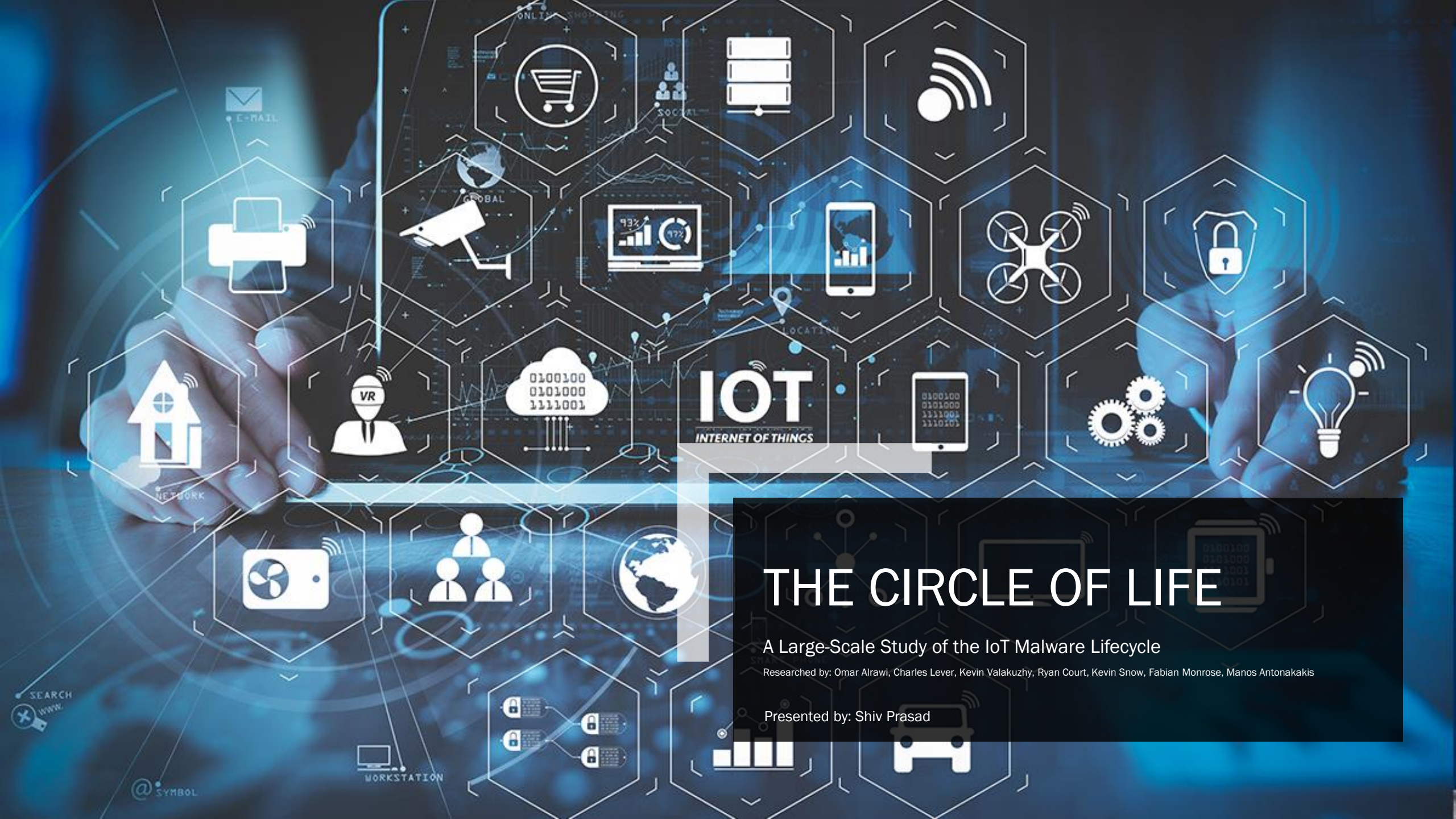


THE CIRCLE OF LIFE

A Large-Scale Study of the IoT Malware Lifecycle

Researched by: Omar Alrawi, Charles Lever, Kevin Valakuzhy, Ryan Court, Kevin Snow, Fabian Monrose, Manos Antonakakis

