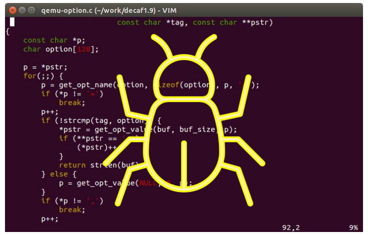
Malware
Operation, Detection, and Evasion

Chester Rebeiro

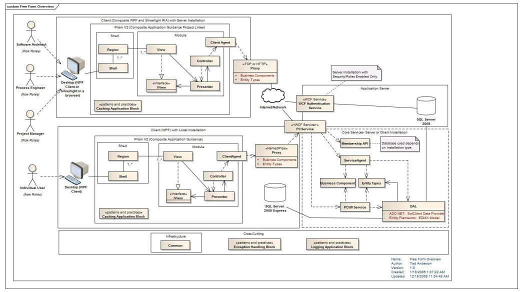
IIT Madras

What we have so far?

Social Engineering



Program Bug



Design Flaw

CR

What next?

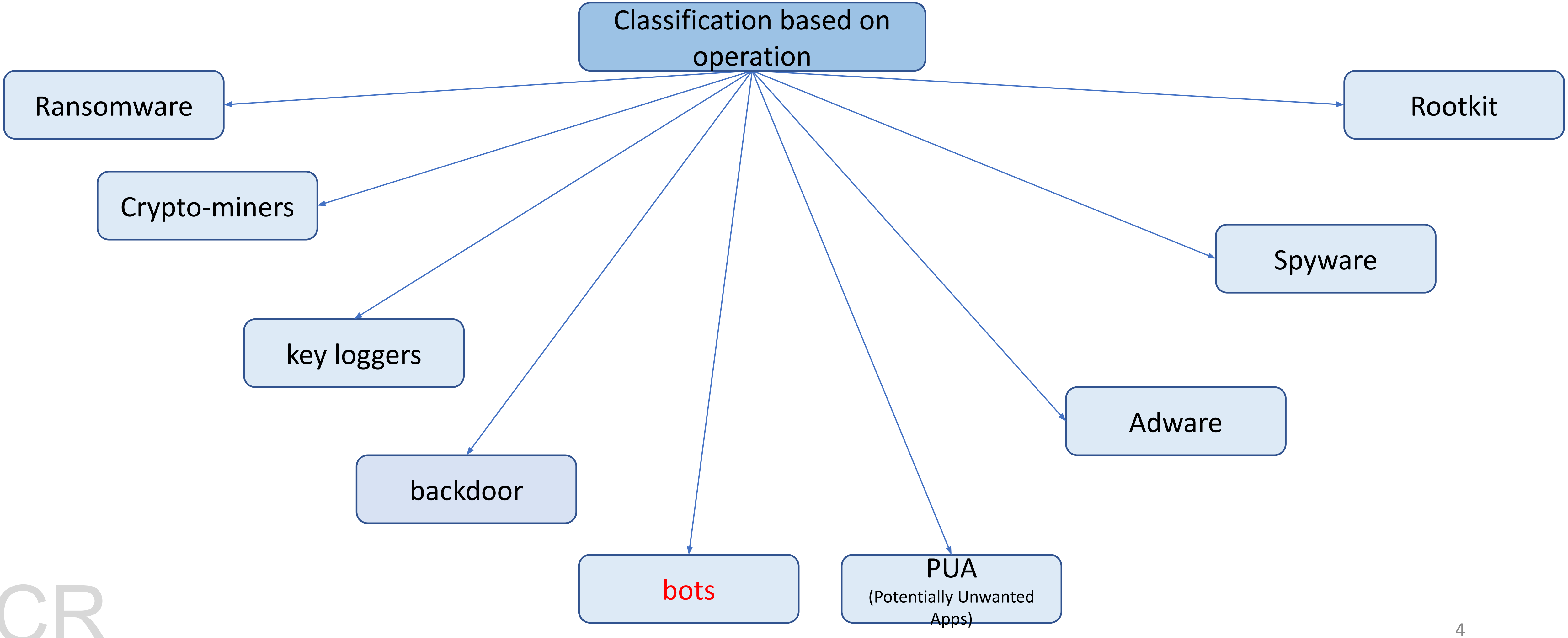
- Depends on the type of malware

Classification based on operation

Classification based on spread

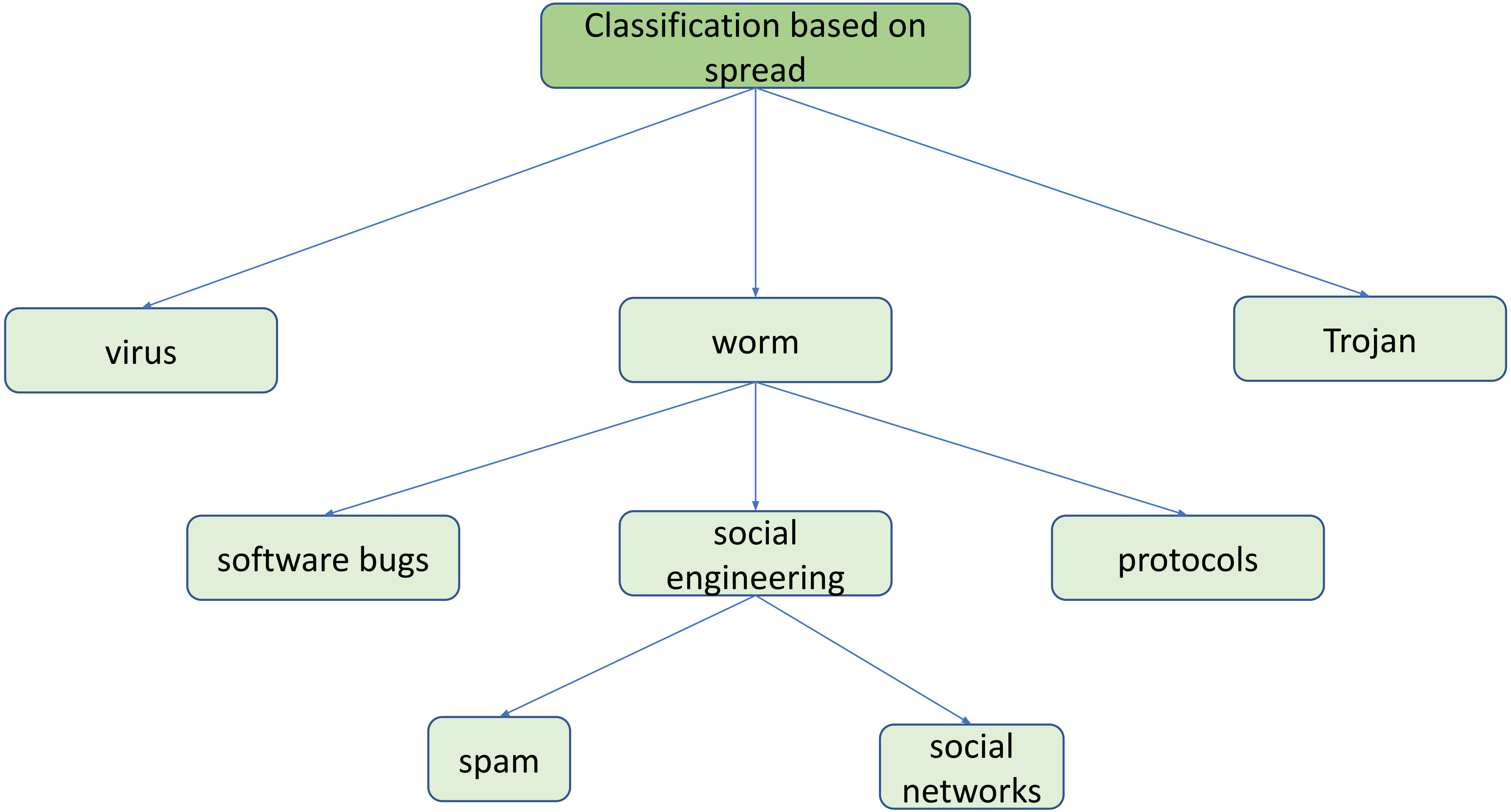
CR

Classification based on operation



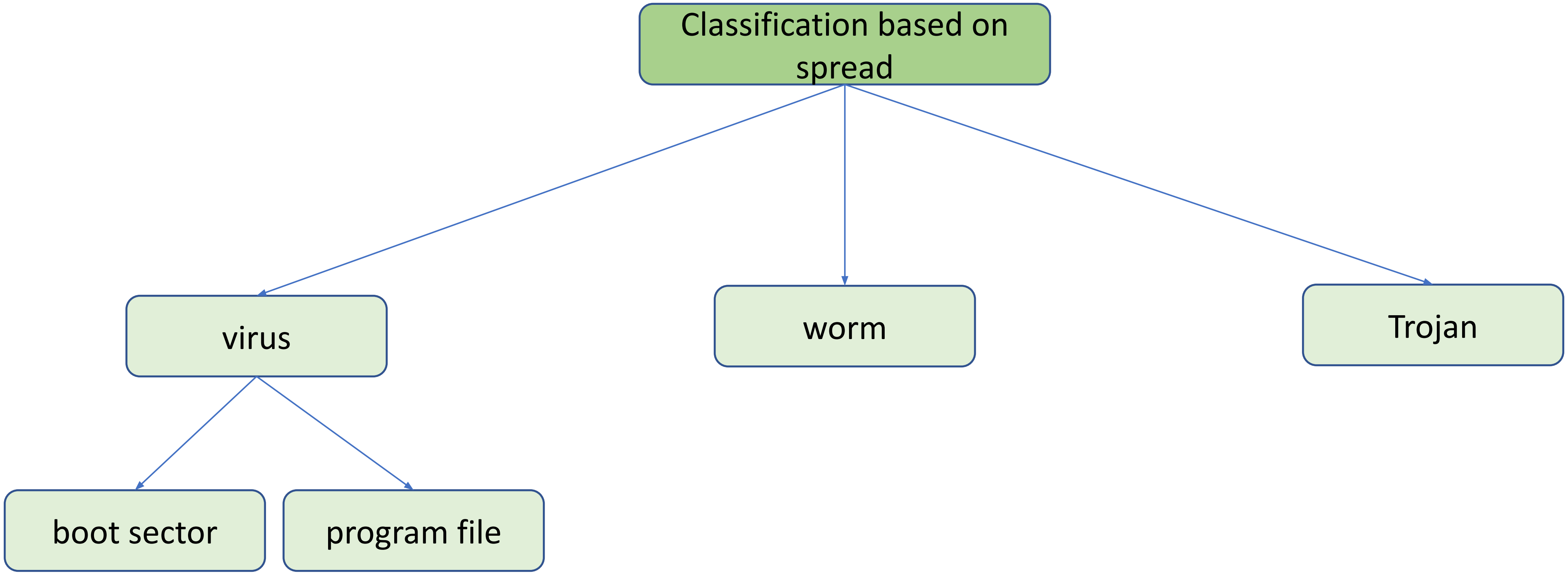
CR

Classification based on spread



CR

Classification based on spread



CR

Bots

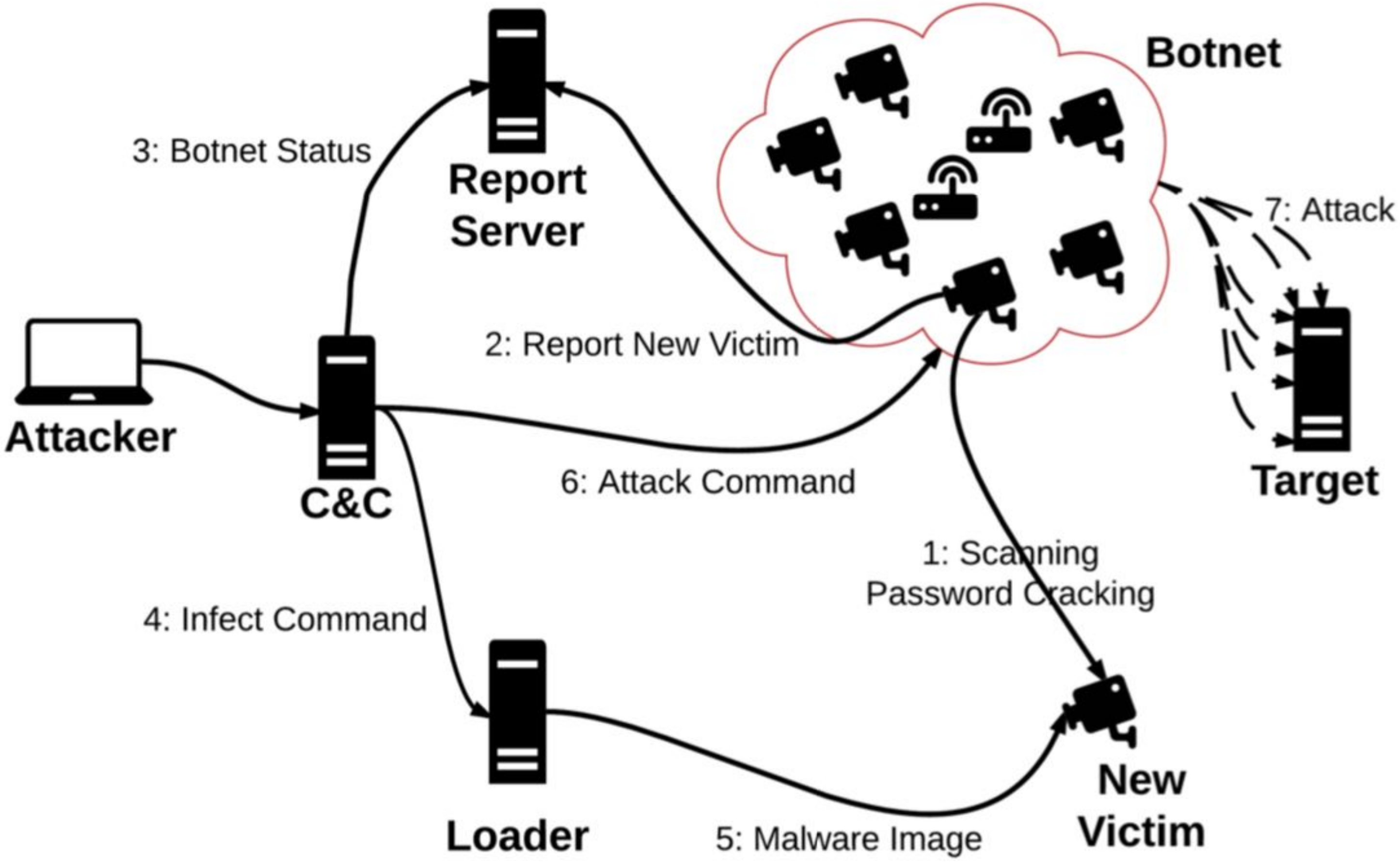
Mirai Botnet Malware

- Number of IoT devices increasing at a rapid rate
- These devices are characterized by
  - Low profile
  - Less user interactions
  - Security often compromised (for better performance / smaller profile)
  - Not always up-to-date with security patches
- Malware with IoT devices as targets
  - Bashlight and Mirai are the most popular
  - PNScan, targets x86 platforms.
    - Try to determine router login baed on a special dictionary
    - Connect using ssh connection using predefined user credentials

