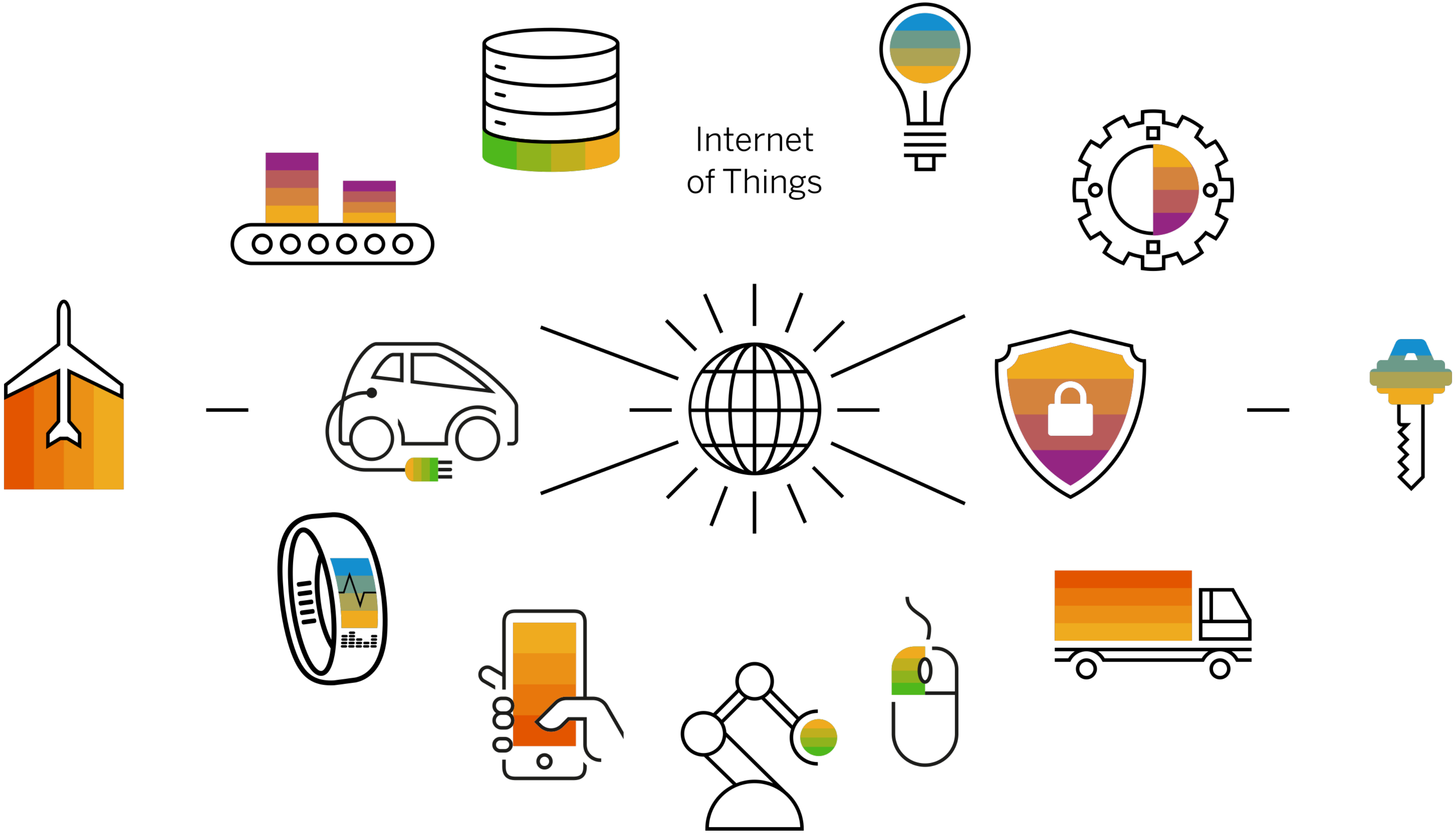
A Large-Scale Empirical Analysis of the Vulnerabilities Introduced by Third-Party Components in IoT Firmware

Binbin Zhao Shouling Ji Jiacheng Xu Yuan Tian Qiuyang Wei Qinying Wang
Chenyang Lyu Xuhong Zhang Changting Lin Jingzheng Wu Raheem Beyah

IoT Device

IoT devices are popular but also in danger.



IoT Device

IoT devices are popular but also in danger.

We Decide What You See: Remote Code Execution on a Major IPTV Platform

June 5, 2019

Hackers Target 300,000 D-Link, Micronet, Tenda, And TP-Link Routers

March 4, 2014

https://msrc.microsoft.com/

CVE-2018-8479 - Azure IoT SDK Spoofing Vulnerability

A spoofing vulnerability ...

...ing for the C SDK library

| SC Media

Users of IoT products from three major vendors at risk of DoS ...

... expose product users to denial-of-service (DoS) attacks, remote code execution, and sensitive data leakage. The three IoT vendors – Softing ...

