A Large-Scale Analysis of the Security of Embedded Firmwares

Andrei Costin, Jonas Zaddach, Aurélien Francillon, and Davide Balzarotti, *Eurecom*
https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin

Saeid Mofrad

INTRODUCTION:

Embedded systems:

-Embedded system are computers with dedicated functionality such as routing packet, printing pages or making VOIP phone calls and etc.

-They are broadly used.

-They are getting complex and smarter and proving administrative or non-administrative interface

in example they implements

-Complex software

-Complex protocols

-Process various type of data

-They are getting heavily interconnected in private or public networks (IoT)

Firmware:

The software and data which supports the functionality of embedded system is called firmware.

Security Problem: Many examples of insecure embedded system have been seen in daily bases:

Routers
Printers
VoIP
Cars
...
Each of the above findings is a result of an individual analysis and research
It includes manual and tedious effort and does not scale.

Paper Goal:

Performing a large scale analysis to provide a better understanding of the problem

Problem with Large Scale Analysis in embedded systems:

Heterogeneity of
-Hardware, Architecture and Operating Systems
-Intended users, Requirements of the devices
-Security Goals for each firmware and device

Manual Analysis does not scale, it requires
-Finding and downloading Firmware files
-Unpacking and performing initial or subsequent analysis
-Rediscovering the same or similar bugs in other Firmware files

Previous Approaches:

Test of real devices [Bojinobv9ccs]
-It is accurate
-Does not scale very well because it needs physical devices, logistic and management
Scan Devices on the internet
Large scale testing [Coi10ACASC]
-Can only test for known vulnerability(like default passwords)
-Using Blackbox approach
Some research is too intrusive [Census2012]
-They are close to being unethical.
-They had to inject code and compromise devices hence attacking devices for study-

This paper approach to the large scale analysis

-Collect a large number of firmware images
-Perform Broad but simple static analysis
-Correlate across firmwares the findings or results

Advantages:
-No intrusive online testing
-No device at all in the experiment
-Scaleable (in terms of hardware resources, computing power)
-But there are many challenges

Challenge observation:

Mainstream Systems have centralized update and update channels and formats are well understood and very well stablished. Such as

-Microsoft update

-Apple update

-Linux update manager

Embedded system does not have centralized updates

A firmware update or firmware recovery or dump involve a combination of very restricts process including building schematics using development boards ,custom drivers or custom utilities.

-No large scale firmware dataset

Challenge A: Building a Representative Dataset:

They collected a subset of firmwares available for download.
-This is subset because many firmware and not publicly available
-No intended to have an upgrade
-Needs product purchase and registration.