Presented by: Shiv Prasad

Overview

INTRODUCTION

SOLUTION

CRITICISM

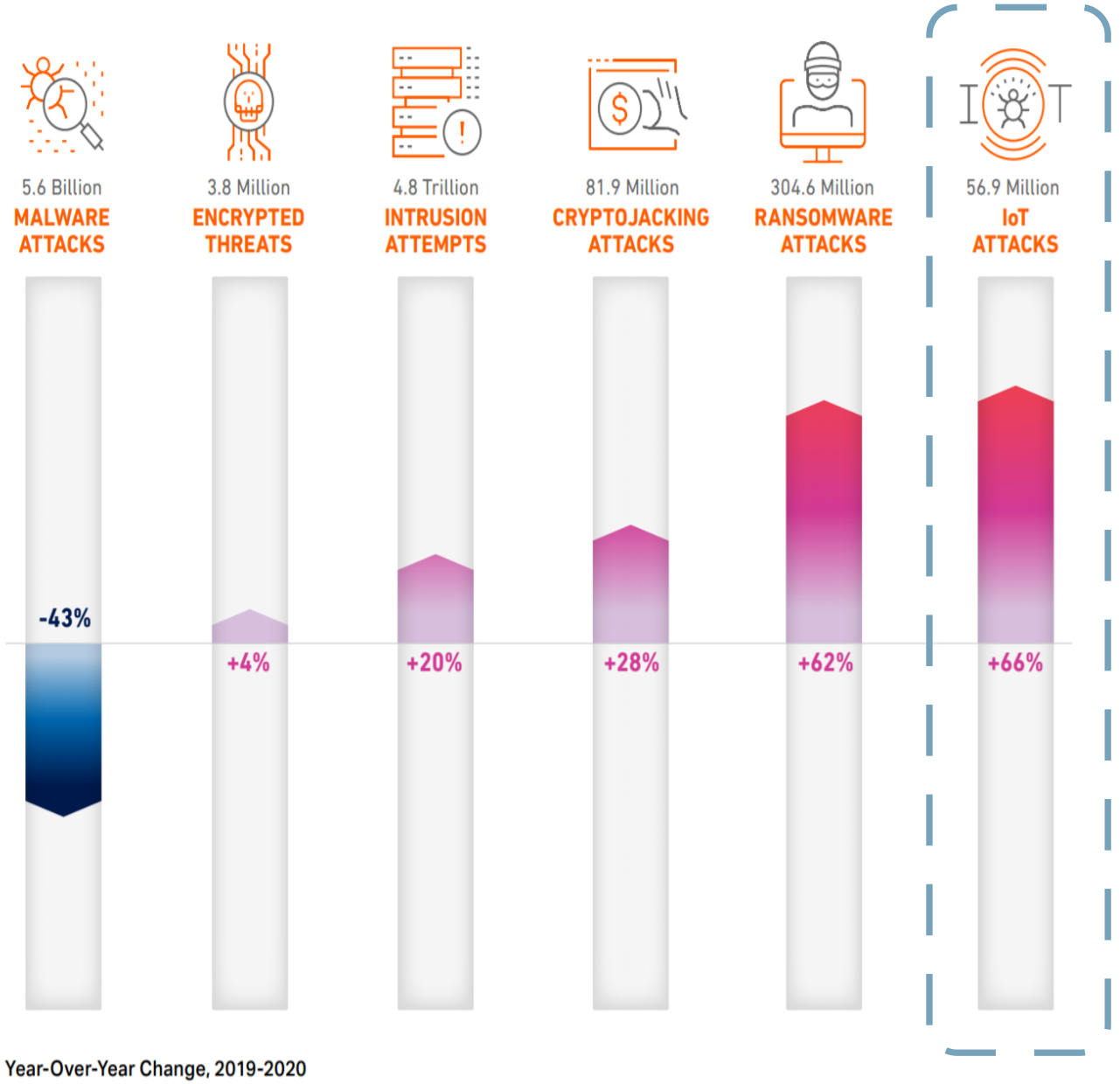
Scope

- ✓ IoT Malware Lifecycle and Defence Mechanisms
- ✓ Comparison of IoT and Traditional Malware Lifecycle
- ✗ History, Analysis and Evolution of Mirai
- ✗ Review of DDoS Attacks/Other Capabilities of the Malware

INTRODUCTION

Motivation

- Evolving cyber threat landscape shows that IoT attack activity continues to increase
- Mirai botnet attacks were largest DDoS attacks to date
- Mirai is malware that specifically targets IoT devices



Source: SonicWall 2021 Cyber Threat Report

Background

- 2008 – First reports of malware targeting embedded Linux-based systems
- Various other IoT malware families released over time, targeting specific devices/kernels
  - *Heterogeneity of devices*
  - *Relatively limited in their impact*
- Mirai was a step up and had a significant impact
  - *Runs on diverse set of devices, spreads efficiently and actively targets insecure IoT devices on the internet*
- Mirai source code publicly released
  - *Significantly reduced barrier to entry in carrying out IoT attacks, increased volume*
  - *Continuous evolution of variants with enhancements*

Problem

Do current defence mechanisms for traditional malware provide adequate protection and remediation capability against IoT malware attacks?

- Previous work on IoT malware focussed on single malware families or individual components of the lifecycle
- Two main research questions (RQ):
  - *RQ1: How is IoT malware different than traditional malware?*
  - *RQ2: Are current anti-malware techniques effective against IoT malware?*