25 Jan 2021

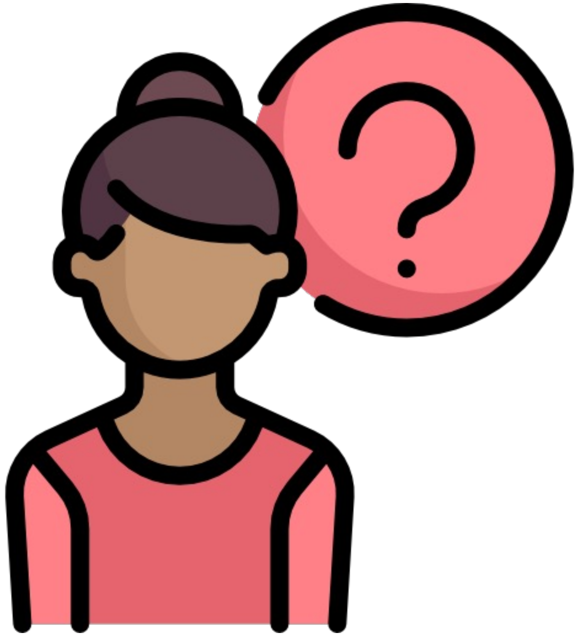
Third-party Component

- Third-party components (TPCs) are widely used in IoT firmware but also bring potential security risks.
- Many TPCs have known vulnerabilities, e.g., the Heartbleed vulnerability in OpenSSL.
- It is important to identify the vulnerable TPCs used in IoT firmware.

Limitations of Existing Works

- **Pay less attention to the vulnerabilities caused by TPCs in firmware.**
- **Lack the consideration of non-Linux based firmware.**
- **Unscalable on large-scale firmware security analysis.**

How to build a scalable and automatic tool to identify the vulnerable TPCs used in IoT firmware?



Challenges

¤ **Firmware Dataset Construction**

- **No publicly accessible firmware dataset for research.**
- **More and more vendors begin to prohibit the public from downloading firmware.**

¤ **Firmware Processing**

- **Difficult to unpack and extract different kinds of firmware images.**
- **Hard to deal with the monolithic firmware.**

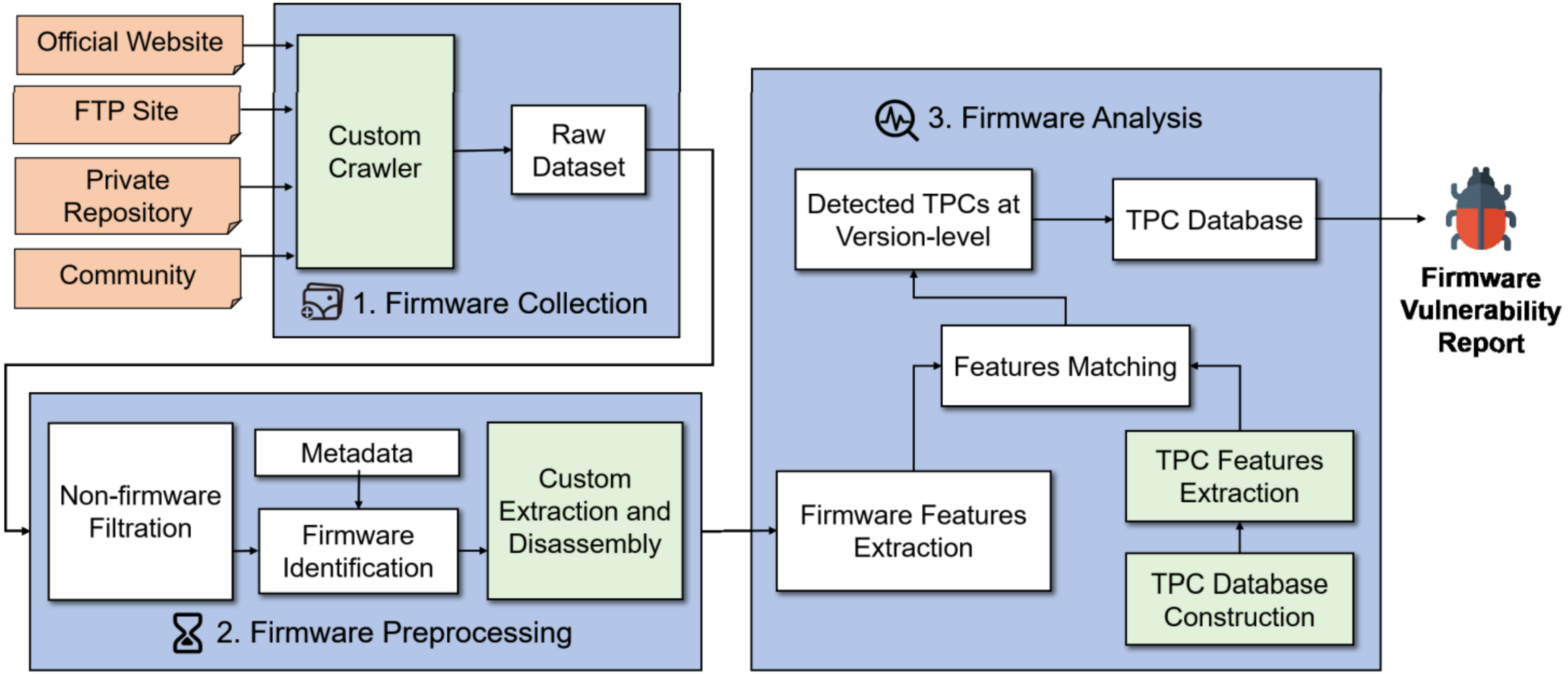
¤ **TPC Detection and Vulnerability Identification.**

- **Hard to distinguish the same TPCs at version-level in firmware.**
- **No TPC database that indicates the possible TPCs used in firmware.**

System Design

Overview of FirmSec

FirmSec: a scalable and automatic framework to analyze the TPCs used in firmware and identify the corresponding vulnerabilities.