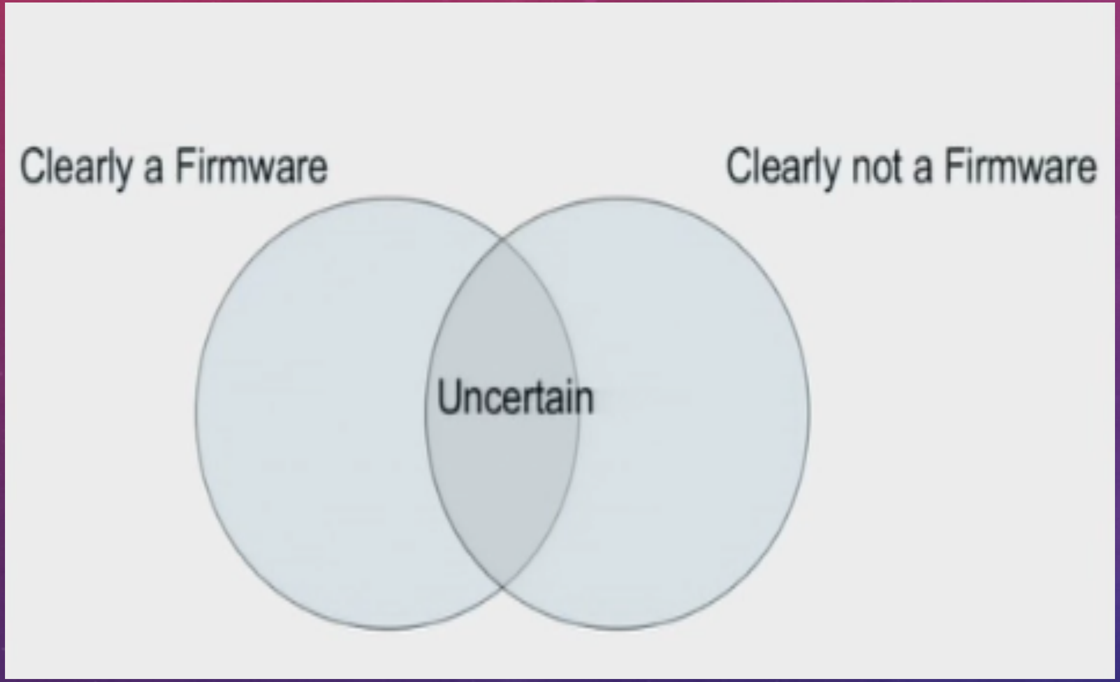
www.firmware.re project

Challenge B: Firmware Identification

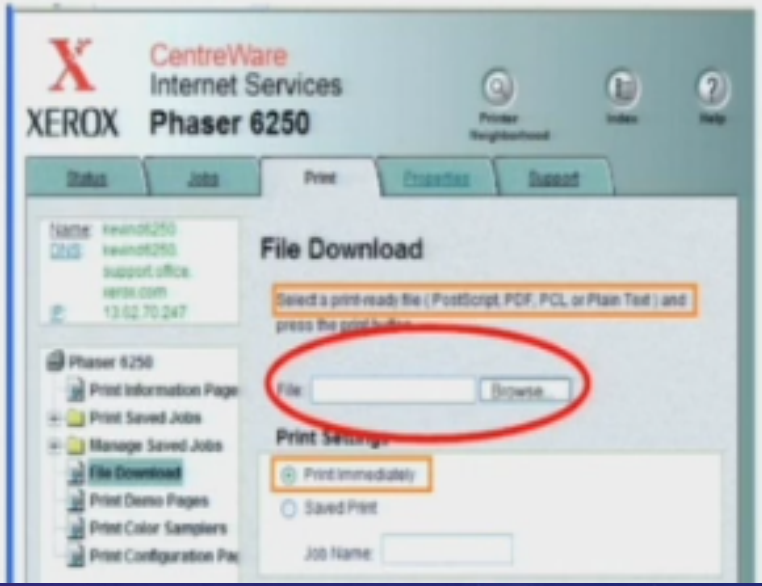
In the collected data set there are some files in gray area “Uncertain”

Good example is printer upgrade:

Upgrade by printing specially crafted PS document. So it seems not firmware file but it is firmware! So it belongs to uncertainty area and can not be discarded.



- E.g., upgrade by printing a PS document



Name: kevind6250
DNS: kevind6250.support.office.xerox.com
IP: 13.62.70.247

File Download

Select a print-ready file (PostScript, PDF, PCL or Plain Text) and press the print button.