Low hanging fruit for hackers

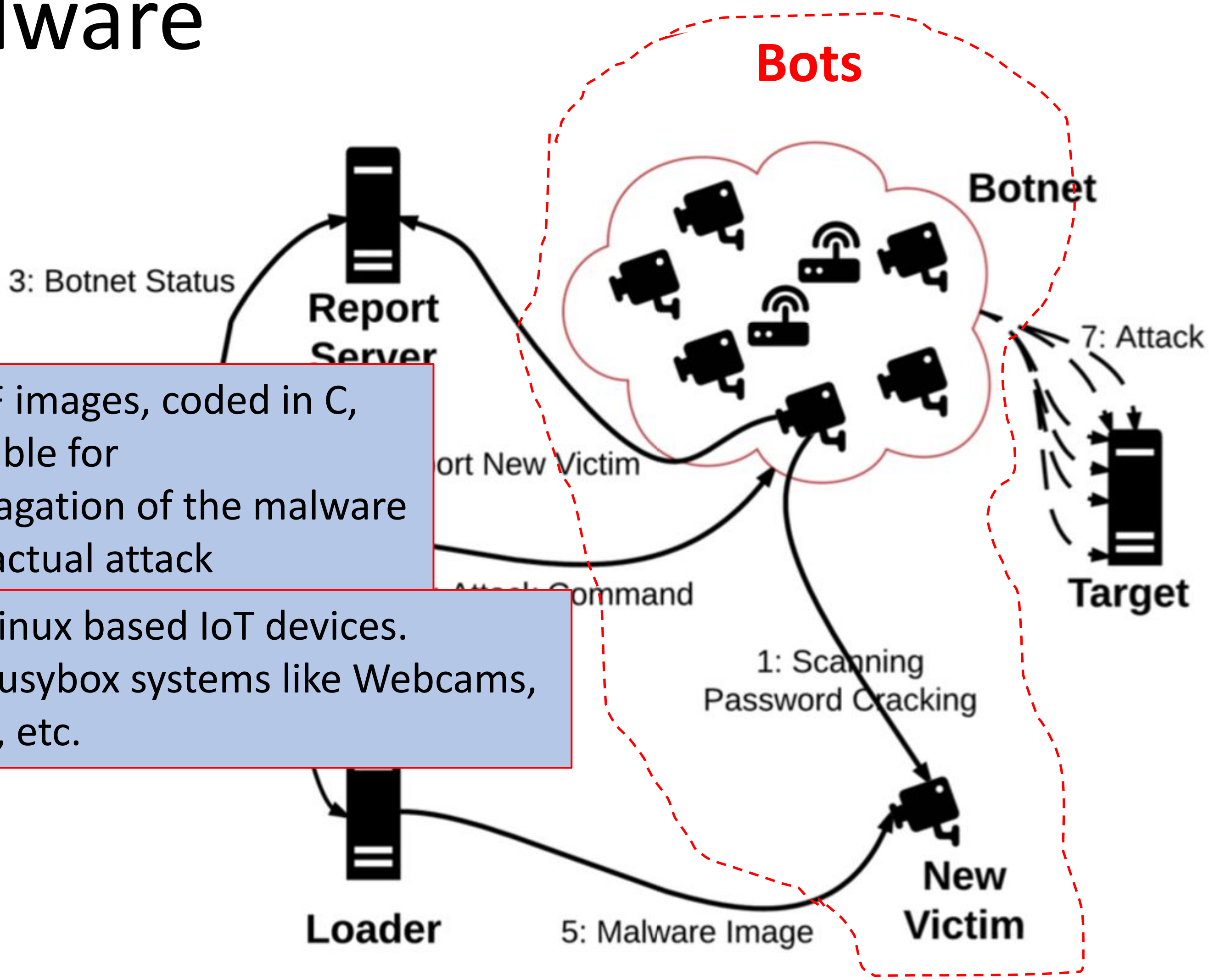
Mirai BotNet Malware

280 Gbps max flooding
50,000 unique Ips
164 countries



The Mirai Botnet and the IoT Zombie Armies, 2017

Mirai BotNet Malware



Bots: ELF images, coded in C,
Responsible for

(1) Propagation of the malware
(2) The actual attack

Targets Linux based IoT devices.
Mostly busybox systems like Webcams, Cameras, etc.

The Mirai Botnet and the IoT Zombie Armies, 2017

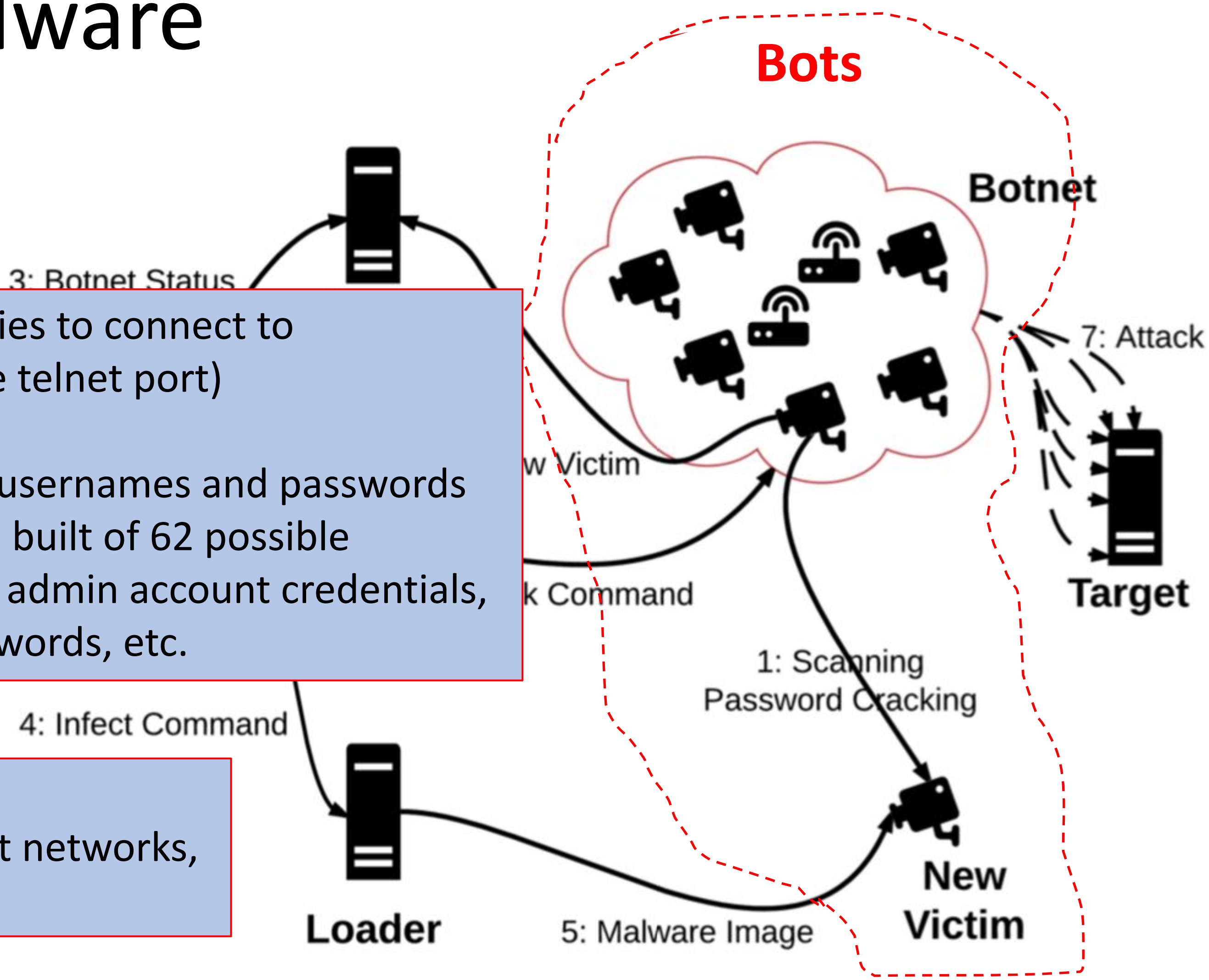
Mirai BotNet Malware

Infiltration

Every bot generates random IPs and tries to connect to Port 23 (telnet) or port 2323 (alternate telnet port)

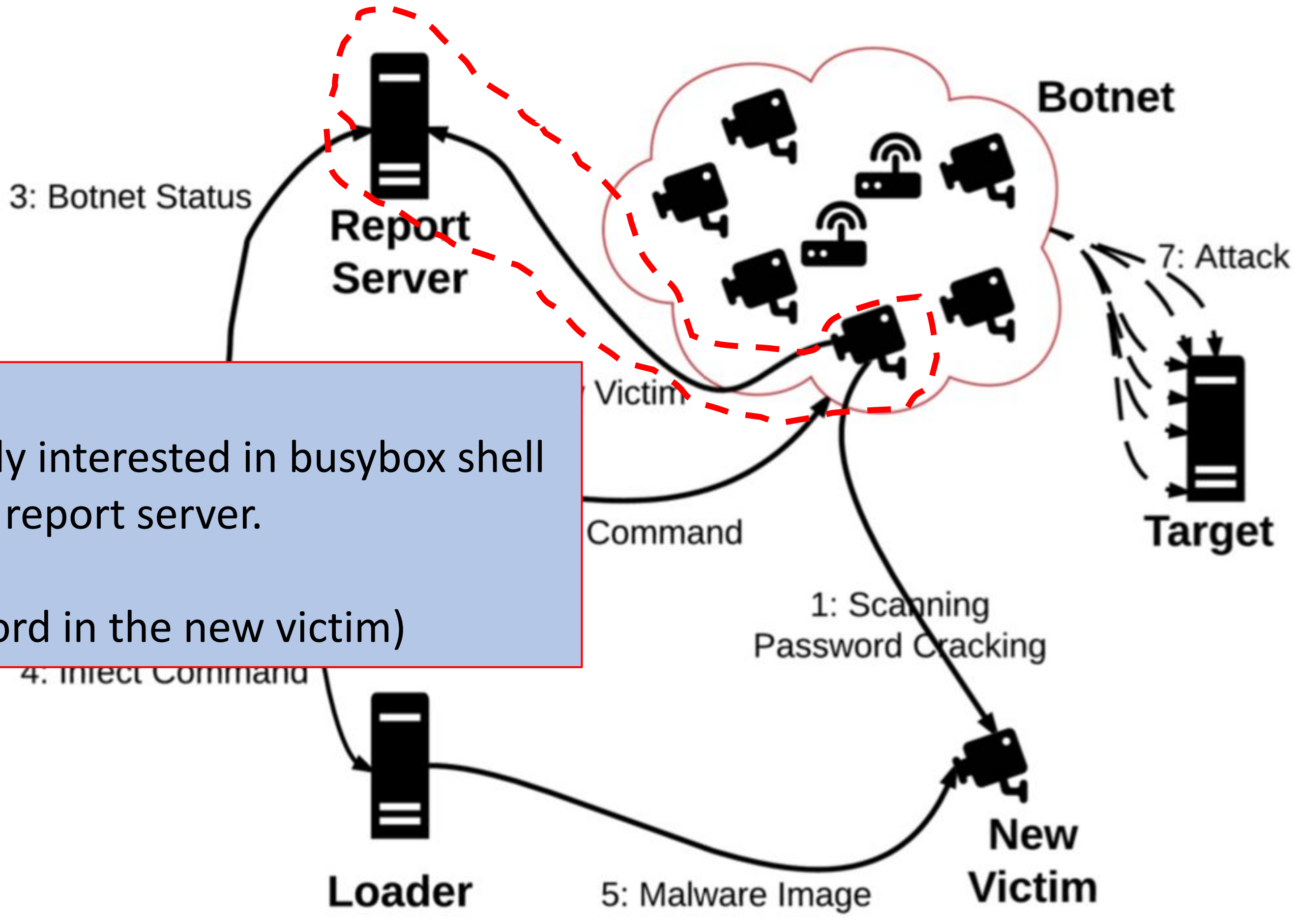
Brute force dictionary search for valid usernames and passwords that will permit login. The dictionary is built of 62 possible username / passwords. These include, admin account credentials, debug logins, usernames with no passwords, etc.