- **SCOPE**

  Previous IoT malware studies have only focused on single families or devices

- **DATA SET**

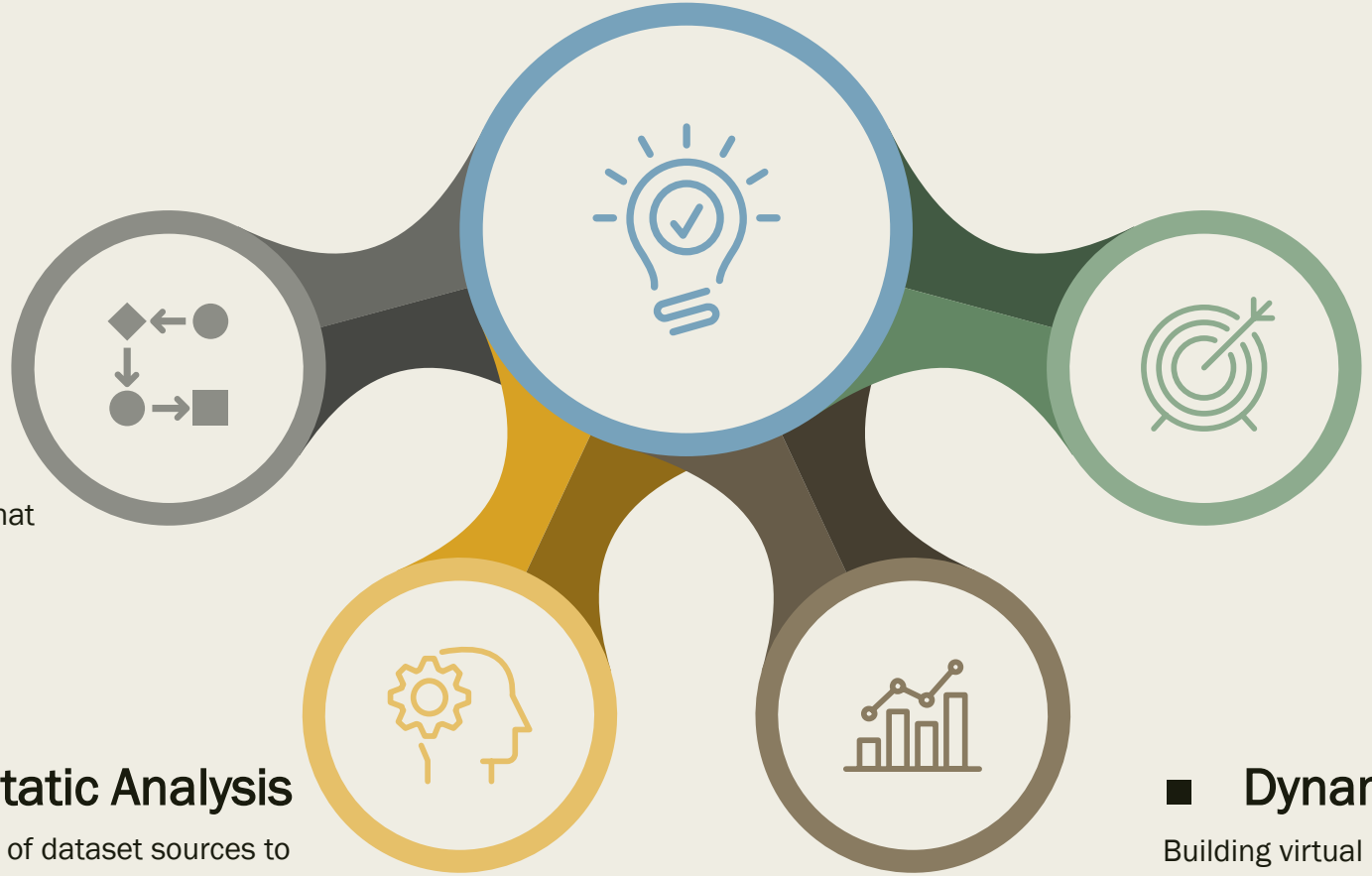
  Lack of large-scale measurements or samples over a meaningful duration of time

- **LIFECYCLE VIEW**

  Previous studies have focused on individual components of the lifecycle such as infection tactics, payload properties or malware capabilities

SOLUTION

Approach - Overview



Framework

A novel analysis framework that covers the lifecycle of IoT malware

Infrastructure Analysis

Identifying and filtering any benign domains identified by the static and dynamic analysis

Static Analysis

Analysis of dataset sources to identify target architecture, linking method, anti-analysis tactics, packing, embedded domain and IP addresses and infection vectors.

Dynamic Analysis

Building virtual machines and executing samples on them. This allows for the study of infection attempts, persistence methods, malware capabilities and C&C communication.