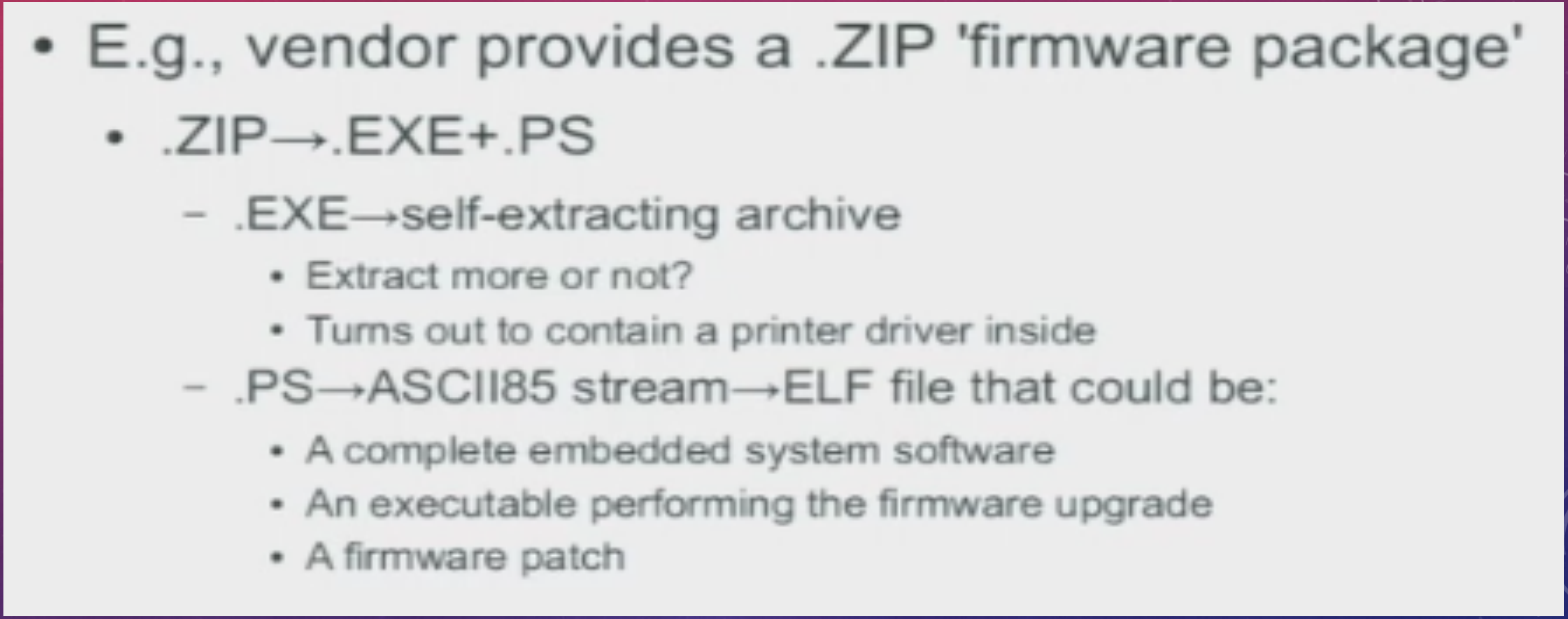
File:

Print Settings

[✓ Print Immediately]
Saved Print
Job Name:

Challenge C: Unpacking and Custom Formats

How to reliably unpack and learn format?



- E.g., vendor provides a .ZIP 'firmware package'
  - .ZIP→.EXE+.PS
    - .EXE→self-extracting archive
      - Extract more or not?
      - Turns out to contain a printer driver inside
    - .PS→ASCII85 stream→ELF file that could be:
      - A complete embedded system software
      - An executable performing the firmware upgrade
      - A firmware patch

Often a firmware image is just data binary blob or asci blob (no headers)

Paper approach for unpacking & custom format challenge

They compared several existing unpacking tools (binwalk,FRAK,BAT**)**
They used BAT (Binary analysis Toolkit)
Extended it with multiple custom unpackers.
Because Often a firmware image is just data binary blob or asci blob (no headers)
File Carving is required (to give more chance of extracting something)
Carving uses Brute force at every offset with all known unpackers
Heuristic for detecting where to stop carving since it results to high false and noisy data

Challenge D: Scalability and Computational Limits:

-Unpacking and file carving is very CPU intensive

-Unpacking results in millions of files
so manual analysis if infeasible
-One-to-One fuzzy hash comparison on big data set is CPU intensive

Challenge E: Results Confirmation

- Ann issue which is found statically
  -May not apply to a real device
  -Cannot guarantee exploitability
  In example vulnerable daemon present but never started

- Issue confirmation is difficult problem
  -Required advanced analysis (static & dynamic)
  -Often required real embedded devices for final confirmation
  -Does not scale well in heterogeneous environment (involving many devices and)

Architecture:

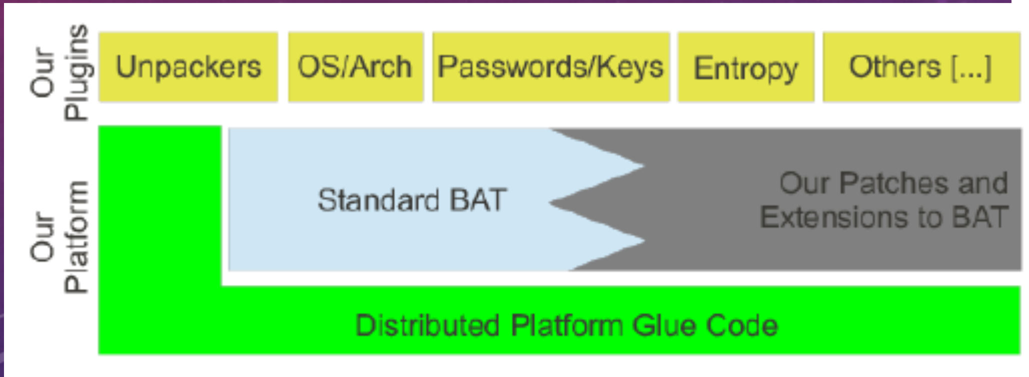
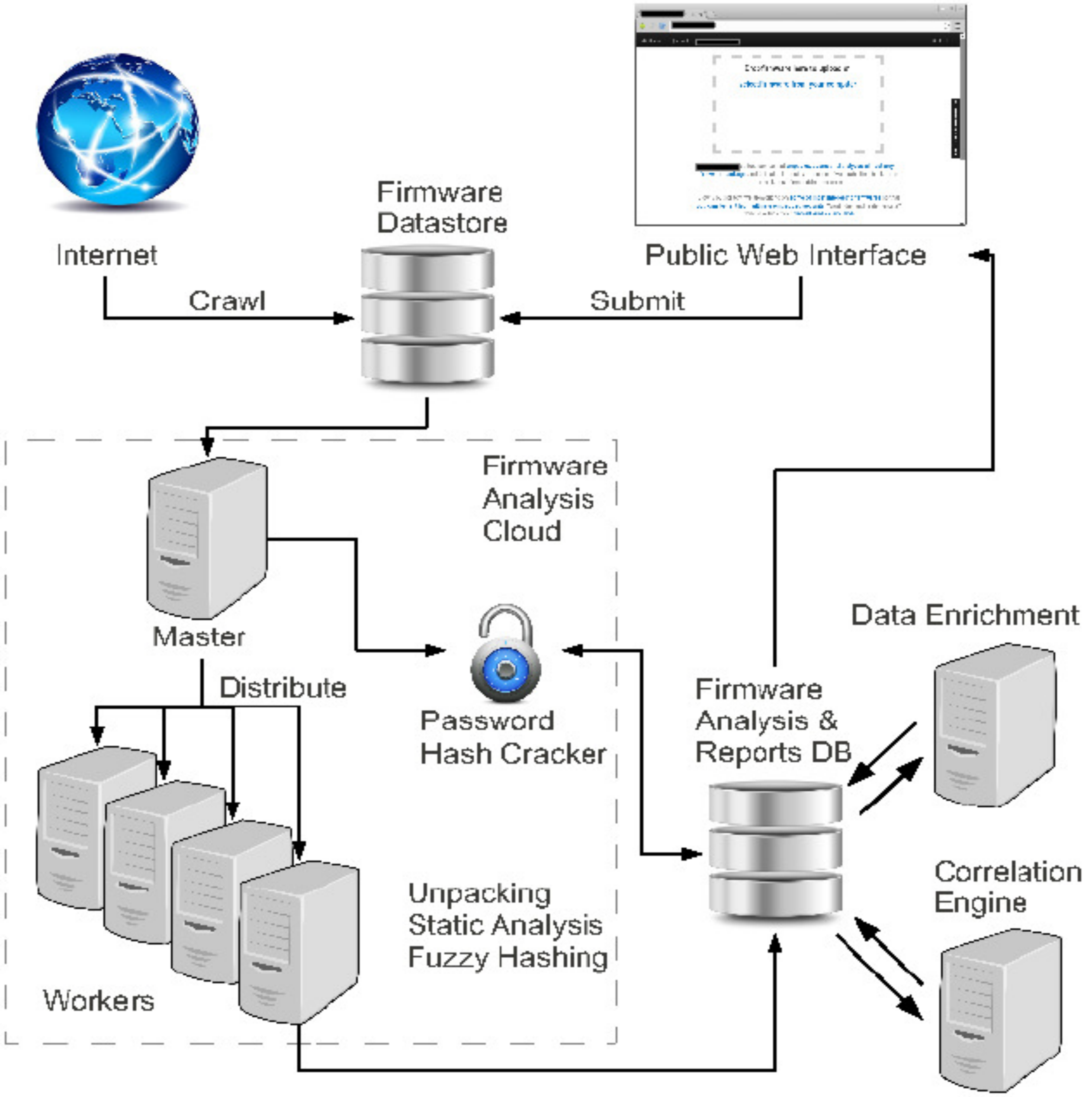


Figure 2: Architecture of a single worker node.