Some IP addresses are blacklisted.
Loopback, internal networks, multicast networks, US postal servcie, DoD, GE, HP, IANA,



The Mirai Botnet and the IoT Zombie Armies, 2017

Mirai BotNet Malware



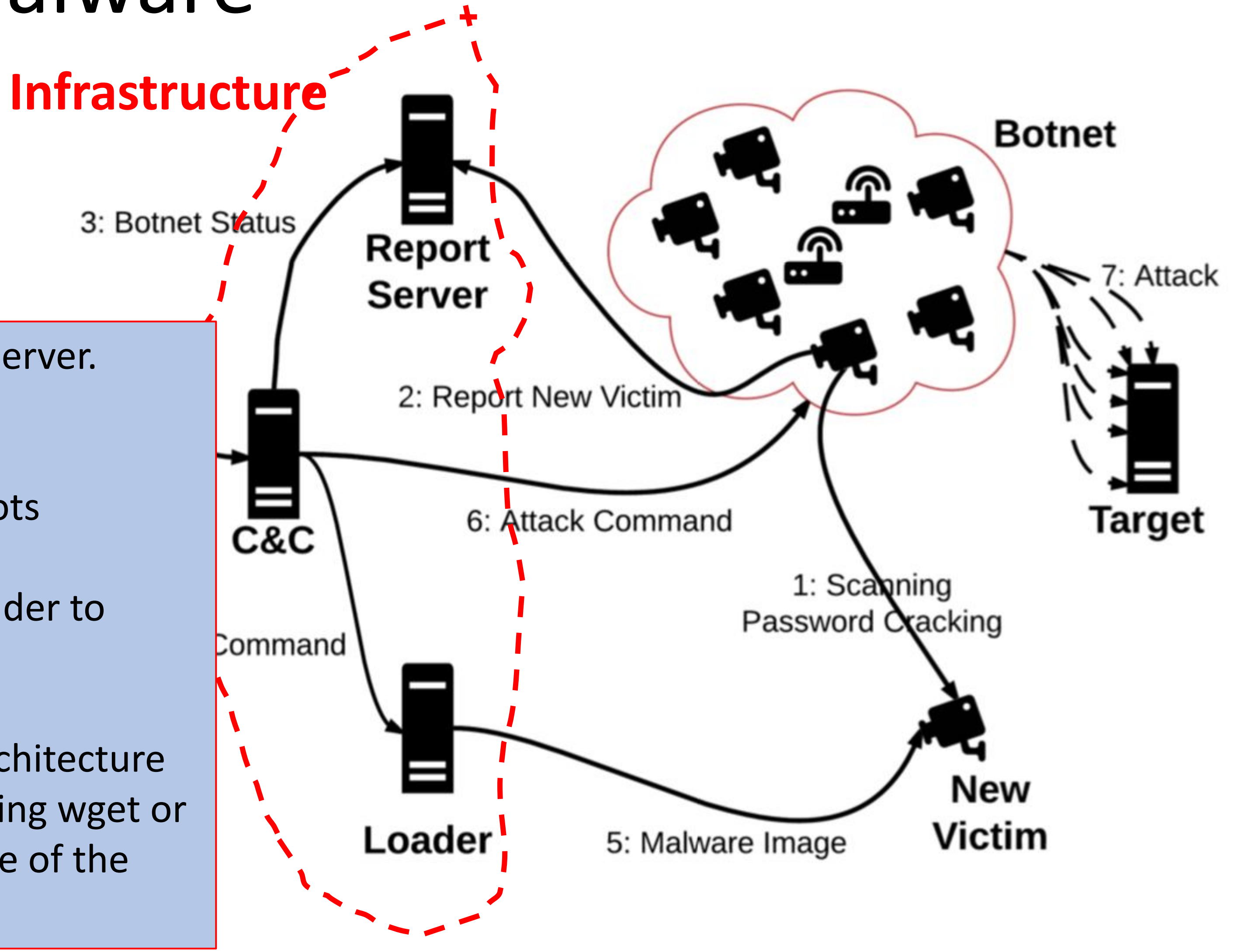
On a successful login:

- Try to establish a shell. Especially interested in busybox shell
- Report a successful login to the report server.

(does not try to change the password in the new victim)

The Mirai Botnet and the IoT Zombie Armies, 2017

Mirai BotNet Malware



Command and Control. Management server. Implemented in Go.

At any time, it can get a list of active bots from the report server.
It can, also, at anytime, instruct the loader to load malware into the bot.

Loader, depending on the hardware architecture of the bot, instructs it to download (using wget or tftp) and execute required binary image of the malware.

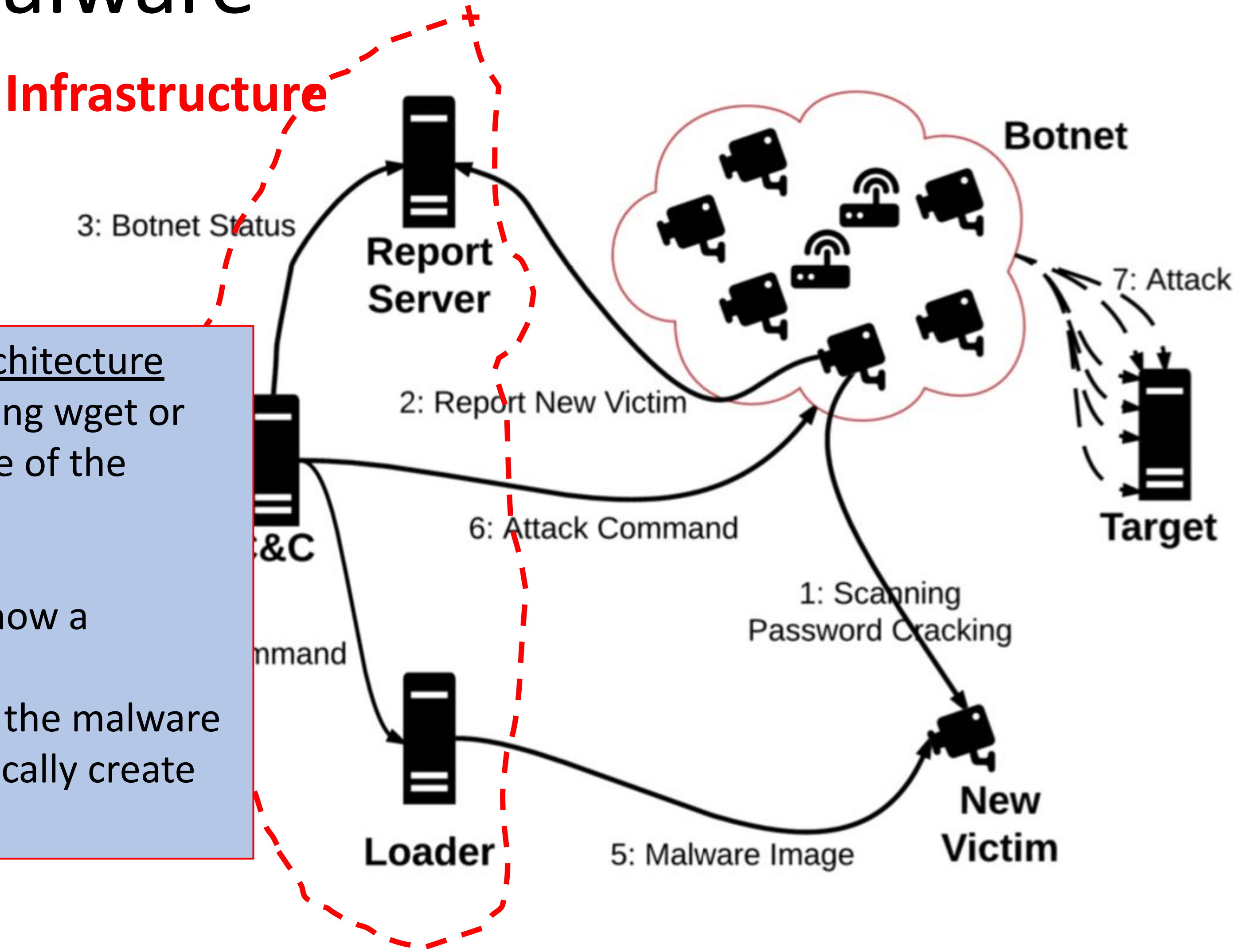
The Mirai Botnet and the IoT Zombie Armies, 2017

Mirai BotNet Malware

18 hardware variants supported including ARM, MIPS, x86, SPARC

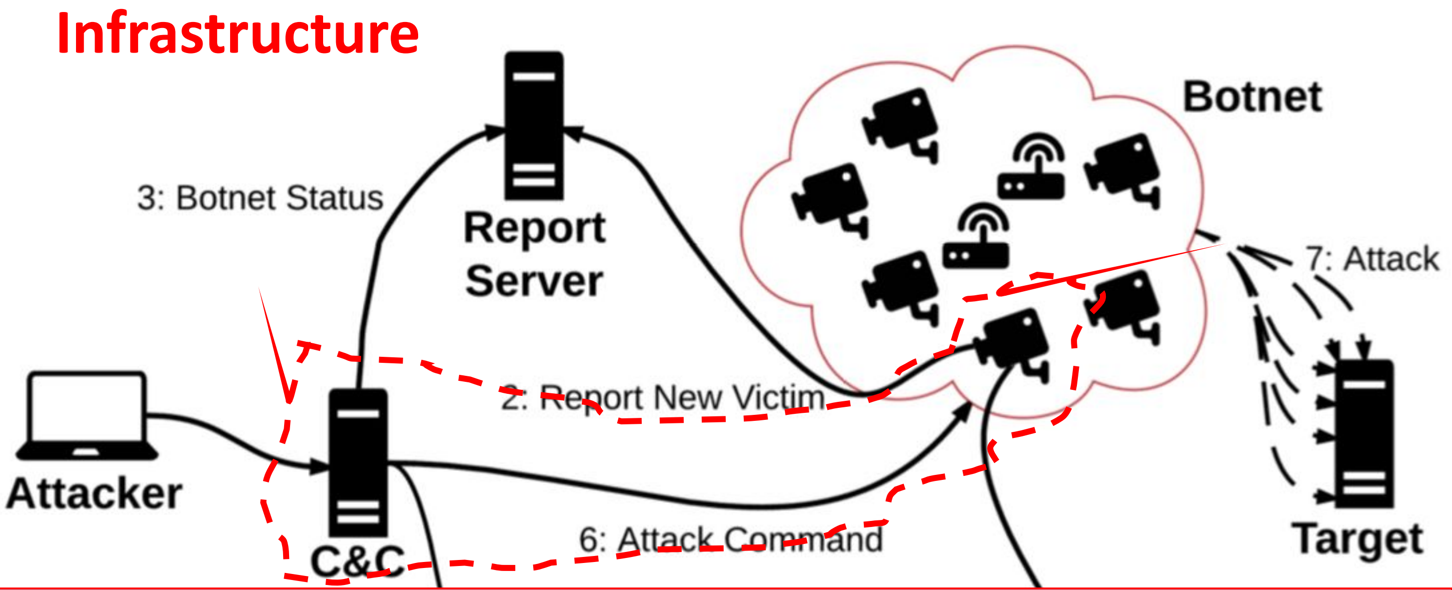
Loader, depending on the <u>hardware architecture</u> of the bot, instructs it to download (using wget or tftp) and execute required binary image of the malware.

Before this is done, the bot needs to know a partition that is writeable.
If tftp or wget clients are not available, the malware will employ *echo* commands to dynamically create the executable.



The Mirai Botnet and the IoT Zombie Armies, 2017

Mirai BotNet Malware



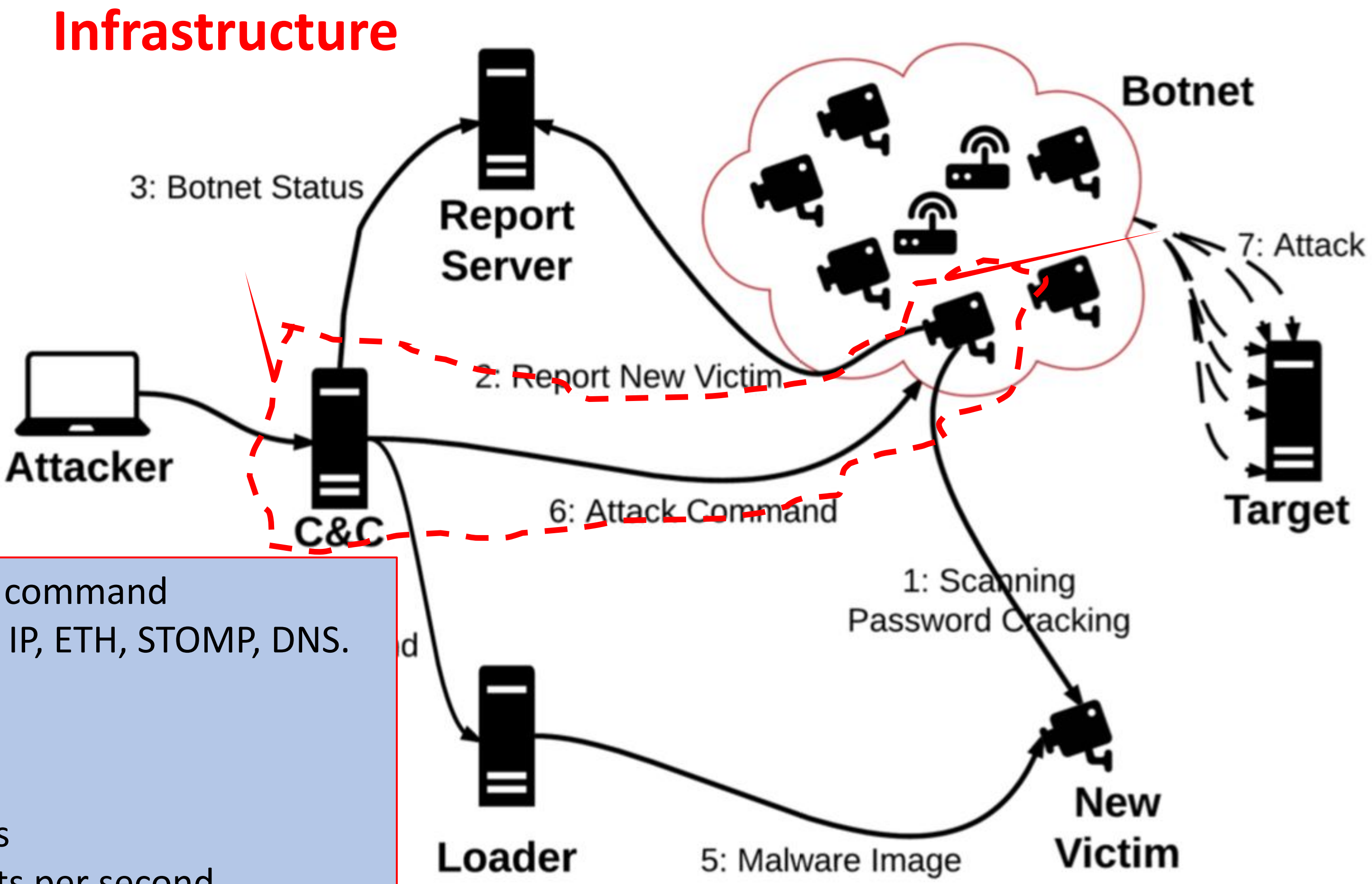
Newly formed bot establishes connection with C&C. Periodic heartbeats between the two.

Other activities in the bot:
* memory scraping to identify other malware present in the bot. If found, kill the process.
  (wants to be the own the device)

- Deletes itself from the persistent storage. Will only be available in RAM (fileless malware)
- Monitors the watchdog timer to defend against system hangs and reboots.
- On command, can start a variety of floods: SYN, ACK, UDP, GRE, DNS, STOMP, ETH. Application layer flooding. 25,000 SYN packets per second.

CR

Mirai BotNet Malware



All botnets attack the target on command
by flooding SYN, ACK, UDP, GRE IP, ETH, STOMP, DNS.
Application level flooding.