Framework

- **Infection Vector**

How the malware attacks a system

- **Payload**

The dropped malware code after exploitation

- **Persistence**

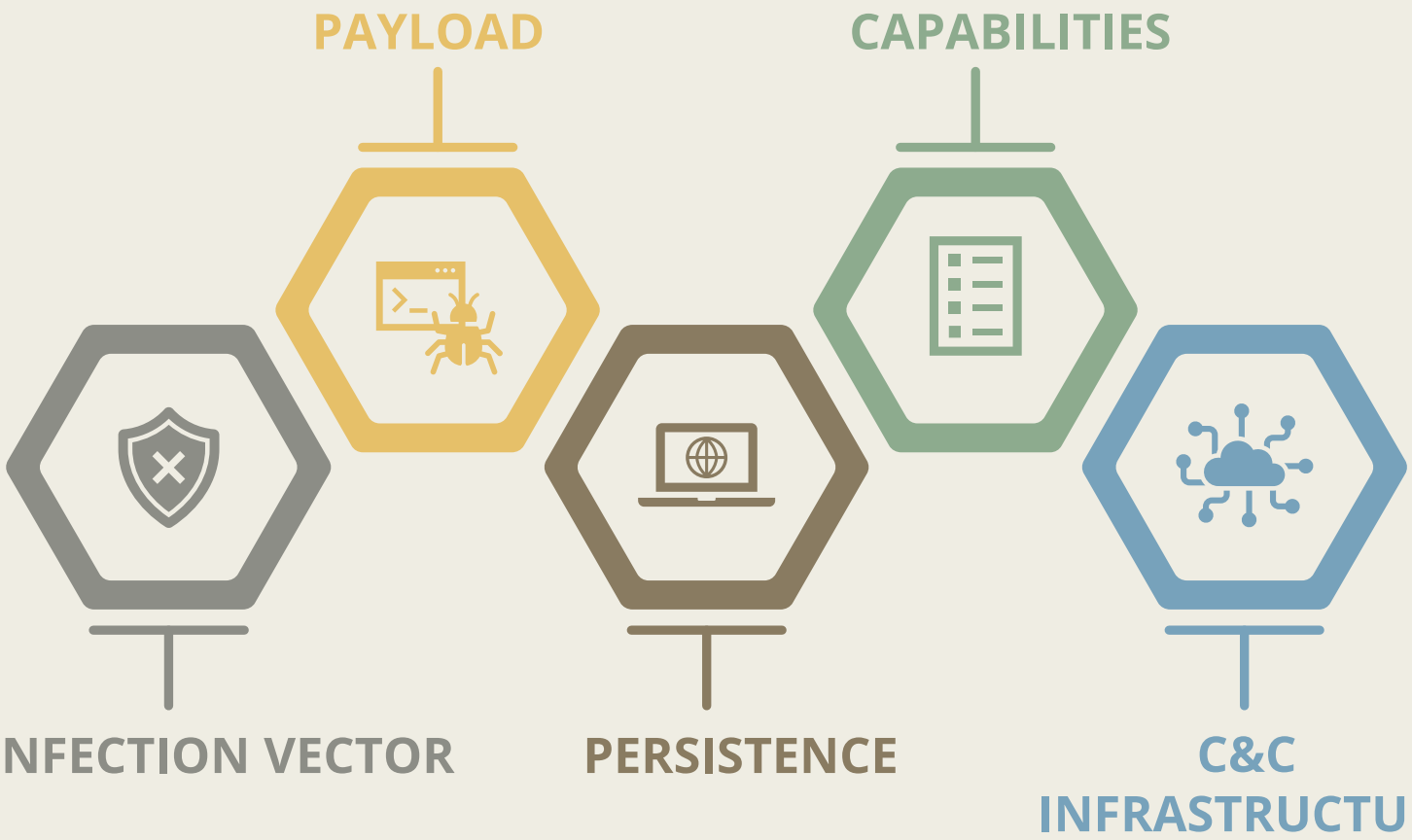
How the malware installs on a system

- **Capabilities**

The functions in the malware code

- **C&C Infrastructure**

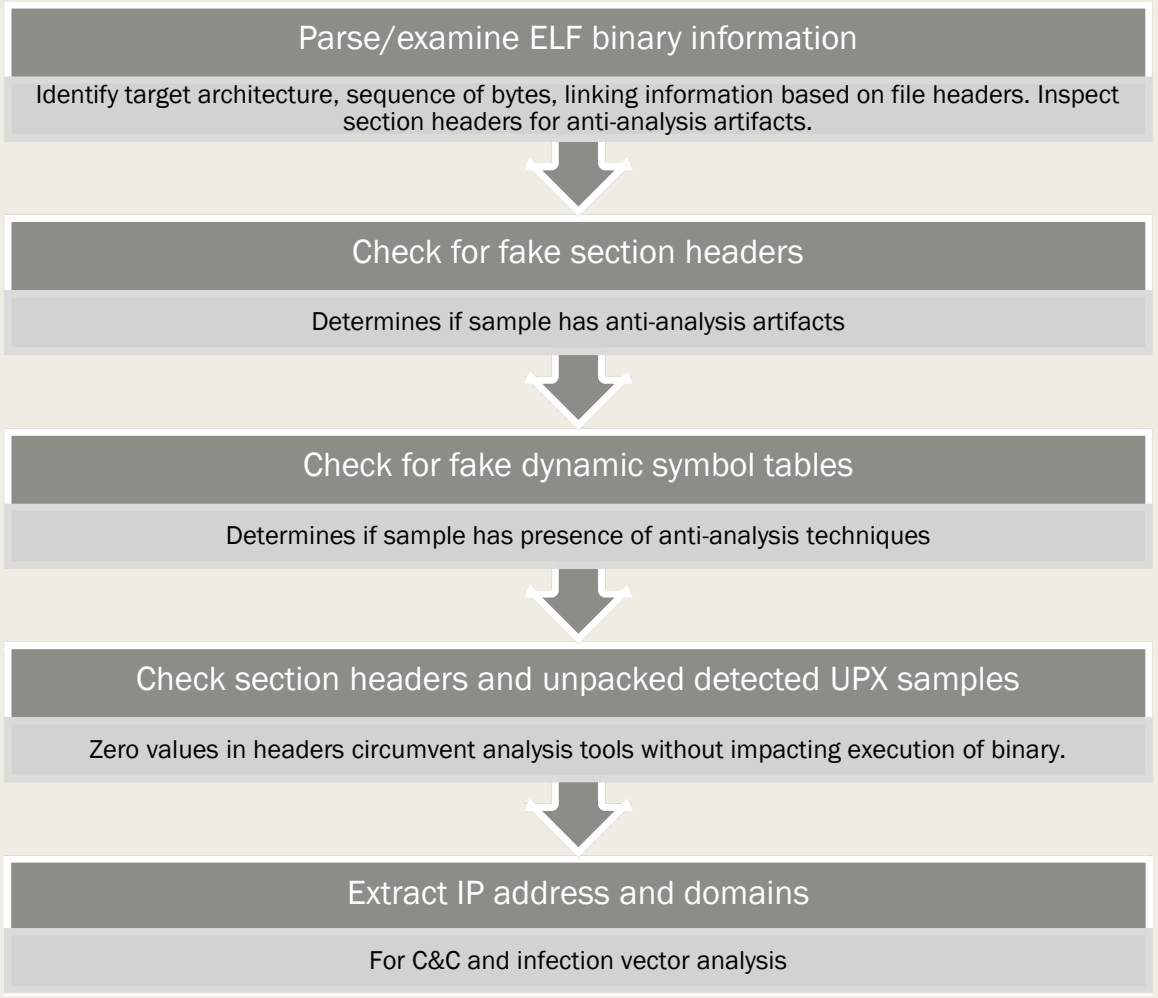
How the malware communicates



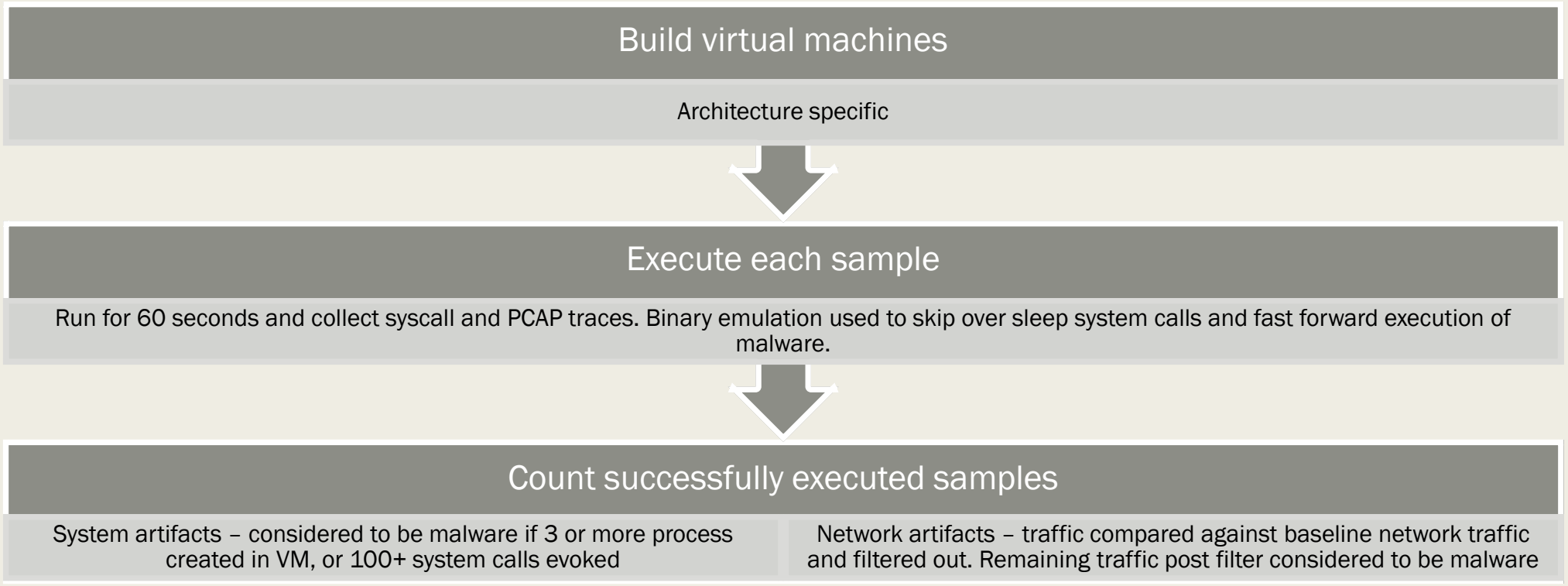
Static Analysis

- Datasets:

1. VirusTotal binaries. Main source of data for analysis. Filtered by:
   - *Malware targeting embedded IoT systems (Linux-based OS with RISC architecture)*
   - *Excludes non-ELF, CISC architecture (servers, desktops, laptops), android ELF, detected by AV*
2. DNS. To identify relationships between IP's and Domains and determine C&C activity.
   - *Active: ActiveDNSProject*
   - *Passive: anonymised data from ISP*
3. Honeypots. Insight into devices IoT malware targets.
   - *Bad Packets, aggregate from June 19*
4. Tranco Top Site Ranking. To filter out benign domains identified.



Dynamic & Infrastructure Analysis



- Infrastructure Analysis – 3 tiered process to check against benign domains, then inspect list to remove benign domain. Finally, a bipartite graph between domains and IPs to find connected components and filter benign clusters.