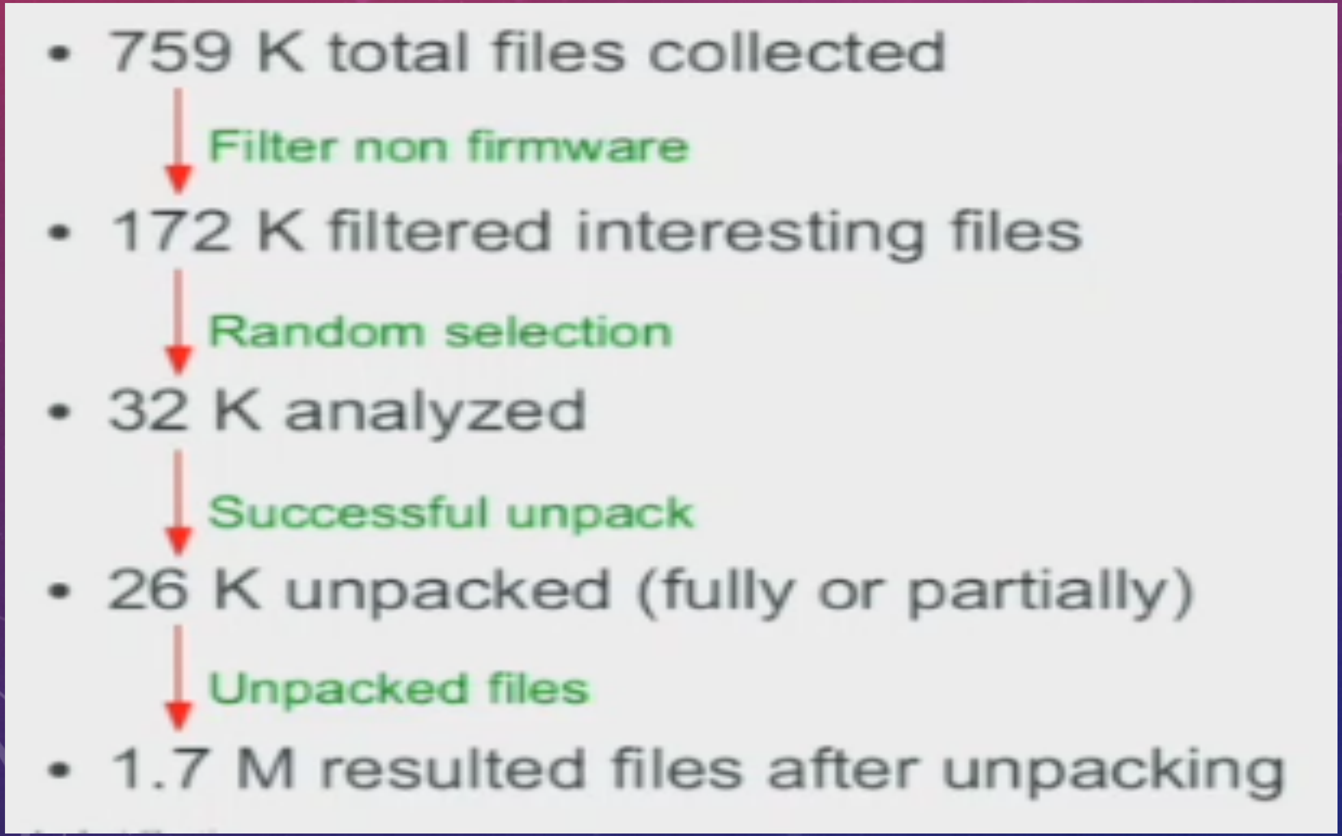


Crawler:

759k collected files. 1.8 TB of disk space

Uses FTP-index Engines and GCSE (Google custom search engine) API

Unpacking:



- 759 K total files collected

  ↓ Filter non firmware

- 172 K filtered interesting files

  ↓ Random selection

- 32 K analyzed

  ↓ Successful unpack

- 26 K unpacked (fully or partially)

  ↓ Unpacked files

- 1.7 M resulted files after unpacking

Static analysis:

- **Correlation/clustering:**
  based on fuzzy hashes, Private SSL keys, Credentials

- Web Server Configs, hard coded credential, Code Repositories

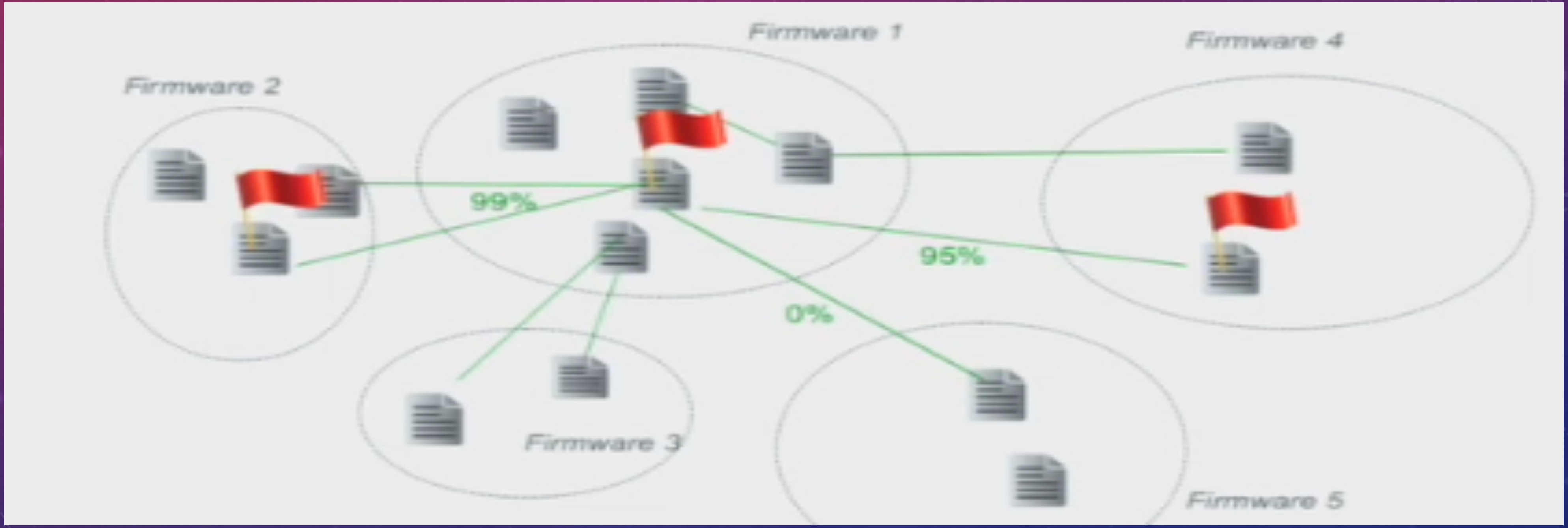
- Data Enrichment:
  Version Banners
  Specific Keywords(e.g Telnet, Shell, UART, backdoor)

Example of Correlation:

correlation via fussy hashes similarities(ssdeep,sdhash)

Strong Correlation between two firmware by having Shared Credentials and Self-Signed Certificates

E.G vulnerability propagation.