Peak: 620 Gbps
Controls: 0.5 million IOT devices
On raspberry Pi 3: 25000 packets per second

CR

Other Mirai Variants

- US university (Feb 2017)
- Windows based strain
- Mirai strain used for bitcoin mining

Advanced Persistent Threat (APT)

- Attackers establish long-term presence in an network in order to mine highly sensitive data

Target network:
- Large enterprises
- Government enterprises
- Critical infrastructure (nuclear plants / power plants)

Attackers:
- Group (typically not an individual)
- Considerable funding
- Government backed

Objective:
- IP theft (trade secrets / patents)
- Compromise sensitive information
- Sabotage
- Total site take over
- Financial

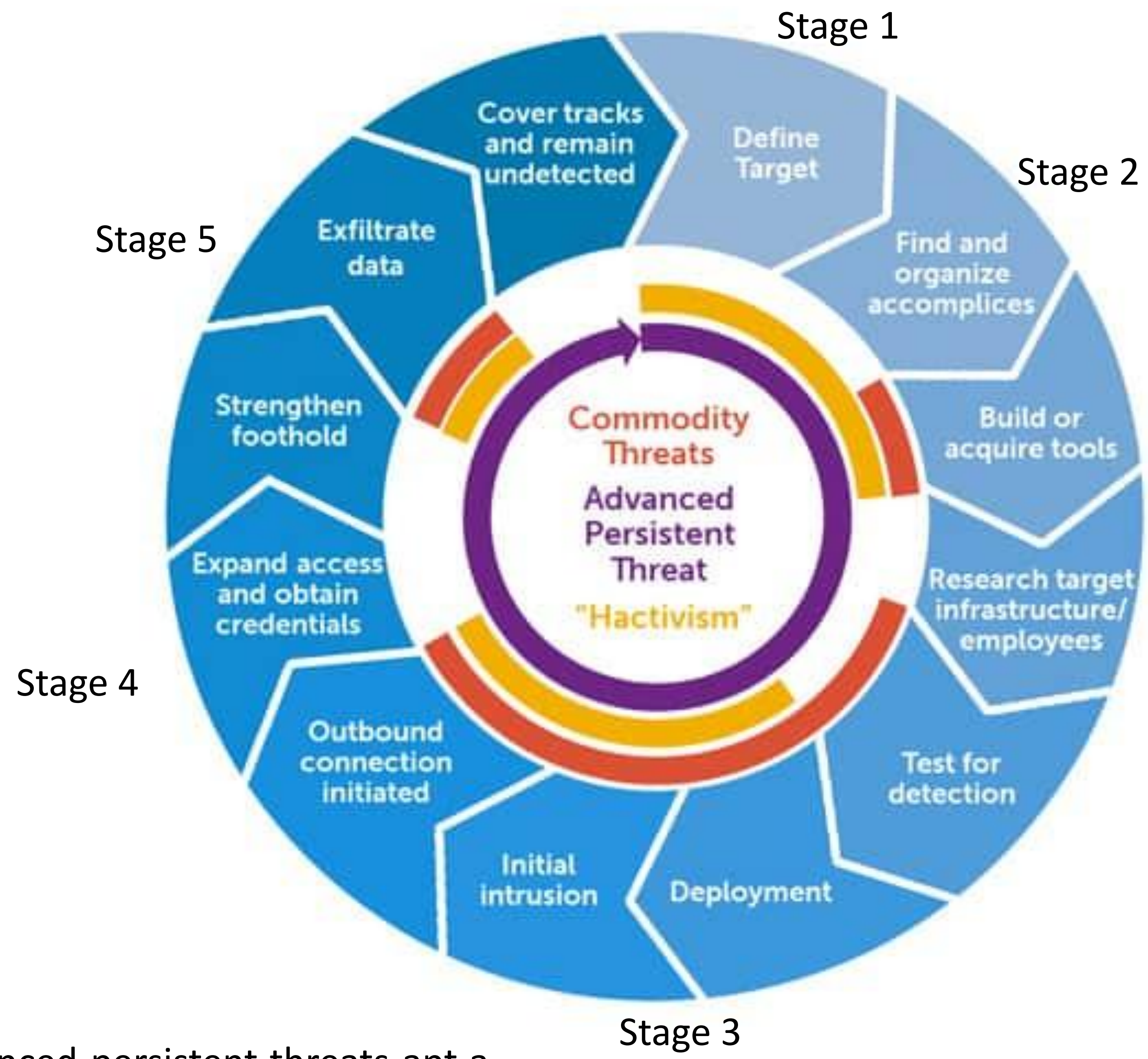
The attack:
- Few days to several months
- * Targets too well guarded.Use multiple techniques
- * 0-day
- * multiple vulnerabilities

CR

Popular cyber attacks based on APTs

- 2015. Ukraine. 230,000 people left without power
- 2019. North American Electric Reliability Corporation (NERC) in 2019
  - Attack on communication using Firewall Firmware Vulnerabilities
    Firewalls between control center and equipments on site.
    All were perimeter based firewalls
    Attacks caused firewalls to reboot. This occurred in a 10-hour time period with each firewall showing offline status for less than 5 minutes
- 2019. US launched cyberattacks on Russian power grid
- 2017. Saudi Armino became target of cyber attacks.
- 2014. Korea Hydro and Nuclear KHNP

APT Stages



CR

https://www.secureworks.com/blog/advanced-persistent-threats-apt-a

APT Stages

Stage 1: Define the target

- Based entirely on the group involved
    - **Financial motivation.** Start scanning financial institutions, large organizations, businesses.

    - **Nation-state motivation.** Target given.
    eg. Steal information about nexgen aircraft design, sabotage government website

APT Stages

Stage 2: Reconnaissance, planning & testing

- Recon. Gather as much information as feasible about the target (infrastructure, defenses, detection methods, assets, employees)
  - Web crawling
  - Employee linkedin pages
  - Inserting agents in target organization
- Planning.
  - How to enter?
  - Team, what skill sets are required?
  - What infrastructure are required?
- PoC
  - prototyping

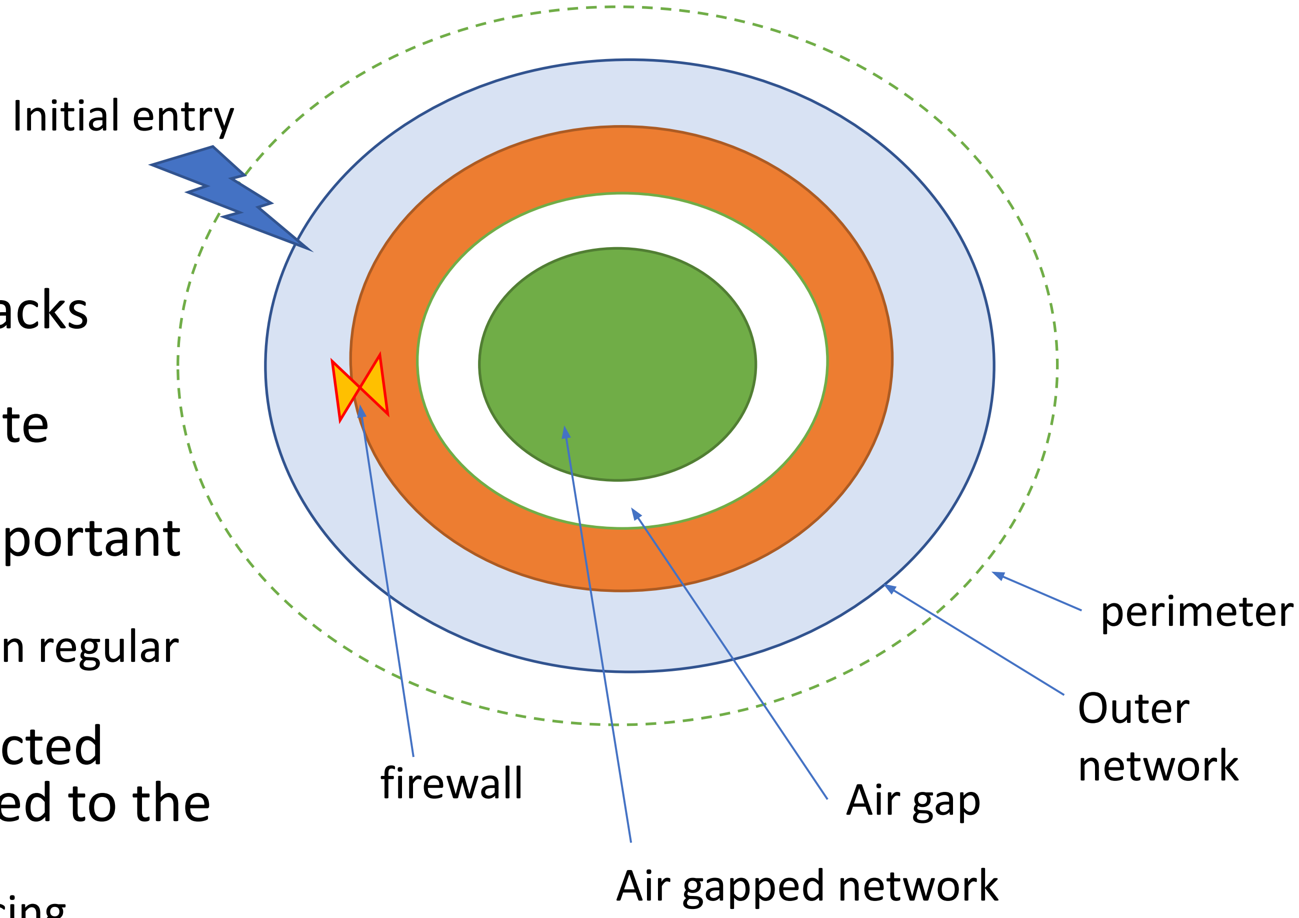
CR

APT Stages

Stage 3: **Infiltration**

Initial entry

- Spam, phishing, social engineering attacks
- Security vulnerabilities
- Luring employees to a malicious website
- Infected USBs
- Spear-phishing: targeted attacks on important people in the organization.
  - Much more organized, more realistic, than regular spam emails
- Entry may also be through a less protected organization that is somehow connected to the target
  - For example, outsourced computer servicing

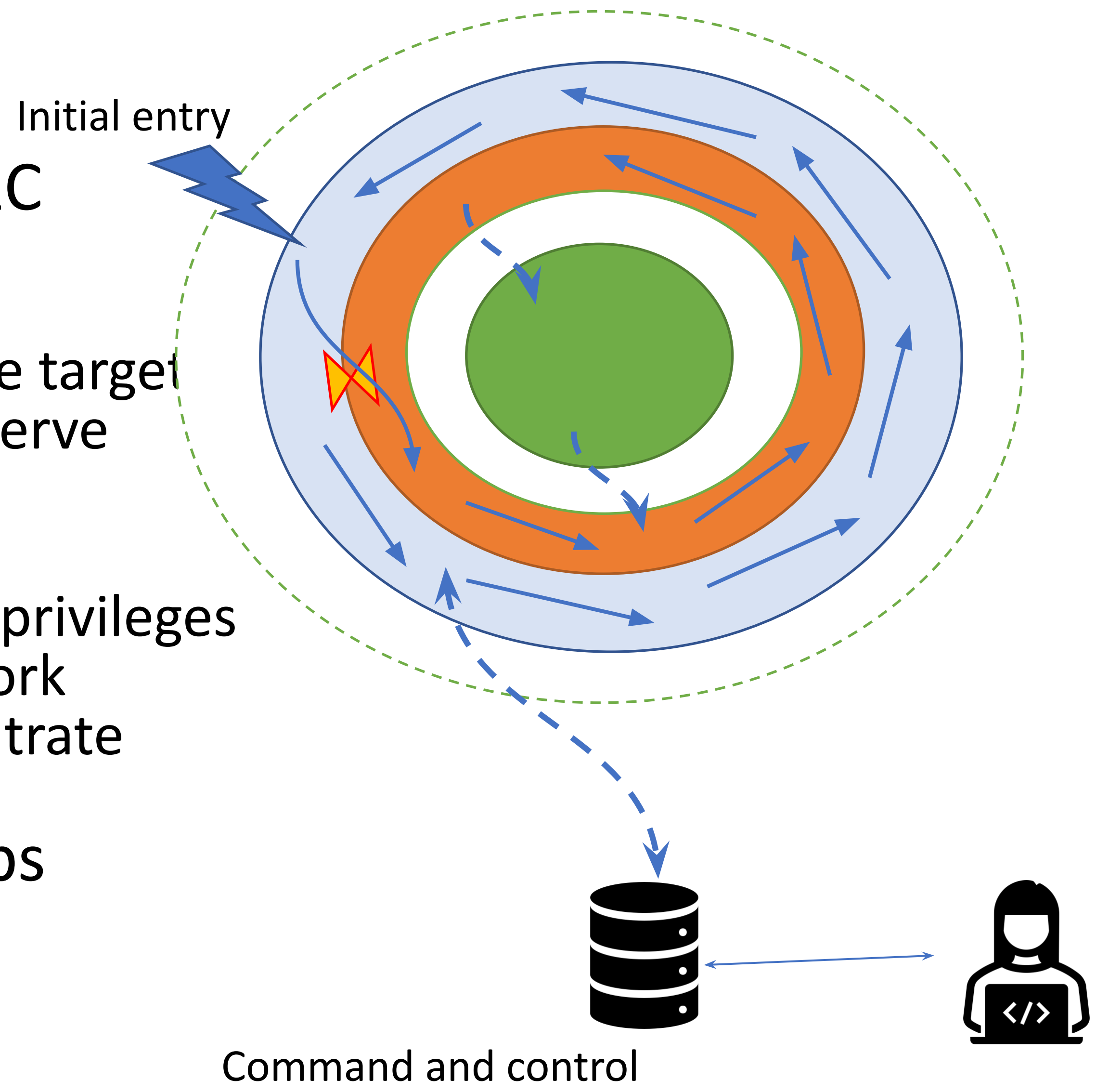


CR

APT Stages

Stage 4: Expansion

- Establish communication with C&C
  - Communication should be discreet (undetectable)
  - Enable attacker to spread within the target network. Can issue commands, observe activities, etc.
- Escalate privileges
  - Move from user privilege to admin privileges
  - Lateral movement within the network
  - Find ways to cross air gaps and exfiltrate data through air gapped network
- Evade detection during these steps



CR

APT Stages

Stage 5: **Completion (final attack)**

- Perform the final operation
- Sabotage operations
- Theft
  - Over-the-air communication
- Destroy data

CR

Stuxnet (An example of an APT)

Stage 1: Target

- Target: Iranian nuclear plants at Natanz

- Objective: Destroy centrifuges that extract fissionable U-235 from natural Uranium.
    - Sufficient amounts of U-235 can be used to build a nuclear bomb.