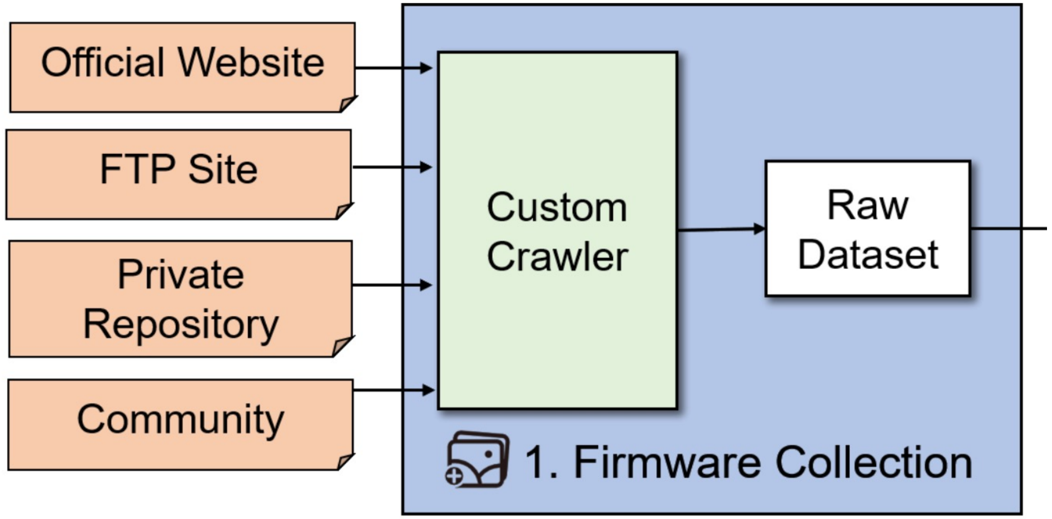


Architecture of FirmSec

Firmware Collection

¤ **Public Firmware**

- **Official website.**
- **FTP site.**
- **Community, e.g., related forums and GitHub repositories.**



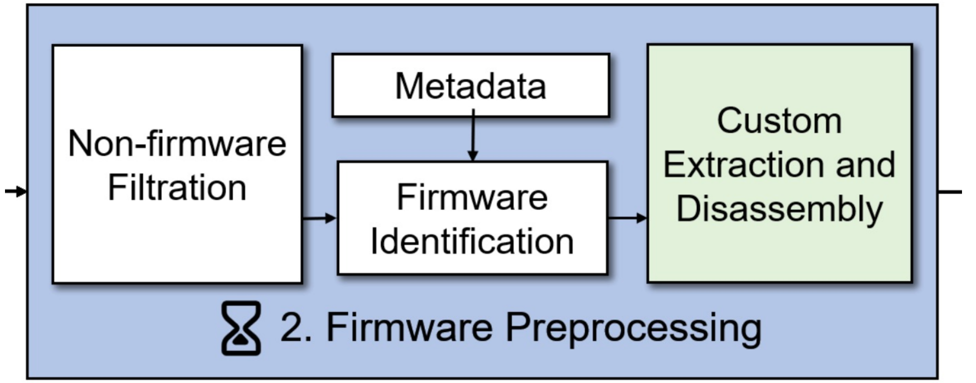
¤ **Private Firmware**

- **TSmart's private firmware repository.**

Firmware Preprocessing

¤ Firmware Filtration

- **Filter the obvious non-firmware files through suffix matching.**
- **Adopt Binary Analysis Next Generation (BANG) to get rid of other non-firmware files.**



¤ Firmware Identification

- **Extract the information from the metadata files.**
- **Adopt binwalk to scan firmware images.**

Firmware Preprocessing

¤ **Firmware Extraction**

- **Equip binwalk with 3 plugins to deal with SquashFS, JFFS2, and YAFFS filesystems.**