Results – Key Findings

Framework Component	Key Takeaway(s)
Infection Vector	**Most Common Infection Vectors** Default and/or hard coded credentials and exploitation of vulnerabilities. However, findings suggest that IoT malware has evolved and now has the ability to indiscriminately target many diverse IoT device types. **User Interaction** While desktop/laptop attacks target end users via vectors such as phishing or inadvertently downloading malicious software, IoT devices are headless and lack a GUI. Hence, user interaction is not required for IoT malware infection. This feature, in combination with IoT malware being architecture agnostic, enables rapid infection of devices.
Payload Analysis	**Detection** IoT malware uses polymorphism and anti-analysis mechanisms to evade signature-based detection. **Target** System shell interface is the primary component for infection.

Results – Key Findings

Framework Component	Key Takeaway(s)
Persistence	**Persistence Methods** Malware is able to persist in many locations and there are many methods to overcome the read-only mount of the file system. This includes attempts to install as a service, startup script, system module or backdoor.
Capability Analysis	**Capabilities** Initial variants focussed on DDoS and scanning capabilities. Modern IoT malware is evolving to include capabilities such as evasion, privilege escalation, data theft and damage to the device and network.
C&C Analysis	IoT malware can use P2P and centralised infrastructure for C&C communication. Additionally, IoT malware rely mostly on hard-coded IP addresses for C&C call-back rather than DNS lookup. Hence, network detection of malware communication can prove to be difficult with P2P channels and evasive DNS resolutions. However, the use of hard-coded IP addresses make IoT botnets less resilient to take downs.

Results – Key Findings

Framework Component	Key Takeaway(s)
Not related to a framework component	**Detection and Labelling** Given that no host-based intrusion detection systems (HIDS) run on IoT devices, detecting malware after an infection is not possible. However, signature-based scanners can detect suspicious binaries forensically captured from the network or the device. Findings suggest that many AV scanners lack support or have limited signature coverage for IoT malware in the wild.

Results – Overview

- IoT findings derived from previously discussed analysis
- Desktop and mobile findings derived from systemisation of 25 prior studies on traditional malware
- Results framed in the context of the novel framework developed, answer to RQ1:
  - *RQ1: How is IoT malware different than traditional malware?*

Table 1: An overview of the results from our findings comparing desktop, mobile, and IoT malware using the proposed framework.

Components		Summary			
	Categories	Desktop	Mobile	IoT	Definition for each component's subcategories
Infection	Remote Exploit	✓		✓	**Remote Exploit** refers to exploiting a service or an application running on a device.
	Repackaging	✓*	✓		**Repackaging** refers to benign application repackaged with malware (i.e. pirated software).
	Drive-by	✓	✓		**Drive-by** refers to infection by redirecting the system to a malicious resource.
	Phishing	✓	✓		**Phishing** refers to social engineering attacks that trick a user into getting infected.
	Default Cred.	✓*		✓	**Default Credentials** refers to the use of vendor default credentials for device access.
	Rem. Media	✓*	✓		**Removable Media** refers to the use of USB for infection between devices.
Payload	Packing	✓	✓	✓	**Packing** refers to the use of packers or polymorphic techniques for obfuscation.
	Env. Keying	✓	✓	✓	**Env. Keying** refers to the dependence on the target's environment artifact (i.e. HW id).
	Scripting	✓*		✓	**Scripting** refers to the use of a scripting interpreter (i.e. Powershell, sh, etc.).
	Cross-Arch/Plat.	✓*	✓	✓	**Cross-Arch/Plat.** refers to using payloads for different architectures (x86, ARM, etc.) or platforms (Windows, Android, etc.).
Persist.	Firmware	✓		✓	**Firmware** refers to persisting by modifying the device's firmware.
	OS - Kernel	✓	✓	+	**OS - Kernel** refers to persisting as a kernel module.
	OS - User	✓	✓	+	**OS - User** refers to persisting in user-space through configuration or process/service.
Capability	Priv. Escalation	✓	✓	✓	**Priv. Escalation** refers to exploiting OS vulnerability to elevate privilege on a device.
	Defense Evasion	✓	✓	✓	**Defense Evasion** refers to actively avoiding or disabling security features on the device.
	Info. Theft	✓	✓	✓	**Info. Theft** refers to profiling and exfiltrating sensitive information from the device.
	Scanning	✓		✓	**Scanning** refers to using the device to scan for other devices.
	DDoS	✓		✓	**DDoS** refers to using the infected device to orchestrate a DDoS attack.
	Destruction	✓	✓	✓	**Destruction** refers to actively destroying or ransoming the device.
	Resource Abuse	✓	✓	✓	**Resource Abuse** refers to using the device to run unauthorized services or applications.
C&C	Peer-2-Peer	✓		✓	**Peer-2-Peer** refers to using peer-2-peer network protocol for managing the botnet.
	Centralized	✓	✓	✓	**Centralized** refers to using a central *C&C* server for managing the botnet.
	Email/SMS	✓	✓		**Email/SMS** refers to using email or short message service for call-back to the bot master.

∗ Techniques documented by security companies. + Unified software layer that integrates OS and firmware.

RQ1: Similarities and Differences

RQ1: How is IoT malware different than traditional malware?