- Main mode of operation:
    - Infect PLCs (Industrial computers) that control the centrifuges
    - Increases pressure inside the centrifuges
    - While at the same time, provide manipulate human-observable displays to show that all is well!

CR

Stuxnet

Stage 2: Recon, planning, and testing

- Not much information is known about authors of Stuxnet
- Who is responsible? More speculation than anything else.
- Evidence that planning and recon started in 2005
- Attack itself took one year, to slowly damage the Centrifuges.

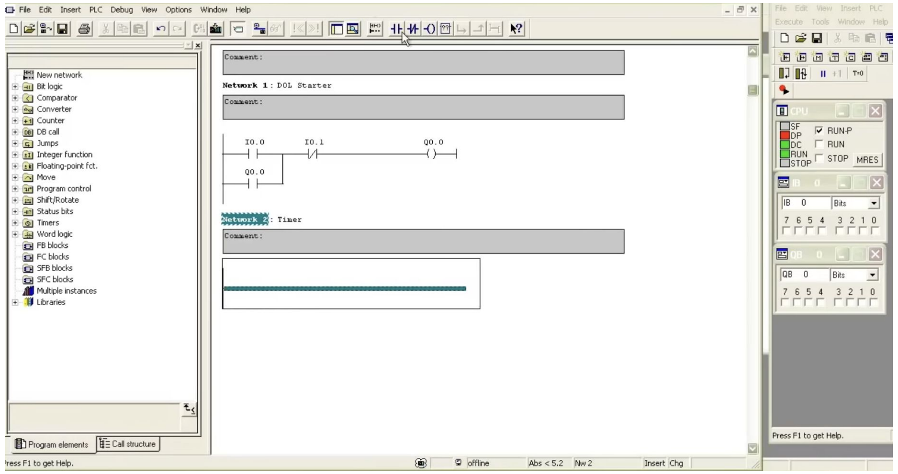
Stuxnet

Stage 3: Initial Entry

- Through four organizations that were somehow related to Natanz plant.
  - For example, manufacture components, assembling, installations
  - Not as secure environment as the Natanz plant

Stuxnet

Stage 4: Expansion



- Infection through Siemens STEP 7 project file
  - Development tool for programming PLCs
- Multiple vulnerabilities exploited
  - CVE-2012-3015: Multiple Siemens SIMATIC Products DLL Loading Arbitrary Code Execution Vulnerability
    - Used to execute a malicious DLL file, which eventually infects services.exe (A Windows service)
    - This is the earliest known vulnerability exploited

CR

Stuxnet

- Over time multiple versions of Stuxnet evolved

Table 1

Evolution of Stuxnet versions

Version	Date	Description
0.500	November 3, 2005	C&C server registration
0.500	November 15, 2007	Submit date to a public scanning service
0.500	July 4, 2009	Infection stop date
1.001	June 22, 2009	Main binary compile timestamp
1.100	March 1, 2010	Main binary compile timestamp
1.101	April 14, 2010	Main binary compile timestamp
1.x	June 24, 2012	Infection stop date

Table 2

Evolution of Stuxnet exploits

Vulnerability	0.500	1.001	1.100	1.101	Description
CVE-2010-3888			X	X	Task scheduler EOP
CVE-2010-2743			X	X	LoadKeyboardLayout EOP
CVE-2010-2729		X	X	X	Print spooler RCE
CVE-2008-4250		X	X	X	Windows Server Service RPC RCE
CVE-2012-3015	X	X	X	X	Step 7 Insecure Library Loading
CVE-2010-2772		X	X	X	WinCC default password
CVE-2010-2568			X	X	Shortcut .lnk RCE
MS09-025		X			NtUserRegisterClassExWow/NtUserMessageCall EOP

https://cyber-peace.org/wp-content/uploads/2013/06/stuxnet_0_5_the_missing_link.pdf

Stuxnet

Stage 4: Expansion

- Multiple Command and Control servers
  - smartclick.org
  - best-advertising.net
  - internetadvertising4u.com
  - ad-marketing.net
- Spread to other systems
- Updating malware versions from the C&C
- Learning about the systems/network/infrastructure

CR

Stuxnet: Expansion

Stage 4.1:

- Multiple Command and Control servers
  - smartclick.org
  - best-advertising.net
  - internetadvertising4u.com
  - ad-marketing.net

- Spread in 4 different countries
- All 4 seemingly independent domains but displayed the same front page
- All created in 2005 (4 years before the attack)

- From Stuxnet infected device to C&C

```
http://<domain>/cgi/link.php?site=xx
```

- Response from C&C, may send updated malware etc. (example)

```
http://<domain>/cgi/click.php?xite=xx&num=yy&c=1&j=%x&k=%x&l=%x
```

CR

https://cyber-peace.org/wp-content/uploads/2013/06/stuxnet_0_5_the_missing_link.pdf

Stuxnet: Expansion

Stage 4.2: Spread

- Spread to other systems
  - Initial version spread only through USB drives carrying STEP 7 project codes
    - Can pass through air-gaps
  - Later versions used other techniques as well
- Looks for STEP 7 software
  - If it doesn't find it on the PC, then ignore and stay dormant

- Sets a registry entry in infected computer to "19790509"
    - To indicate that the system is already infected

Table 3

Evolution of Stuxnet replication

Replication Technique	0.500	1.001	1.100	1.101
Step 7 project files	X	X	X	X
USB through Step 7 project files	X			
USB through Autorun		X		
USB through CVE-2010-2568			X	X
Network shares		X	X	X
Windows Server RPC		X	X	X
Printer spooler		X	X	X
WinCC servers		X	X	X
Peer-to-peer updating through mailslots	X			
Peer-to-peer updating through RPC		X	X	X

CR

https://cyber-peace.org/wp-content/uploads/2013/06/stuxnet_0_5_the_missing_link.pdf

Stuxnet: Expansion

Stage 4.3: Update

- Updated malware spread using a P-2-P network
  - As long as one infected system is updated with latest malware, the updates spread to other systems as well
  - P2P network established using Windows Mailslots
    - Mailslots allow a process in 1 Windows machine to send a message to another windows machine

CR

Stuxnet: Expansion

Stage 4.3: Learn

- How devices are addressed?
- Identify symbols in PLC code that represent the devices controlled by the PLC

PV A21-8-160

Associated with centrifuge

cascade

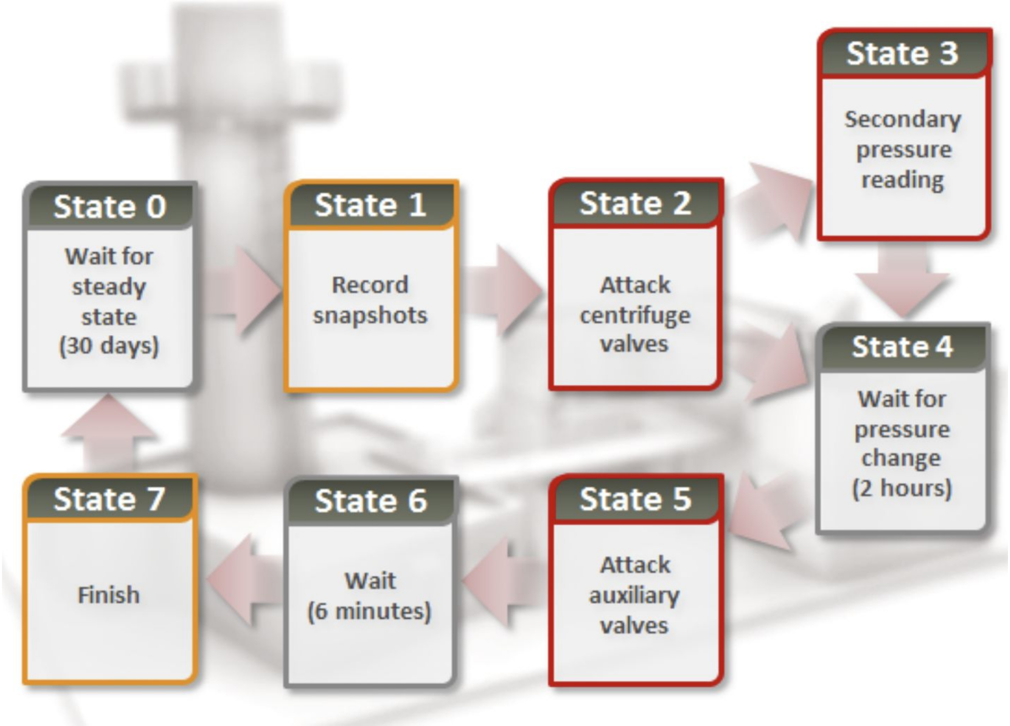
module

Pressure valve

CR

Stuxnet: Final Attack

- DB8061 code that runs in the PLCs



https://cyber-peace.org/wp-content/uploads/2013/06/stuxnet_0_5_the_missing_link.pdf

Malware Detection

Malware Detection

- Static analysis / Dynamic analysis
  - Done on binaries vs done by monitoring execution behavior
- Detection techniques:
  - Signatures based on a database of updated signatures
    - Not scalable because (a) too many malware to keep database up-to-date
  - Makes use of heuristics
    - Static analysis: Decompile binary and examine source code
    - Dynamic analysis: For example, too many files opened per second
  - False positives
    - Signature based approaches 0% false positives
      AI based approaches >0% false positives

CR

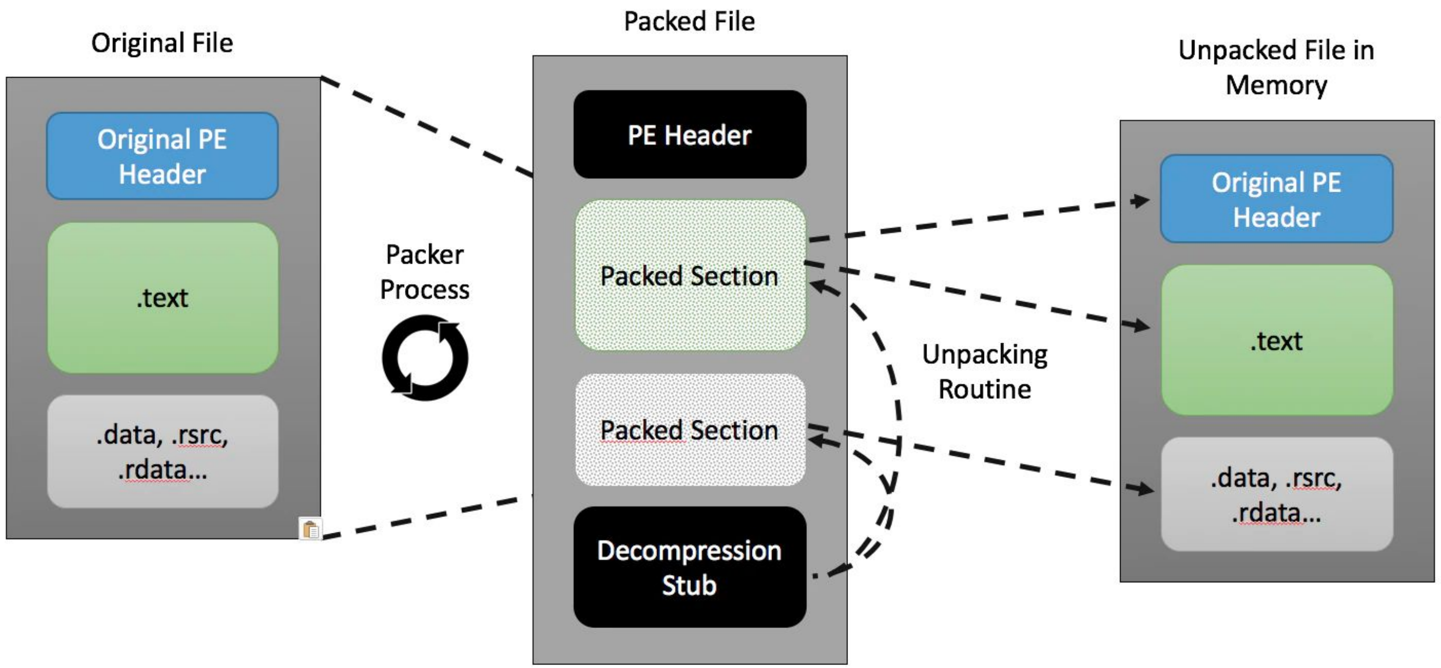
Malware Evasion

Malware Evasion

- Evading Static Analysis
- Evading Dynamic Analysis
- Evading Sandbox environments

Evading Static Analysis with Polymorphic malware

- Polymorphic malware looks to hide known patterns in malware binaries, either by compressing them or encrypting them
- Each replica has a different signature, however, when unpacked they are all identical.
- Detection can be done in the memory after unpacking



CR

https://www.mcafee.com/blogs/enterprise/malware-packers-use-tricks-avoid-analysis-detection/