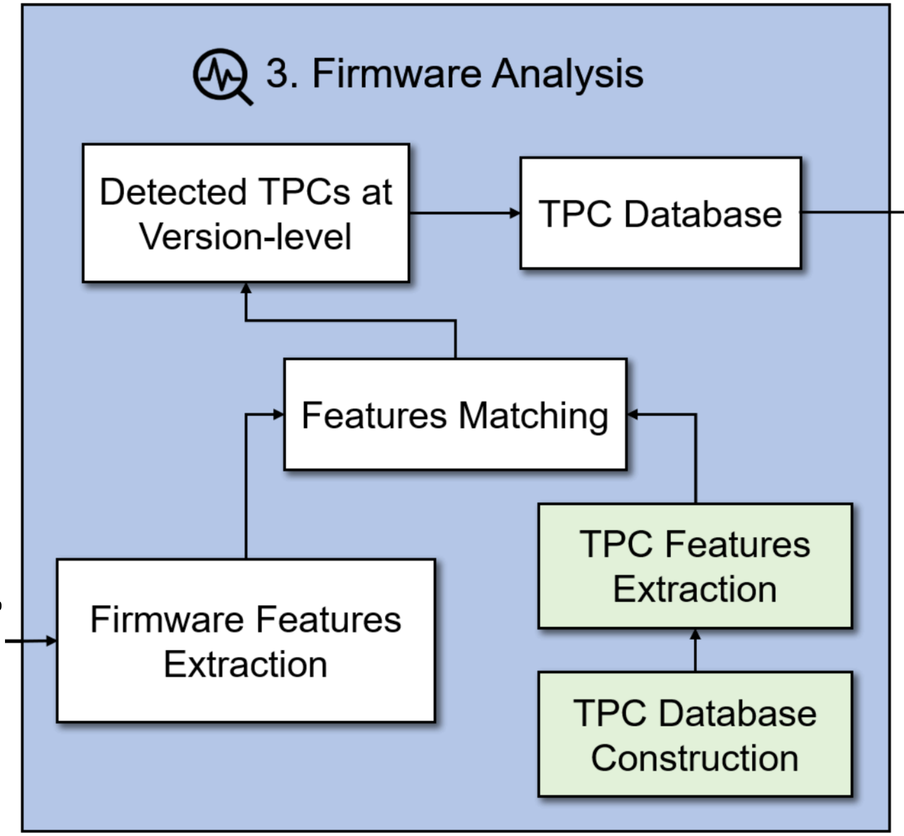
¤ **Firmware Disassembly**

- **Analyze the processors used in monolithic firmware to recover the missing information, e.g., the RAM/ROM start address.**
- **Customize 7 plugins for IDA to disassemble 7 kinds of processors.**

Firmware Analysis

¤ TPC Database Construction

- **Collect the possible TPCs used in IoT firmware from four sources.**
  1. **Linked libraries extracted from the firmware.**
  2. **Open-source IoT projects.**
  3. **SDKs from multiple IoT platforms, e.g., AWS IoT.**
  4. **A shortlist of TPCs from TSmart.**
- **Query the CVE database, NVD, and CVE Detail to collect the TPC CVEs.**



Firmware Analysis

Insight: Syntactical features and control flow graph (CFG) features are hardly changed between the source files and binaries.

¤ TPC Detection

- **Extract the above two features from TPCs and firmware.**
- **Use the edit distance and ratio-based matching to calculate the similarity of syntactical features.**
- **Use customized Gemini to compare the CFG features.**

Firmware Analysis

¤ TPC Feature Extraction

1. Implement a parser to extract the syntactical features from the C/C++ source files of TPCs.
   - Sharing syntactical features: the common syntactical features in all versions of the TPC.
   - Unique syntactical features: the specific syntactical features in each version of the TPC.
2. Extract the attributed control-flow graphs (ACFGs) from each version of TPCs.
   - Each vertex in an ACFG is a basic block labeled with a set of attributes.
   - Use three extra function-level attributes.

Type	Attribute Name	FirmSec	*Gemini* [56]
Block-level	String Constants	✓	✓
	Numeric Constants	✓	✓
	No. of Transfer Instructions	✓	✓
	No. of Calls	✓	✓
	No. of Instructions	✓	✓
	No. of Arithmetic Instructions	✓	✓
	No. of Offspring	✓	✓
	Betweenness	✓	✓
Function-level	No. of Basic Blocks	✓	✗
	No. of Edges	✓	✗
	No. of Variables	✓	✗

Block-level and Function-level Attributes

Firmware Analysis

¤ Firmware Feature Extraction

1. **Extract the syntactical features from firmware.**
   - **Equip IDA with many signature files of TPCs.**

2. **Extract the ACFGs from the disassembled firmware.**
   - **Customize an extraction tool by integrating our firmware disassembly module.**

Firmware Analysis

¤ **Syntactical Feature Matching**

1. **Calculate the edit distance.**
   - $D(S_{TPC}, S_{Firmware})$ **represents the edit distance between the syntactical features from TPCs and firmware.**
   - **If** $D(S_{TPC}, S_{Firmware})$ **exceeds the given threshold α, we regard the features are matched.**

2. **Ratio-based matching.**
   - **Record the number of matched features.**
   - $\frac{S_{TPC} \cap S_{Firmware}}{S_{TPC}}$ **represents the ratio of matched features to all features extracted from the TPC.**
   - **If** $\frac{S_{TPC} \cap S_{Firmware}}{S_{TPC}}$ **exceeds the given threshold** β**, we regard the TPC is matched.**

Firmware Analysis

¤ **CFG Feature Matching**

1. **Implement the customized Gemini.**
   - **Give a high weight to the complex CFGs.**
   - **Normalize and aggregate the similarity of each ACFG based on the weight of the corresponding CFG.**