Feature	IoT Malware	Traditional Malware
Code base	Majority based on Mirai code base with minor variants.	Diverse code base with many different malware families.
Evading detection	Polymorphism and anti-analysis capabilities to evade signature based detection	
Infection categories and threat posed	2 predominant infection vectors – remote exploitation and use of default credentials. However, higher threat as attack surface is much larger due to malware attacking a larger set of architecture agnostic, internet-facing devices.	6+ infection vector categories. Relative to IoT malware, threat is lower as malware families target specific architectures or operating systems.
Reliance on system shell	Yes – disabling or limiting can be used as a mitigation.	No reliance on system shell.

RQ1: Similarities and Differences

RQ1: How is IoT malware different than traditional malware?

Feature	IoT Malware	Traditional Malware
File system constraints	Yes – IoT mounts file system as read only. IoT malware has to install as a service, startup script, system module or backdoor to establish persistence.	No file system constraints.
Layered protection	No – unification of user-space, kernel-space and firmware allows IoT malware to have privileged access to device hardware.	Yes – traditional malware needs to contend with separation of user-space, kernel-space and firmware.
Malware capability	Limited capability relative to traditional malware, but increasingly sophisticated attacks are on the rise. Likelihood of tailored IoT device targeting for specific attacks in future.	Full spectrum of malware capability.

RQ1: Similarities and Differences

RQ1: How is IoT malware different than traditional malware?

Feature	IoT Malware	Traditional Malware
Persistence	Limited persistence, however ability to get privileged access to device hardware can lead to more stealthier persistence tactics.	All levels of persistence, ability to persist at many levels from user-space to firmware and outside visibility of detection tools.
C&C capability	Use of P2P and centralised control infrastructure. Reliance on multiple payload domains to be registered, however these are quickly detected and blocked, but still enable the botnet to spread.	Utilise all methods of C&C capability. Greater scalability and resiliency of infrastructure by organising into specific topologies or incorporating pseudo-randomness in domains.

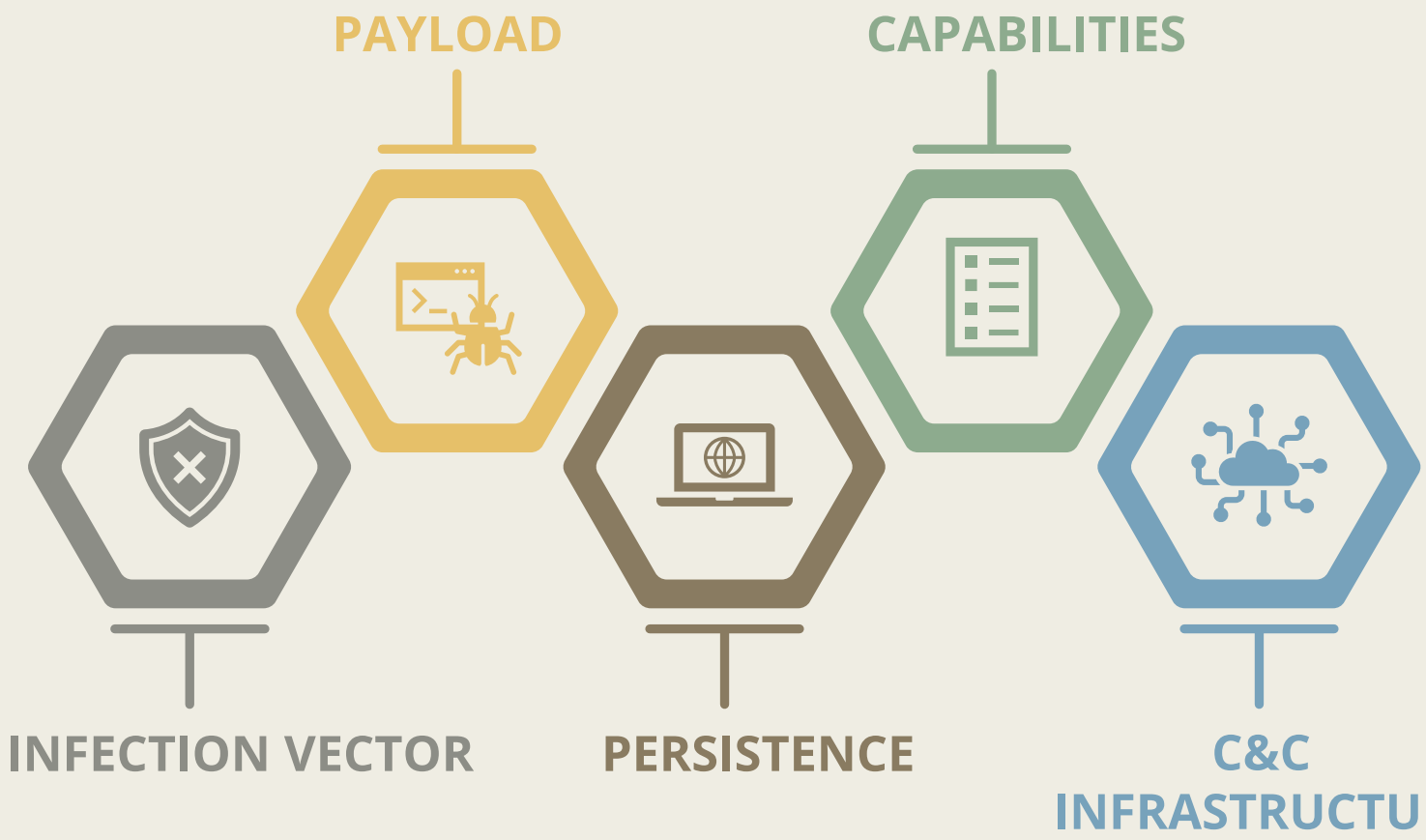
RQ2: Stakeholders and Defences

RQ2: Are current anti-malware techniques effective against IoT malware?