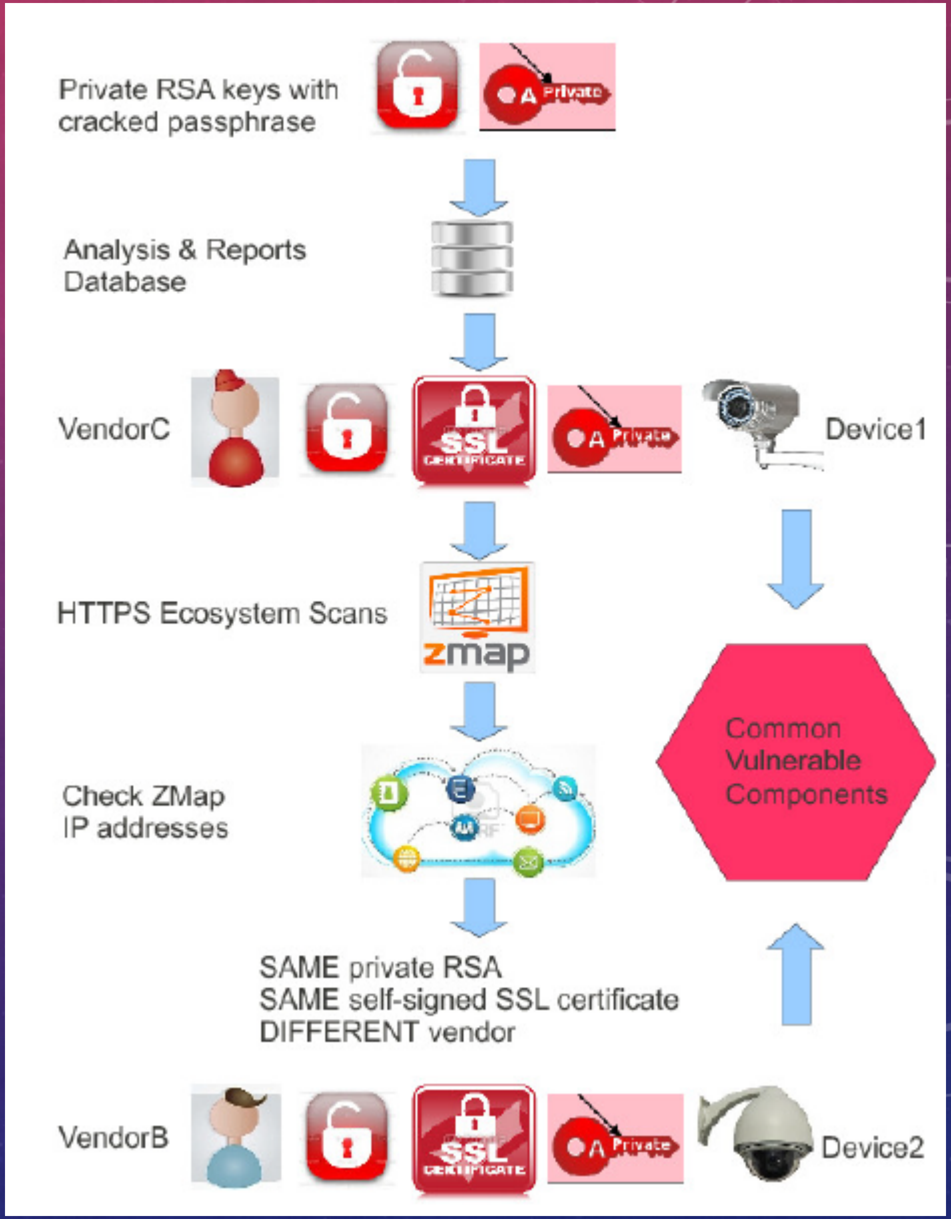


Case Studies:

1-Backdoors in Plain Sight
the backdoor was found to be activated by the string “xmlset roodkcableoj28840ybtide” (i.e., edit by 04882 joel backdoor in reverse).
they performed a string search in the dataset with various backdoor related keywords and found 1198 matches, in 326 firmware candidates.

2-Private SSL (RSA key)

- many firmware images containing public and *private RSA key* pairs
- -platform automatically extracts the fingerprint of the public keys, private keys and SSL certificates.
- -keys are then searched in ZMap’s HTTPS survey Database
- *Vendor C*’s SSL certificate was found to be used by around 30K online IP addresses.
- - then fetched the web pages available at those addresses (without trying to authenticate). Surprisingly, Returned CCTV cameras branded by another vendor – *Vendor B*


Private RSA keys with cracked passphrase
Analysis & Reports Database
VendorC
Device1
HTTPS Ecosystem Scans
zmap
Common Vulnerable Components
Check ZMap IP addresses
SAME private RSA
SAME self-signed SSL certificate
DIFFERENT vendor
VendorB
Device2

Result Summary:

- 38 new vulnerabilities(CVE)
- Correlated them to 140k online devices.
- Affected 693 firmware files by at least one of these vulnerabilities.

Conclusion:

A broader view of firmwares

- Not only beneficial but necessary for discovery and analysis
- Correlation reveals firmware relationship

Show how vulnerability reappear in different product or vendor

- Could allow how firmware are evolve or get fixed

- There are many hidden vulnerability
- Security is a tradeoff with cost and time to market and Security is not priority of some vendors

REFERENCE:

- https://www.usenix.org/node/184450