2. **Matching.**
   - $Sim(TPC, Firmware)$ **represents the similarity between the CFG features from TPCs and firmware.**
   - **If** $Sim(TPC, Firmware)$ **exceeds the given threshold γ, we regard the TPC is matched.**

Firmware Analysis

¤ **Vulnerability Identification**

1. **Results combination.**
   - **Take the union of syntactical feature matching results and CFG feature matching results as the final results.**
2. **Versions check.**
   - **Implement a script to automatically query the TPC database with the TPCs and the corresponding versions (e.g., OpenSSL 0.9.8).**
   - **Record the returned vulnerability information.**
3. **Report generation.**
   - **Indicate the potential risks.**
   - **Provide suggestion for fixing the vulnerabilities.**

System Evaluation

Experiment Settings

¤ Dataset Composition

- 34,136 valid firmware images, including 11,086 public firmware images and 23,050 private firmware images.
- Involve 13 vendors and 35 kinds of different IoT devices.
- Camera: 2,694 (7.9%), Router: 7,293 (21.3%), Switch 1,191 (3.5%), Smart Homes: 23,050 (67.5%).
- ARM (23.9%) takes the majority and MIPS follows (4.9%).
- 12,342 (36.2%) firmware images are Linux-based and 21,794 (63.8%) firmware images are non-Linux based.

Evaluation

¤ Model Accuracy

1. Train the customized Gemini on the training set of Dataset I for 100 epochs.
   - Dataset I includes the ACFGs we extracted from 1,192 TPCs in our TPC database.
   - Split Dataset I into three subsets for training, validation, and testing respectively according to the ratio of 6:2:2.
2. Save the model when it achieves the best AUC (Area Under the Curve) on the validation set.
3. Test the model on the testing set.
   - Our AUC is 0.953 while the AUC of the original Gemini is only 0.912.

Evaluation

¤ Threshold Selection

1. Manually create Dataset II, which includes 17, 918 TPC-version pairs, for threshold selection.
2. Utilize the true positive rate (TPR) at version-level as the metric to select the appropriate thresholds.
3. Combine the three thresholds and their corresponding TPR as a four-dimensional vector: [α, β, γ, TPR].
4. Select the thresholds when the TPR reaches the highest.
5. FirmSec achieves the highest TPR (91.47%) when α=0.74, β=0.52, γ=0.64.

Evaluation

¤ **Performance**

1. **Manually create Dataset III, which includes 19, 645 TPC-version pairs, for performance evaluation.**
2. **FirmSec achieves 92.09% precision, 95.24% recall at TPC-level, and 91.03% precision, 92.26% recall at version-level.**
3. **FirmSec is better than three state-of-the-arts both at TPC-level and version-level.**

Tools	TPC-level		Version-level	
	Precision	Recall	Precision	Recall
FIRMSEC	92.09%	95.24%	91.03%	92.26%
Syntax-based	92.38%	86.29%	91.47%	81.66%
CFG-based	93.72%	82.76%	94.65%	80.90%
Gemini [56]	89.60%	74.19%	90.78%	71.73%
BAT [38]	70.74%	56.38%	NA	NA
OSSPolice [31]	86.63%	71.85%	82.51%	67.05%

Comparison of FirmSec, Gemini, BAT, and OSSPolice

Data Characterization

TPC Usage

¤ **Results**

- **Successfully unpack and disassemble 96% firmware images.**
- **Identify 584 different TPCs used in 34,136 firmware images.**

Vendor	Category	# Firmware	# TPC	# $\overline{TPC}$	# Vul.	# $\overline{Vul.}$
Xiongmai	Camera	520	232	0.45	313	0.60
Tomato-shibby	Router	230	2,088	9.08	11,948	51.95
Phicomm	Router	107	405	3.79	1,818	16.99
Fastcom	Router	149	274	1.83	1,849	12.41
	Unknown	10	0	0	0	0
Trendnet	Camera	477	136	0.28	1,395	4.32
	Router	336	1,762	5.24	7,903	23.52
	Switch	162	366	2.26	3,157	19.49
	Unknown	106	164	1.54	158	1.52
Xiaomi	Router	21	251	11.95	2,440	116.19
TP-Link	Camera	319	1,981	6.21	27,001	84.64
	Router	606	4,222	6.97	30,612	50.51
	Switch	484	77	0.16	795	1.64
	Unknown	48	67	1.40	639	13.31
D-Link	Camera	360	113	0.31	737	2.04
	Router	552	2,823	5.11	14,495	26.26
	Switch	545	80	0.15	1062	1.95
	Unknown	91	30	0.33	266	2.92
Hikvision	Camera	139	8	0.06	127	0.91
Foscam	Camera	113	0	0	0	0
Dahua	Camera	419	43	0.10	430	1.03
TSmart	Smart Homes	23,050	856	0.04	4,353	0.19
OpenWrt	Router	5,292	300,020	56.69	13,486	2.55

Analysis Results of the Dataset

TPC Usage

¤ **Findings**

- **Routers from OpenWrt contain the most TPCs.**
- **Smart Homes from TSmart use few TPCs.**