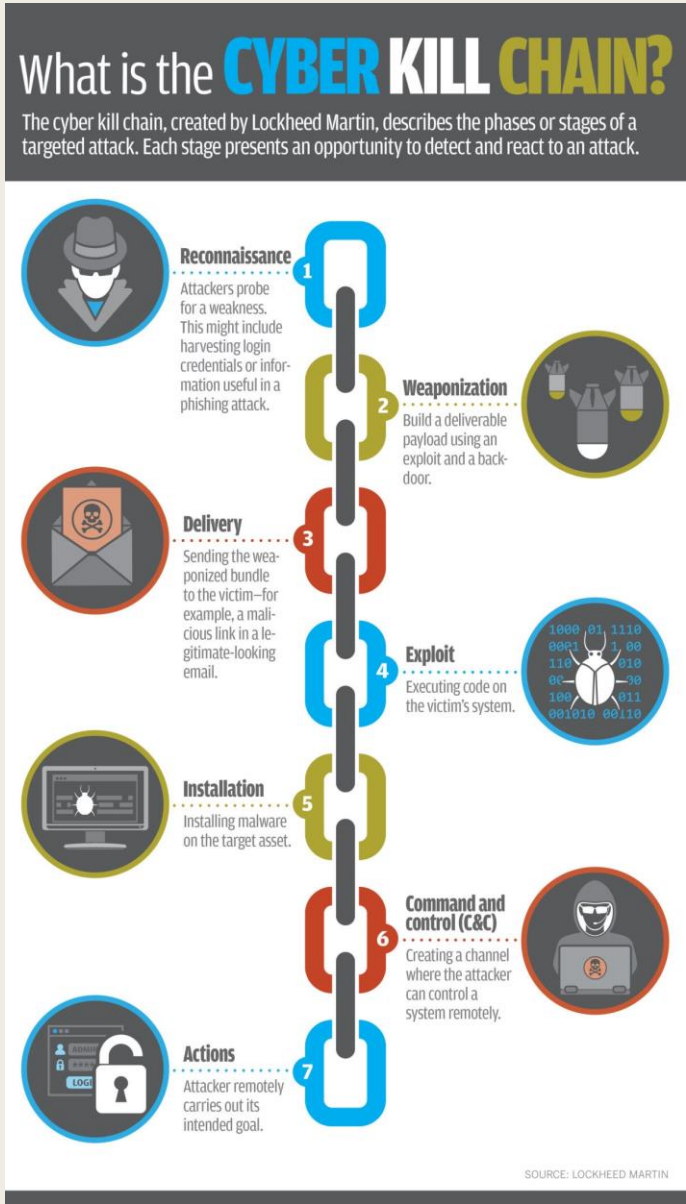
Device Owners	Device Vendors	ISP Operators
Disable internet facing services	Telemetry to detect anomalies	Creating walled gardens for infected customers
Change default credentials	Limit shell interaction	IP blocking or redirection for known IoT C&C or payload servers
Segment network	Limit cross-process interaction via containerisation	Intercepting malicious payloads through continuous monitoring
Reboot or reimage device	Process whitelisting to only allow trusted processes	
	Remote attestation to guarantee a clean state	
	Client-server design to limit exposed services	

CRITICISM

Criticism 1: Novel Framework



VS



Criticism 2: RQ2 Coverage

- RQ2 considered:
  - *Are current anti-malware techniques effective against IoT malware?*
- The findings and discussion were relatively light
- Focussed on what different stakeholders ***can do*** as opposed to an evaluation or analysis of current anti-malware techniques
- Provided little insight into existing security solutions or consideration for a defence in depth approach to mitigating risk

Criticism 3: Focus on signature/rule based solutions

- The analysis covered (well-known) limitations of signature or rule based tools, which are reactive in nature
- This is an inaccurate representation of the current state of techniques available
- AI and Machine Learning based security solutions have been prevalent in industry for many years, and are extremely effective and well advanced
- These can work without any prior knowledge of the devices, their supplier, or patch history, and without using malware signatures or indicators of compromise
- This includes products from vendors such as Forescout, Darktrace, Vectra, Nozomi, Dragos, etc.

Criticism 4: Lifecycle of malware without consideration of device lifecycle

- This work looks at the malware lifecycle, and makes a number of recommendations for device owners and vendors
- A majority of these can be implemented via a device management platform which allow for devices to be:
  - *Provisioned*
  - *Deployed*
  - *Monitored*
  - *Maintained & Updated*
  - *Decommissioned*

Device Owners	Device Vendors
Disable internet facing services	Telemetry to detect anomalies
Change default credentials	Limit shell interaction
Segment network	Limit cross-process interaction via containerisation
Reboot or reimage device	Process whitelisting to only allow trusted processes
	Remote attestation to guarantee a clean state
	Client-server design to limit exposed services