Evading Static Analysis with Metamorphic malware

- Malware recodes itself each time it propagates

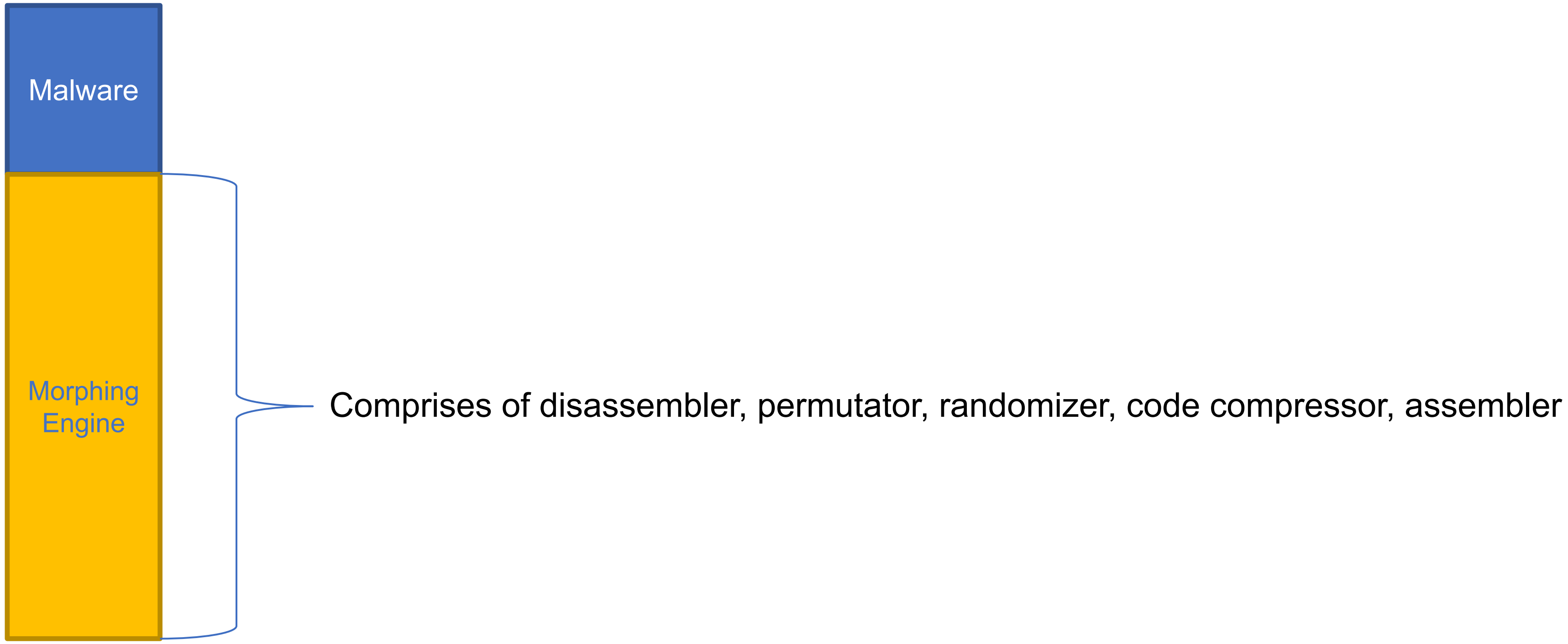
- Assembly-level rewriting
  - adding NOPs
  - permuting use of registers in instructions
  - inserting arbitrary instructions to confuse

- Higher-level modifications
  - Alter control flow (inserting additional jumps)
  - Reorder elements in structure

https://www.blackhat.com/presentations/bh-usa-08/Hosmer/BH_US_08_Hosmer_Polymorphic_Malware.pdf

Evading Static Analysis with Metamorphic malware

- Limitations
  - Identify morphing engine code



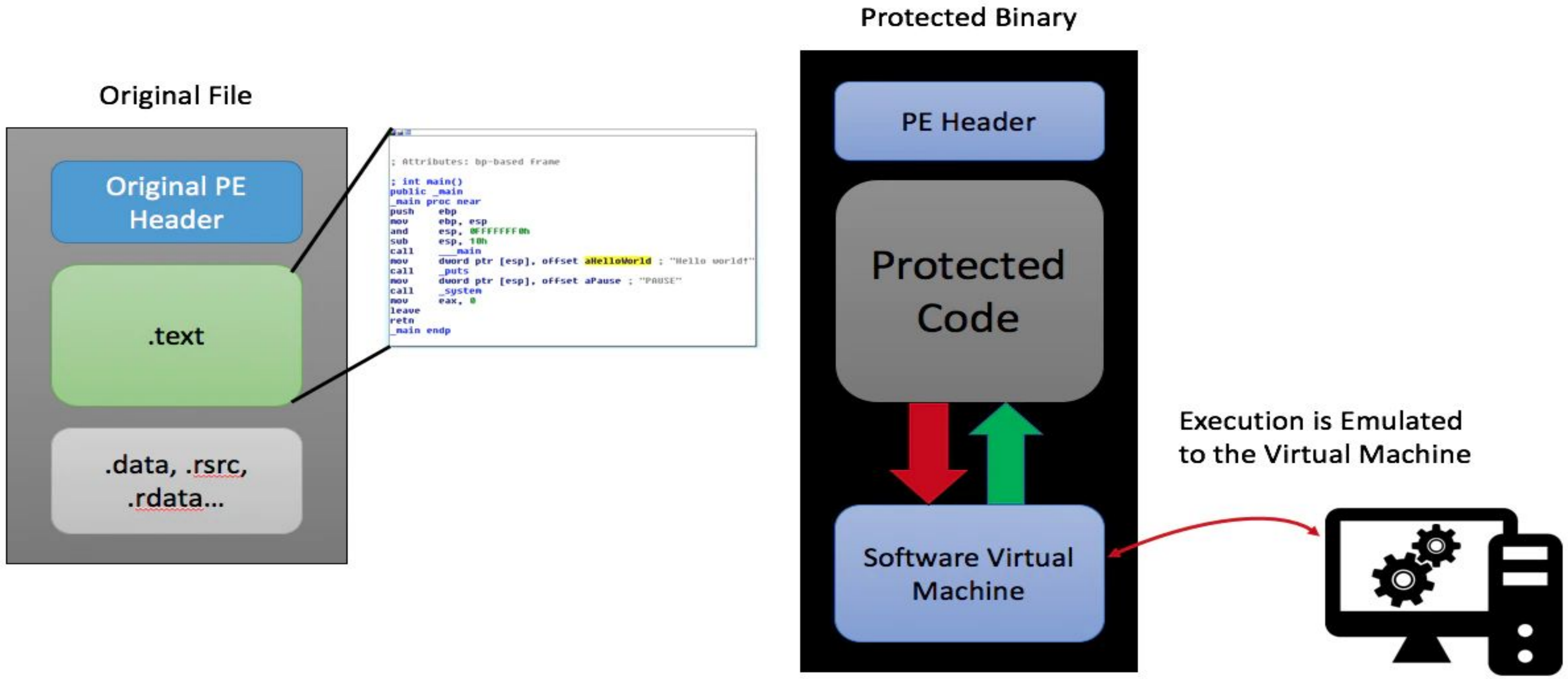
CR

https://www.blackhat.com/presentations/bh-usa-08/Hosmer/BH_US_08_Hosmer_Polymorphic_Malware.pdf

Evading Static Analysis (other techniques)

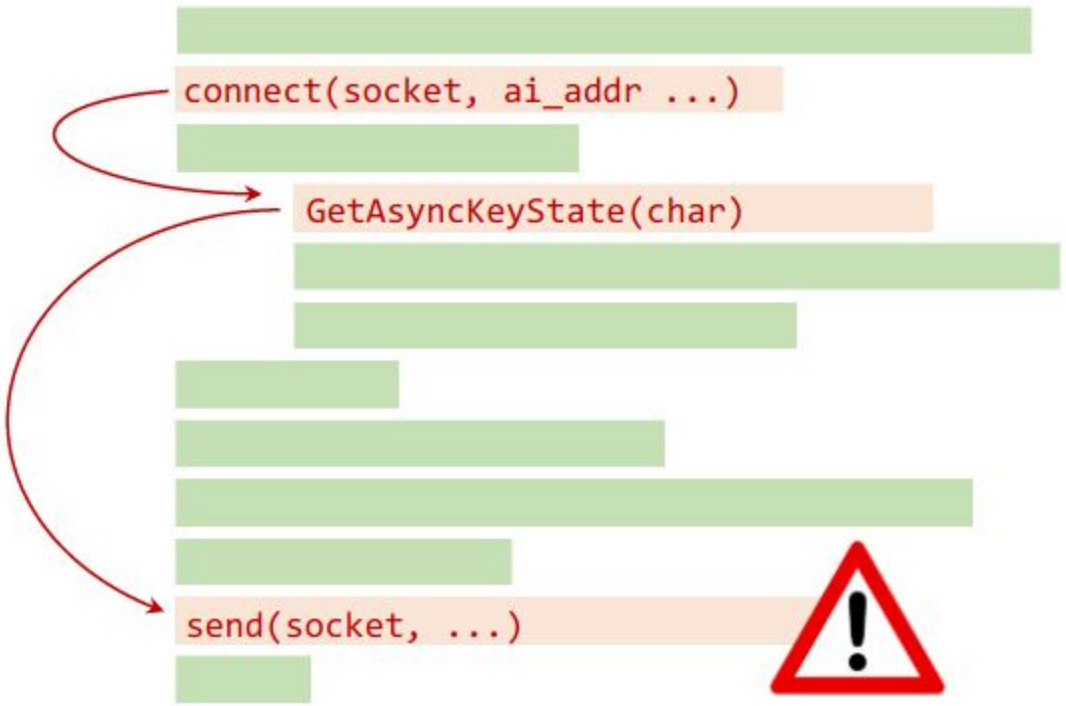
Use a single instruction to hide patterns

customized hypervisors



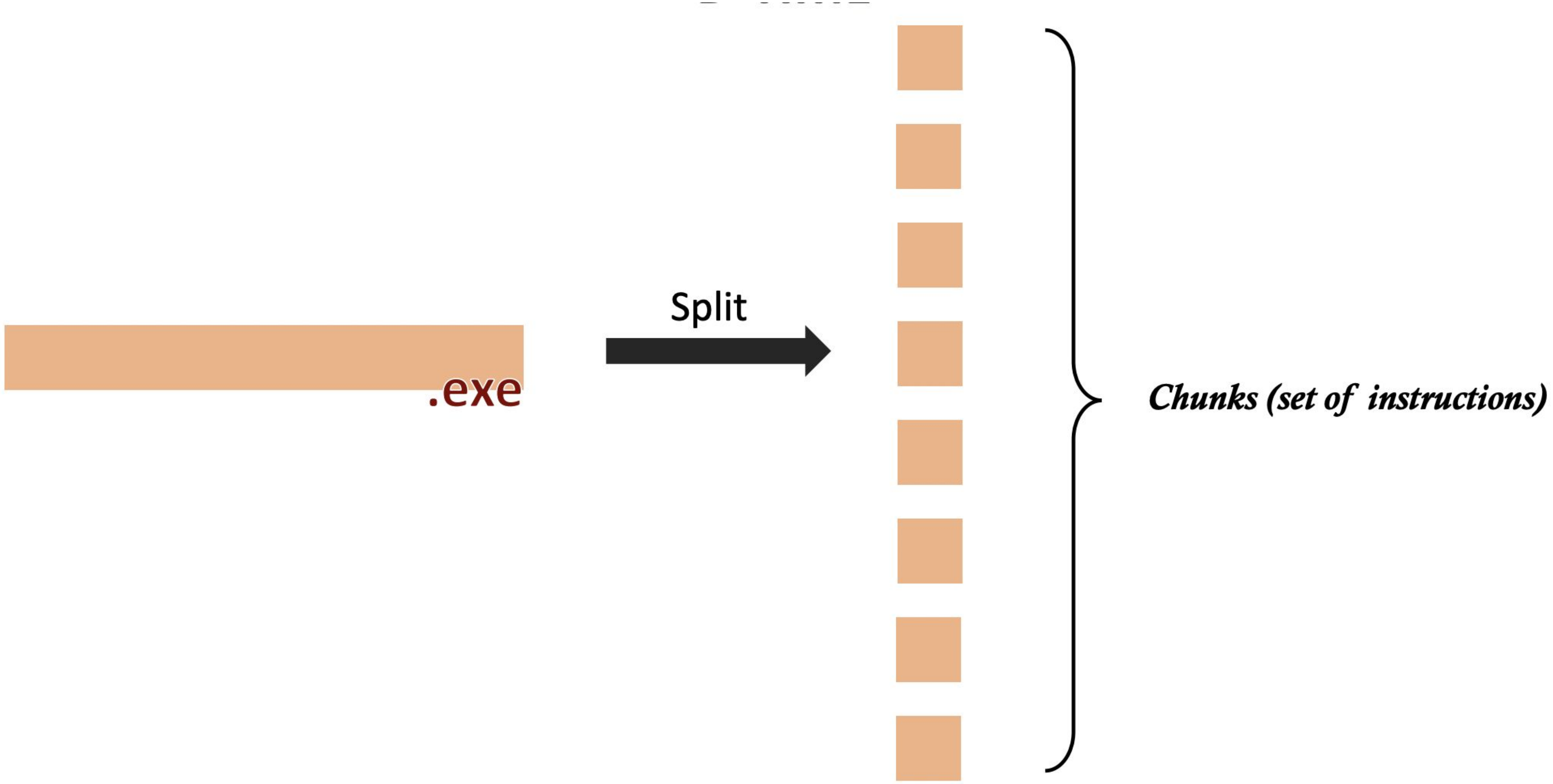
CR

Dynamic Detection



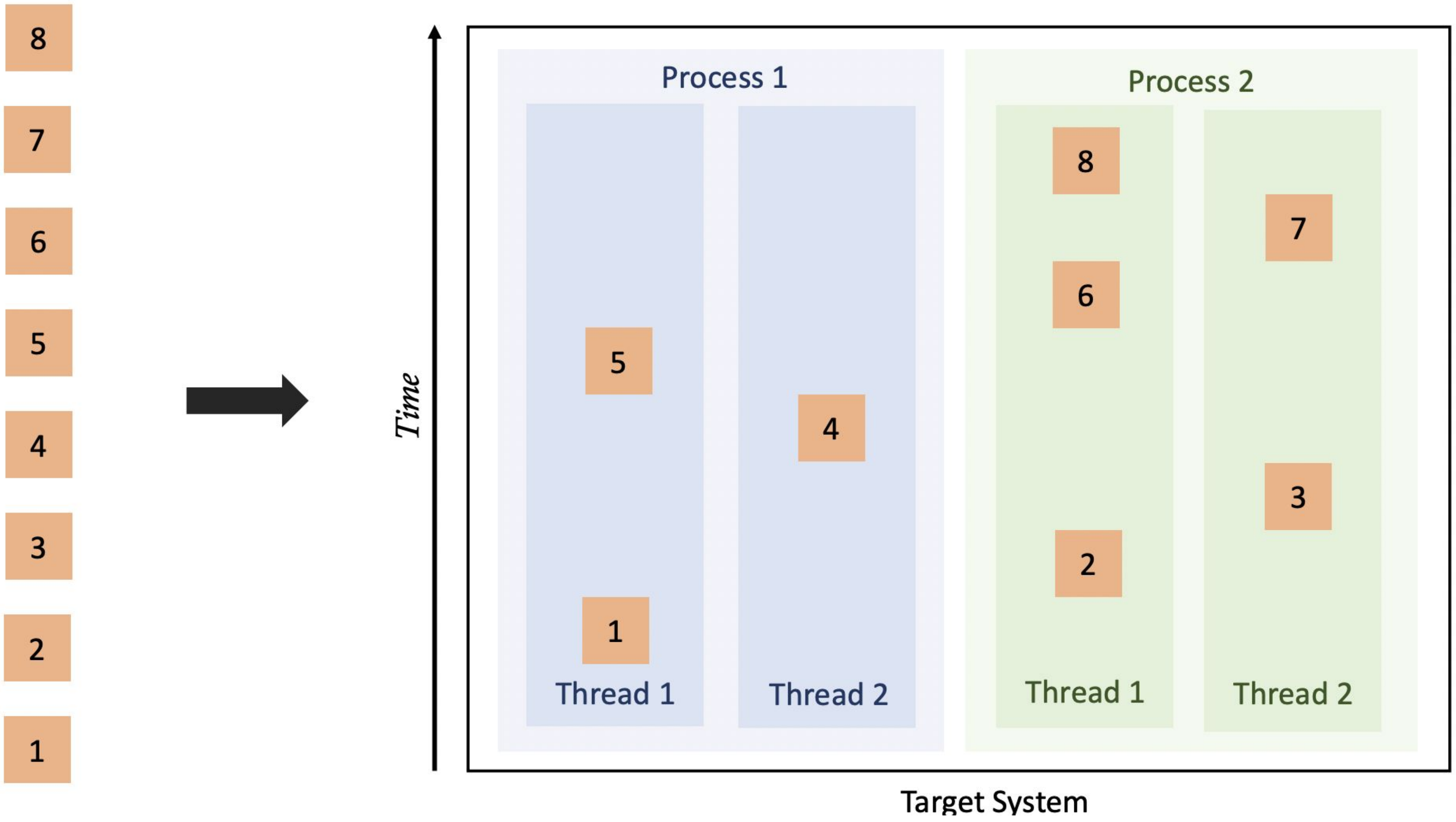
Track patterns in API (system calls)