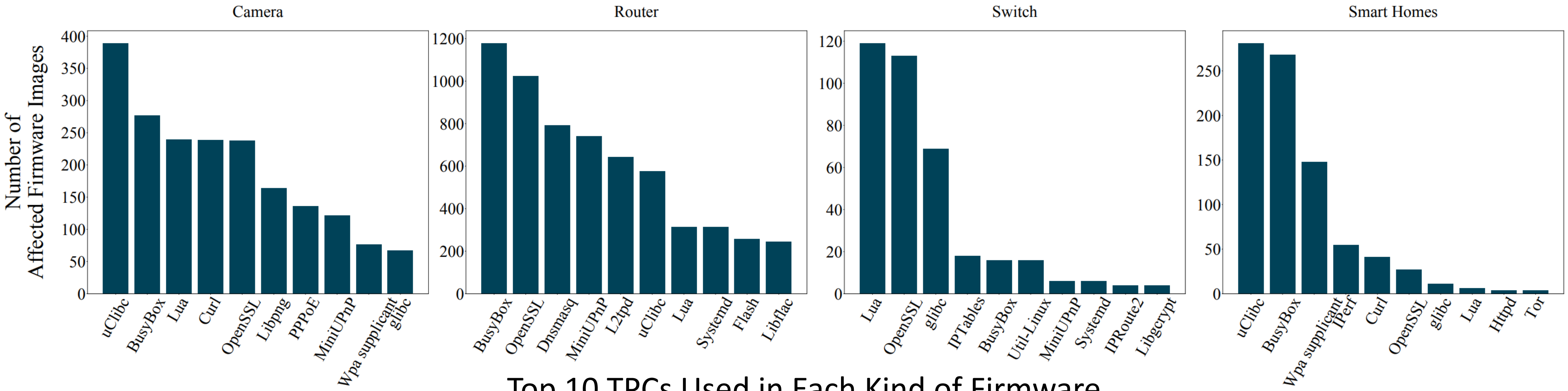
Vendor	Category	# Firmware	# TPC	# $\overline{TPC}$	# Vul.	# $\overline{Vul.}$
Xiongmai	Camera	520	232	0.45	313	0.60
Tomato-shibby	Router	230	2,088	9.08	11,948	51.95
Phicomm	Router	107	405	3.79	1,818	16.99
Fastcom	Router	149	274	1.83	1,849	12.41
	Unknown	10	0	0	0	0
Trendnet	Camera	477	136	0.28	1,395	4.32
	Router	336	1,762	5.24	7,903	23.52
	Switch	162	366	2.26	3,157	19.49
	Unknown	106	164	1.54	158	1.52
Xiaomi	Router	21	251	11.95	2,440	116.19
TP-Link	Camera	319	1,981	6.21	27,001	84.64
	Router	606	4,222	6.97	30,612	50.51
	Switch	484	77	0.16	795	1.64
	Unknown	48	67	1.40	639	13.31
D-Link	Camera	360	113	0.31	737	2.04
	Router	552	2,823	5.11	14,495	26.26
	Switch	545	80	0.15	1062	1.95
	Unknown	91	30	0.33	266	2.92
Hikvision	Camera	139	8	0.06	127	0.91
Foscam	Camera	113	0	0	0	0
Dahua	Camera	419	43	0.10	430	1.03
TSmart	Smart Homes	23,050	856	0.04	4,353	0.19
OpenWrt	Router	5,292	300,020	56.69	13,486	2.55

Analysis Results of the Dataset

TPC Usage

¤ Findings

- The same kind of firmware from different vendors adopts similar TPCs.
- Different kinds of firmware have commonalities in adopting TPCs.



Top 10 TPCs Used in Each Kind of Firmware

Introduced Vulnerabilities Overview

¤ **Results**

- **Detect a total of 128, 757 potential vulnerabilities, which involve 429 CVEs.**
- **88% of all the CVEs are caused by 10 CWE software weaknesses.**

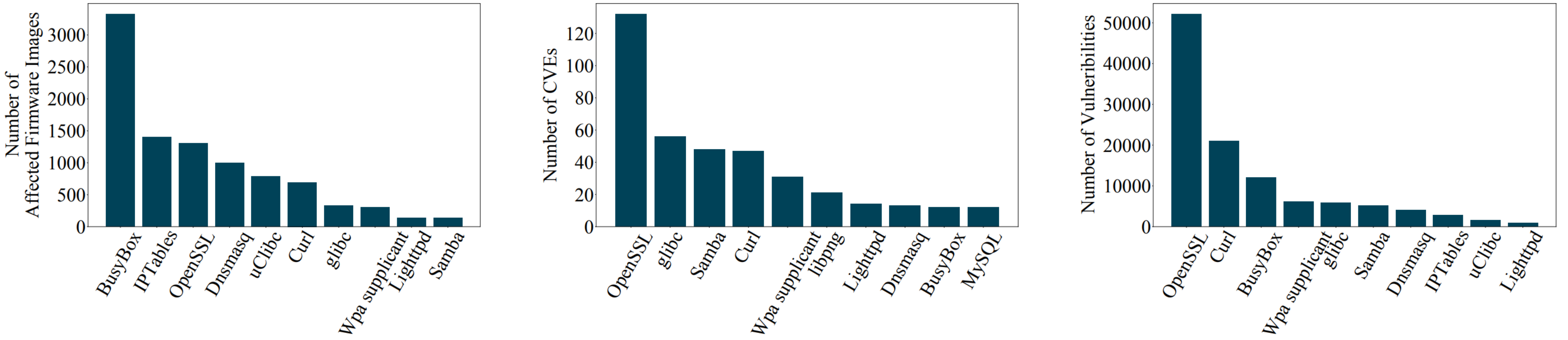
	CWE ID	Weakness	# CVEs
1.	399	Resource Management Error	87
2.	119	Buffer Overflow	84
3.	310	Cryptographic Issues	47
4.	20	Improper Input Validation	39
5.	264	Access Control Error	36
6.	200	Information Disclosure	31
7.	189	Numeric Errors	20
8.	-	Insufficient Information	18
9.	94	Code Injection	8
10.	362	Race Condition	7

