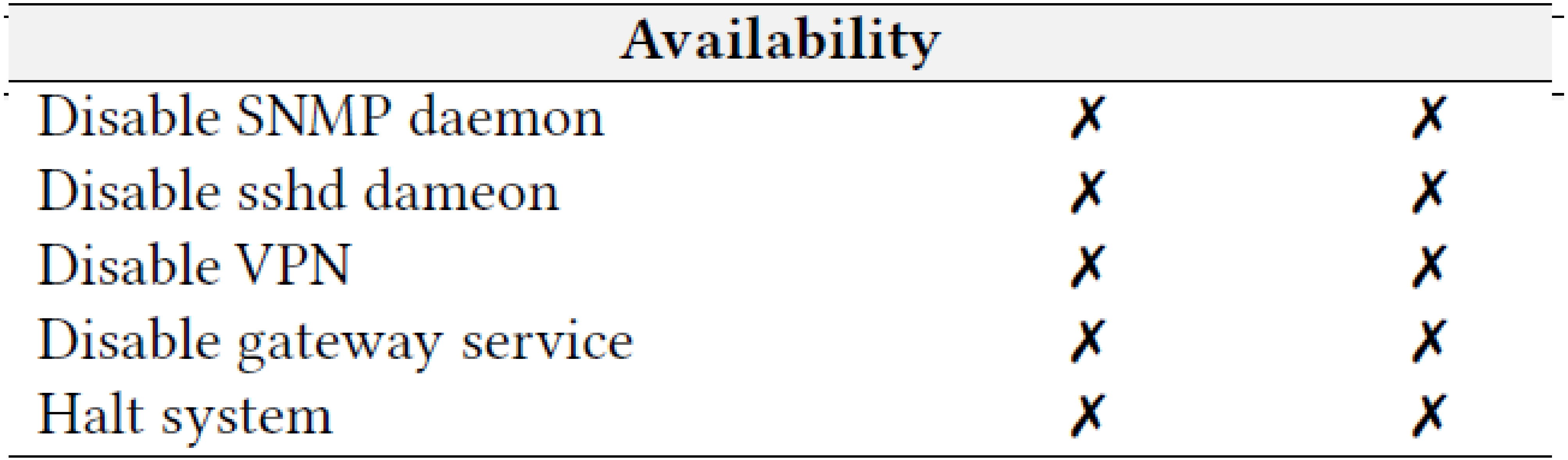
	Limited User	Admin
Confidentiality		
Read VPN configuration	✗	✗
Read VPN private certificate	✗	✗
Dump network traffic	✗	✗
Enumerate firewall policies	✗	✗
List open tcp/udp ports	✗	✗
Read sshd configuration	✗	✗
Read snmp configuration	✗	✗
Read logs of the services	✗	✗
Read /etc/shadow	✗	✗
List contents of root directory	✗	✗
List sudo enabled binaries	✗	✗

Pentest Results - Integrity



	Limited User	Admin
Integrity		
Modify /etc/shadow	✗	✗
Modify /etc/passwd	✗	✗
Modify gateway database	✗	✗
Modify gateway configuration	✗	✗
Modify SNMP configuration	✗	✗
Modify sshd configuration	✗	✗
Establish netcat backdoor access	✗	✗
Download and execute pwnkit	✗	✗
Compile pwnkit.c	✗	✗
Escape restricted shell	✗	✗
Enable SUID bit on binary	✗	✗

Pentest Results - Availability



Availability		
Disable SNMP daemon	✗	✗
Disable sshd dameon	✗	✗
Disable VPN	✗	✗
Disable gateway service	✗	✗
Halt system	✗	✗

Conclusion

Developed a secure IoT Gateway

- Using TOMOYO and CaitSith LSMs
- Implementing comprehensive access control using the LSMs
- Facilitating user and application restrictions

Discussed the results of a penetration test

This is a feasible approach to secure resource-constrained devices!

Thank you!

xylem
Let's Solve Water