Evading Dynamic Analysis (with D-TIME)



https://www.usenix.org/conference/woot19/presentation/pavithran

D-Time: Distributed Threadless Independent Malware Execution for Runtime Obfuscation

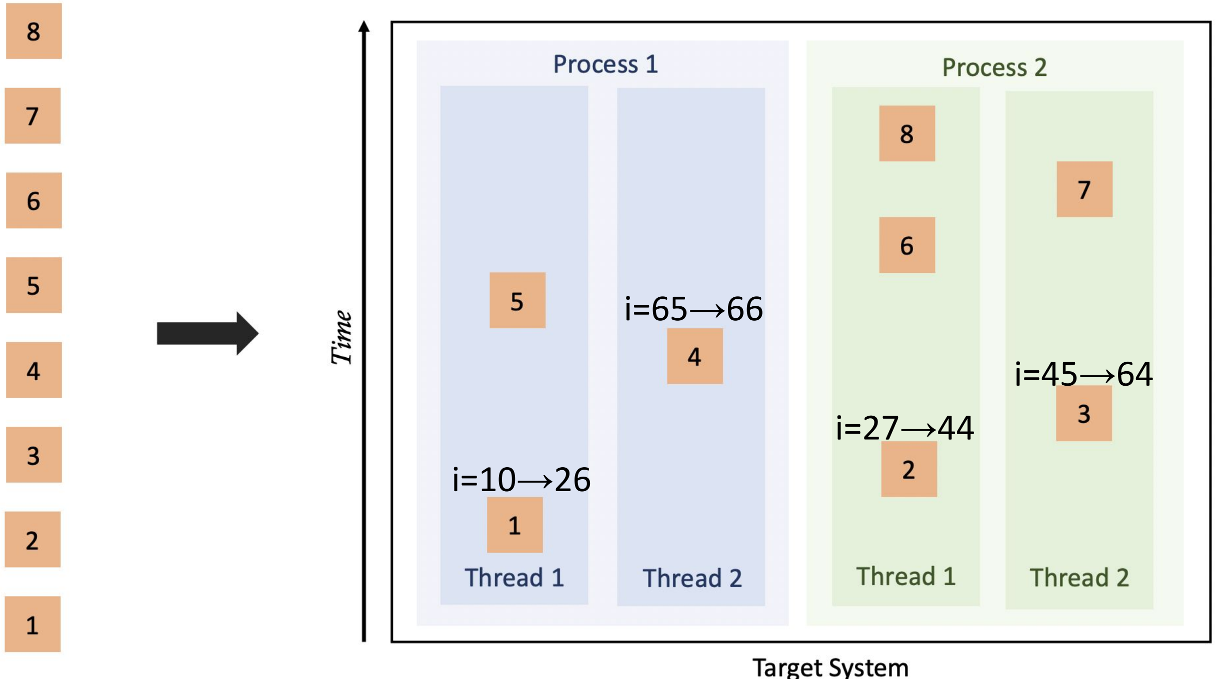


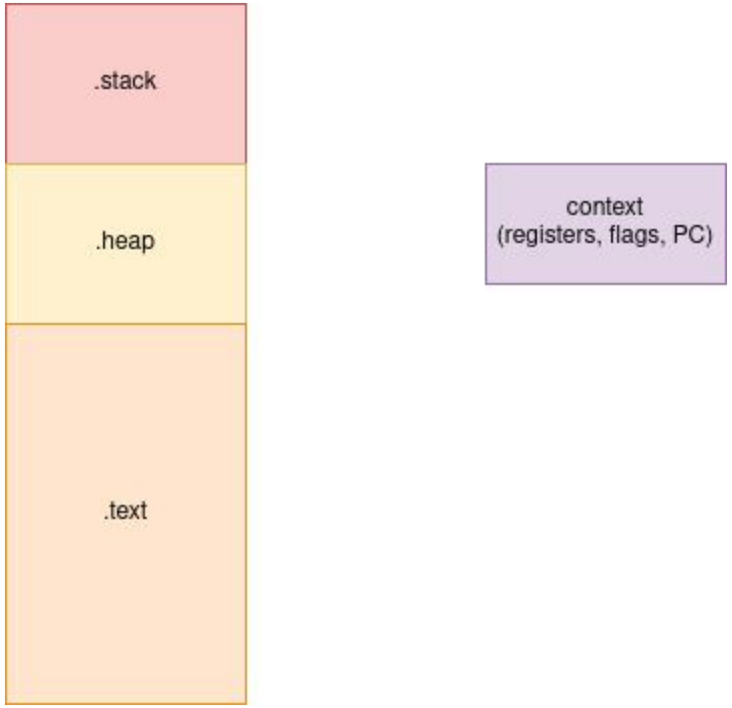
CR

Process State and Context

How to keep track of the context across multiple processes?

```
i = 10;
while(1){
      i++;
}
```





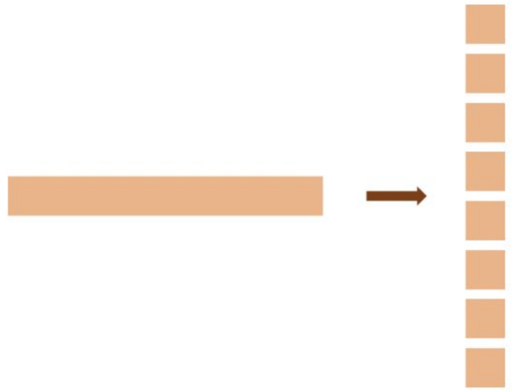
Requirements to achieve Distributed Independent Execution

- How to inject chunks in executing threads?
- How to ensure context is saved and restored between chunk executions?
- How to synchronize execution between threads?

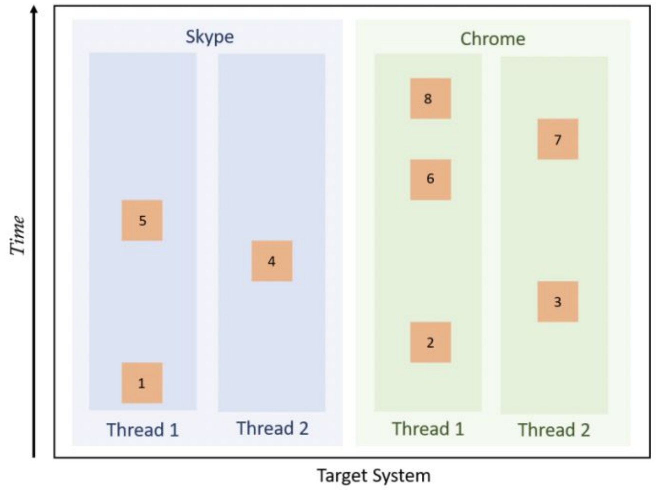
CR

Two phases

Offline Phase



Online Phase



Offline phase: Creating the chunks

```
      ...                    ;
    cmp [a], [b]             ; if(a==b){
    jne a_unequal_b          ;
    mov [a], 0               ;     a = 0
a_unequal_b:                 ; }
    mov [c], [d]             ; c = d
      ...                    ;
```

CR

Offline phase: Creating the Chunks

ebx used to indicate the control flow

Chunk 1

```
  ...                  ;
cmp [a], [b]           ; if(a==b){
jne a_unequal_b        ;
mov ebx, 2
jmp END
jmp a_unequal_b:
mov ebx, 3
```

Chunk 2

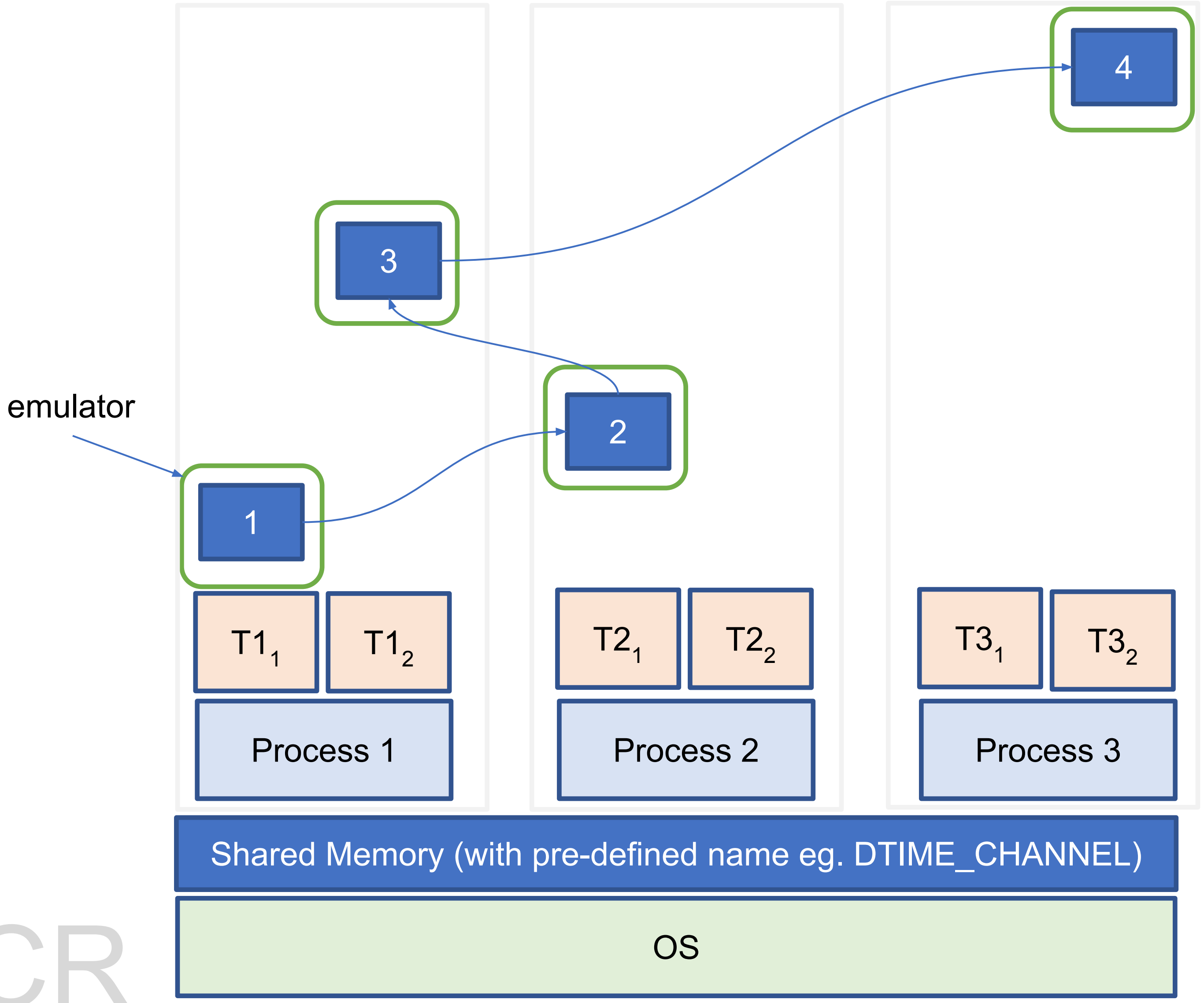
```
mov [a], 0             ;     a = 0
mov ebx, 3
```

Chunk 3

```
mov [c], [d]           ; c = d
  ...                  ;
```

CR

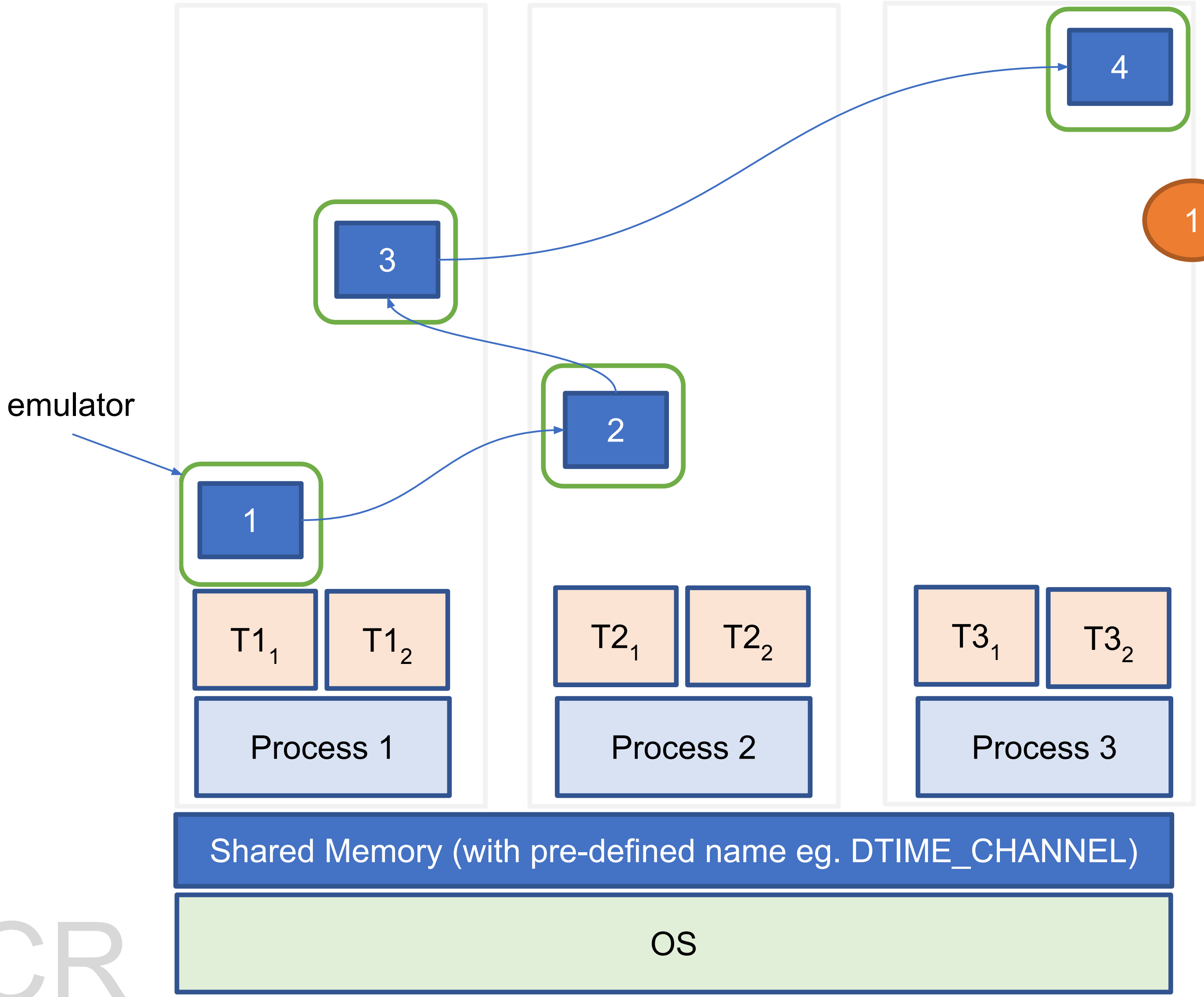
Online Phase: Emulators and Shared Memory