Top 10 CWE Software Weaknesses

Introduced Vulnerabilities Overview

¤ **Findings**

- **Most of the vulnerabilities are concentrated on a few TPCs.**
- **386 CVEs from 10 TPCs, accounting for 90% of all the CVEs we detected.**



(a) Number of Affected Firmware Images by Top 10 TPCs.

(b) Number of CVEs Caused by Top 10 TPCs.

(c) Number of Vulnerabilities Caused by Top 10 TPCs.

Number of Affected Firmware Images, CVEs and Vulnerabilities Caused by the Top 10 TPCs.

Further Analysis

Firmware Vulnerability

RQ1: How vulnerable are firmware images of different kinds and from different vendors?

¤ **Findings**

- **Router is more vulnerable to attacks.**
- **Smart Homes have very few vulnerabilities.**

Category	*Vul.*	*Critical*	*High*	*Medium*	*Low*
Router	22.92	1.48	2.73	17.59	1.12
Camera	9.81	0.32	1.92	7.20	0.37
Switch	5.29	0.22	0.62	3.98	0.47
Smart Homes	0.19	0.01	0.05	0.11	0.02

Vulnerability of Different Kinds of Firmware

Firmware Vulnerability

RQ1: How vulnerable are firmware images of different kinds and from different vendors?

- **Findings**
  - **Xiaomi is in a very critical situation.**

Vendor	$\overline{Vul.}$	$\overline{Critical}$	$\overline{High}$	$\overline{Medium}$	$\overline{Low}$
Xiaomi	116.19	2.86	18.52	78.43	10.67
Tomato-shibby	51.95	2.77	8.46	35.49	1.84
TP-Link	39.20	1.37	6.97	28.59	2.26
Phicomm	16.99	0.41	3.88	11.28	0.54
D-link	11.87	0.55	1.95	8.03	1.34
Trendnet	11.02	0.29	1.85	7.98	0.90
Fastcom	9.13	0.44	1.35	6.81	0.53
OpenWrt	2.55	0.00	0.46	1.58	0.00
Dahua	1.03	0.03	0.14	0.66	0.16
Hikvision	0.91	0.05	0.17	0.60	0.03
Xiongmai	0.60	0.00	0.21	0.32	0.07
TSmart	0.19	0.01	0.05	0.11	0.02

Vulnerability of Firmware From Different Vendors

Firmware Vulnerability

RQ1: How vulnerable are firmware images of different kinds and from different vendors?

¤ **Findings**

- **Two vulnerabilities from OpenSSL and glibc have affected 604 firmware images which account for 1.8% of the dataset.**
- **380 firmware images from 8 vendors are vulnerable to the Heartbleed.**
- **224 firmware images from 4 vendors are vulnerable to the GHOST.**

Vendors	Heartbleed (CVE-2014-0160)	GHOST (CVE-2015-0235)
Fastcom	2	1
Trendnet	36	87
Tomato-shibby	24	-
TP-Link	301	91
D-Link	3	45
Hikvision	1	-
Dahua	5	-
TSmart	8	-

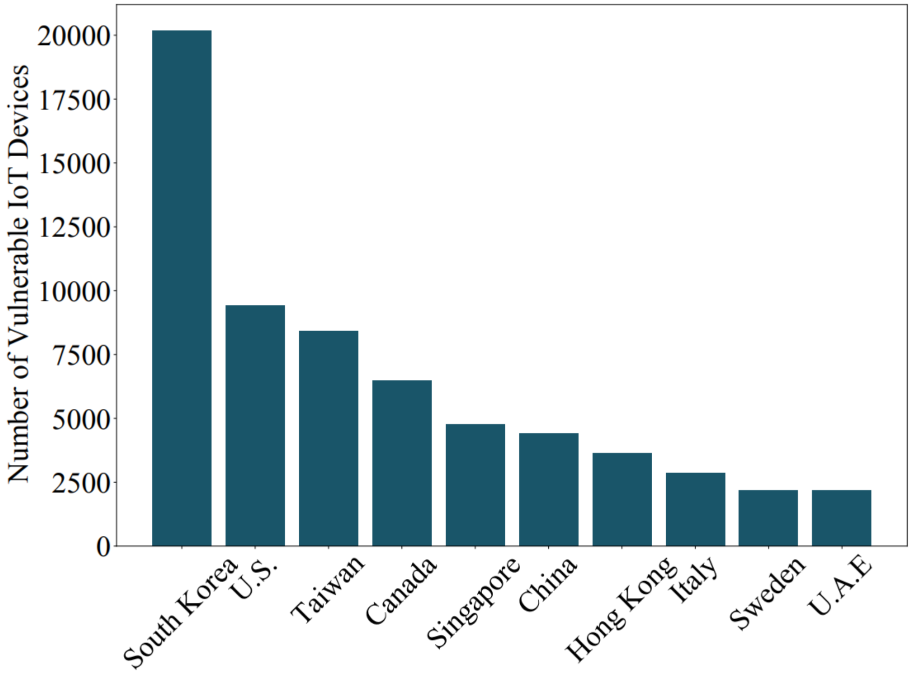
Firmware Affected by Two Vulnerabilities

Geographical Distribution

RQ2: What is the geographical distribution of the devices using vulnerable firmware?

¤ **Findings**

- **Six regions locate in Asia.**
- **South Korea contains the most vulnerable IoT devices.**
- **U.S. and Canada both have many vulnerable IoT devices.**
- **Europe contains relatively few vulnerable IoT devices.**



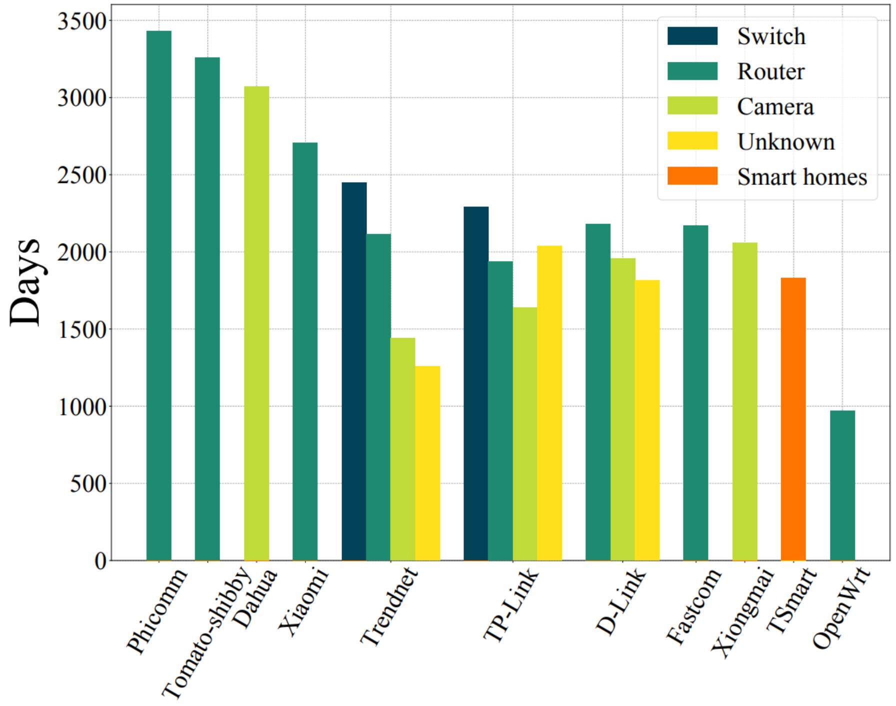
Top 10 Regions with the Most Vulnerable Devices

Delay Time of TPCs

RQ3: Does the firmware adopt the latest TPCs at the time when it was released?

¤ **Findings**

- **The average delay time of TPCs for all involved firmware images is 1948.2 days.**
- **Phicomm has the longest delay time, which reaches 3457.2 days.**
- **OpenWrt has the shortest delay time, which is less than two years.**



Delay Time of TPCs

License Violations

RQ4: Are there any TPC license violations?

¤ **Findings**

- **2, 478 commercial firmware images that have potentially violated GPL/AGPL licensing terms.**
- **4 vendors have provided distribution sites for downloading the source code of some GPL/AGPL licensed firmware.**

Vendors	# Firmware	Source Code Available
Xiongmai	195	✗
Phicomm	96	✗
Fastcom	17	✗
Trendnet	433	✓
Xiaomi	20	✗
TP-Link	847	✓
D-Link	487	✓
Hikvision	2	✓
Dahua	11	✗
TSmart	370	✗

Potential License Violations

Discussion

Discussion

- ## Mitigate Ethical Issues
  - **All firmware images are collected and treated legally.**
  - **Have reported all the vulnerabilities to vendors.**
  - **Open-source the dataset with a legal and ethical issues free plan.**
- ## Limitations and Future Work
  - **Collect more firmware images to extend the dataset.**
  - **Enrich the TPC database.**
  - **Adopt new techniques, e.g., fuzzing, to conduct a more in-depth analysis.**

Summary

Summary

- **The first scalable and automatic framework for analyzing the TPCs used in firmware and identifying the corresponding vulnerabilities.**
- **The first large-scale analysis of the vulnerable TPC problem in firmware. Identify 584 TPCs and detect 429 CVEs in 34,136 firmware images.**
- **Conduct further analysis from four different perspectives.**
- **https://github.com/BBge/FirmSecDataset**

Q&A

binbin.zhao@gatech.edu