Emulator:
1. Resolves addresses of DLLs (Shared libraries)
2. Is this the first emulator running?
   * YES: create a shared memory
   * NO: find a way to connect to the shared memory
3. Lock Mutex and identify the next chunk to execute
4. Retrieve next chunk
5. Adjust the addresses in the chunk
6. Retrieve context (registers)
7. Execute Chunk
8. Store context (registers) in shared memory
9. Trigger chunks to execute in other threads

Shared memory:
1. Created by the first process
2. Stores state, like registers stack variables, etc.
3. Synchronizes between various chunk executions.
   (so that a new chunk execution knows what to execute)

Online Phase: Emulators and Shared Memory



Emulator:

1. Resolves addresses of DLLs (Shared libraries)
2. Is this the first emulator running?
   * YES: create a shared memory
   * NO: find a way to connect to the shared memory

   1
3. Lock Mutex and identify the next chunk to execute
4. Retrieve next chunk (ebx register)
5. Adjust the addresses in the chunk
6. Retrieve context (registers)
7. Execute Chunk
8. Store context (registers) in shared memory
9. Trigger chunks to execute in other threads 2
10. Unlock mutex

Shared memory:

1. Created by the first process
2. Stores state, like registers stack variables, etc.
3. Synchronizes between various chunk executions.
   (so that a new chunk execution knows what to execute)

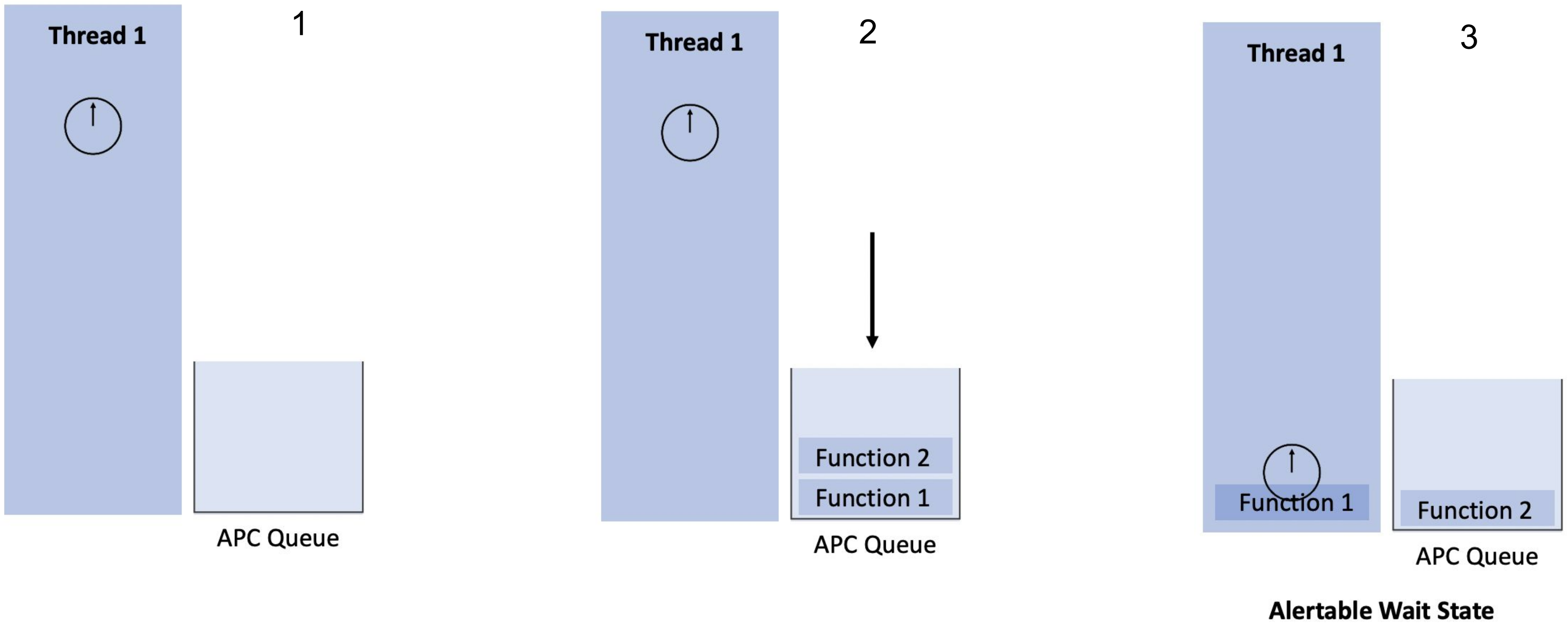
CR

2

APCs to Trigger Chunk Execution

```
DWORD QueueUserAPC(
  PAPCFUNC  pfnAPC,
  HANDLE    hThread,
  ULONG_PTR dwData
);
```

Executing a Chunk using APC (Asynchronous Procedure Call)




```
SleepEx
SignalObjectAndWait
MsgWaitForMultipleObjectsEx
WaitForMultipleObjectsEx
WaitForSingleObjectEx
```

CR

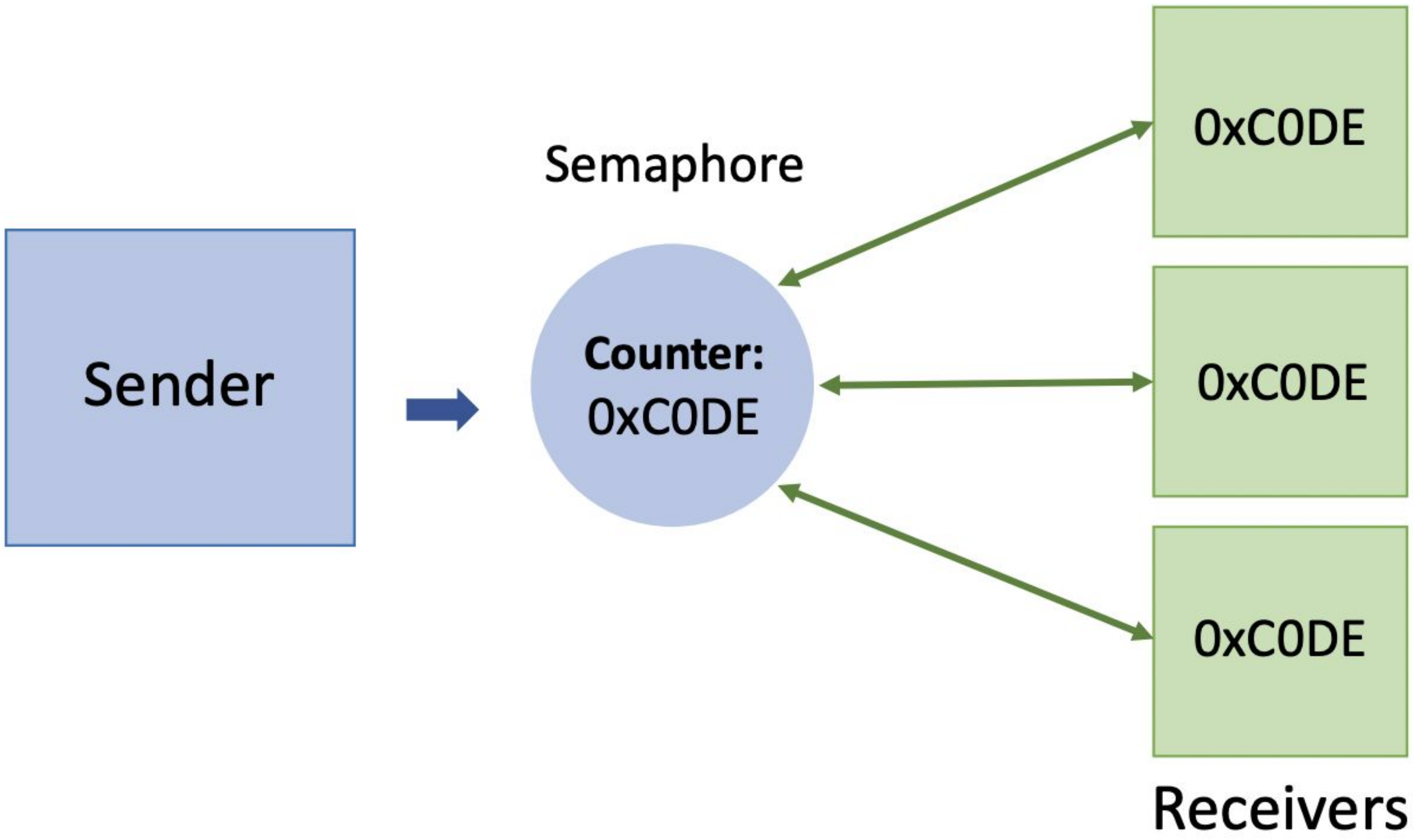
Shared Memory in Windows

- A chunk can connect to a specific shared memory using APIs like OpenFileMapping(DTIME_CHANNEL)
- This would
  - map the shared memory into the virtual address space of the process and
  - Return pointer to the shared memory
- After the chunk executes,
  - Shared memory still mapped
  - But virtual address is lost

How does a subsequent chunk executing in the process know where the shared memory is?

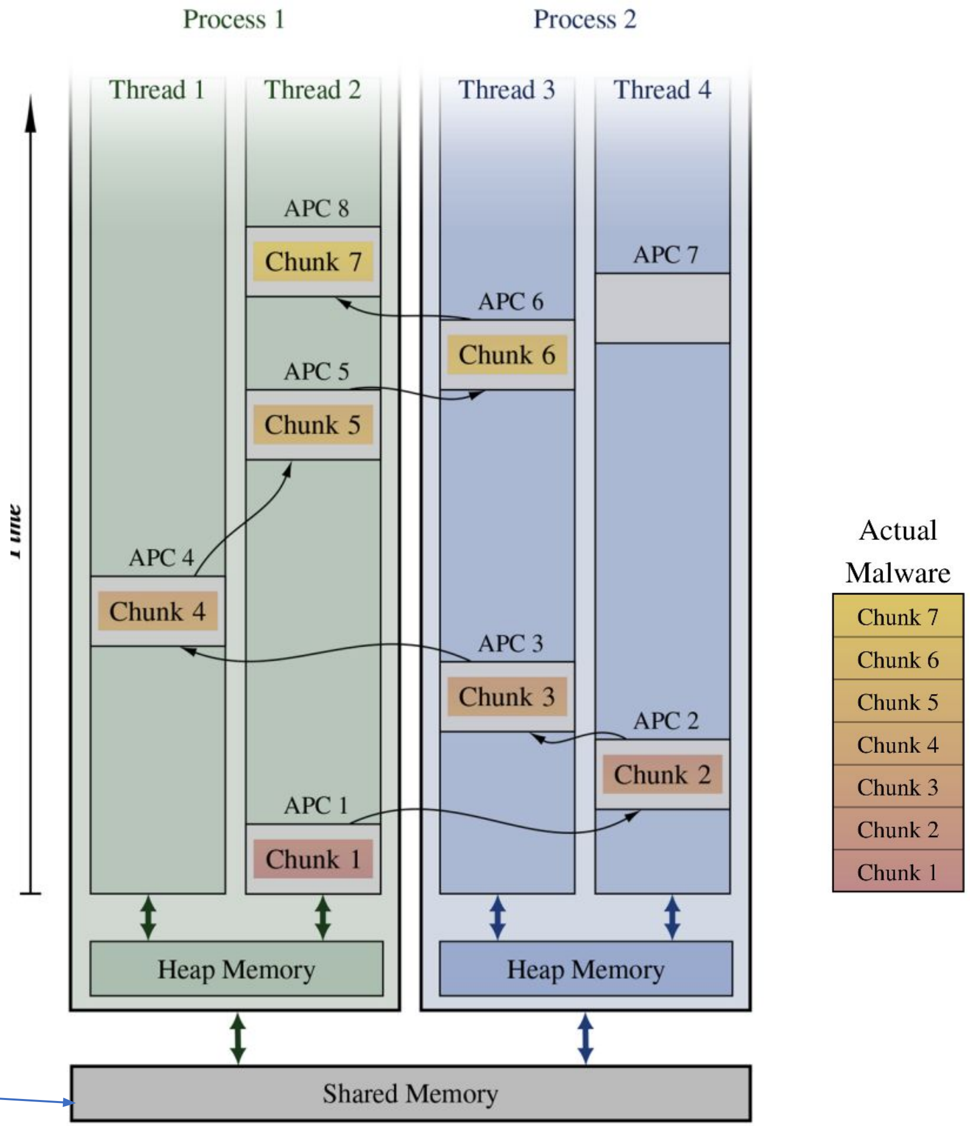
CR

SCBC: Semaphore Based Covert Channel



CR

D-TIME Execution



Used to ensure context is saved.
Stack, global, heap present here

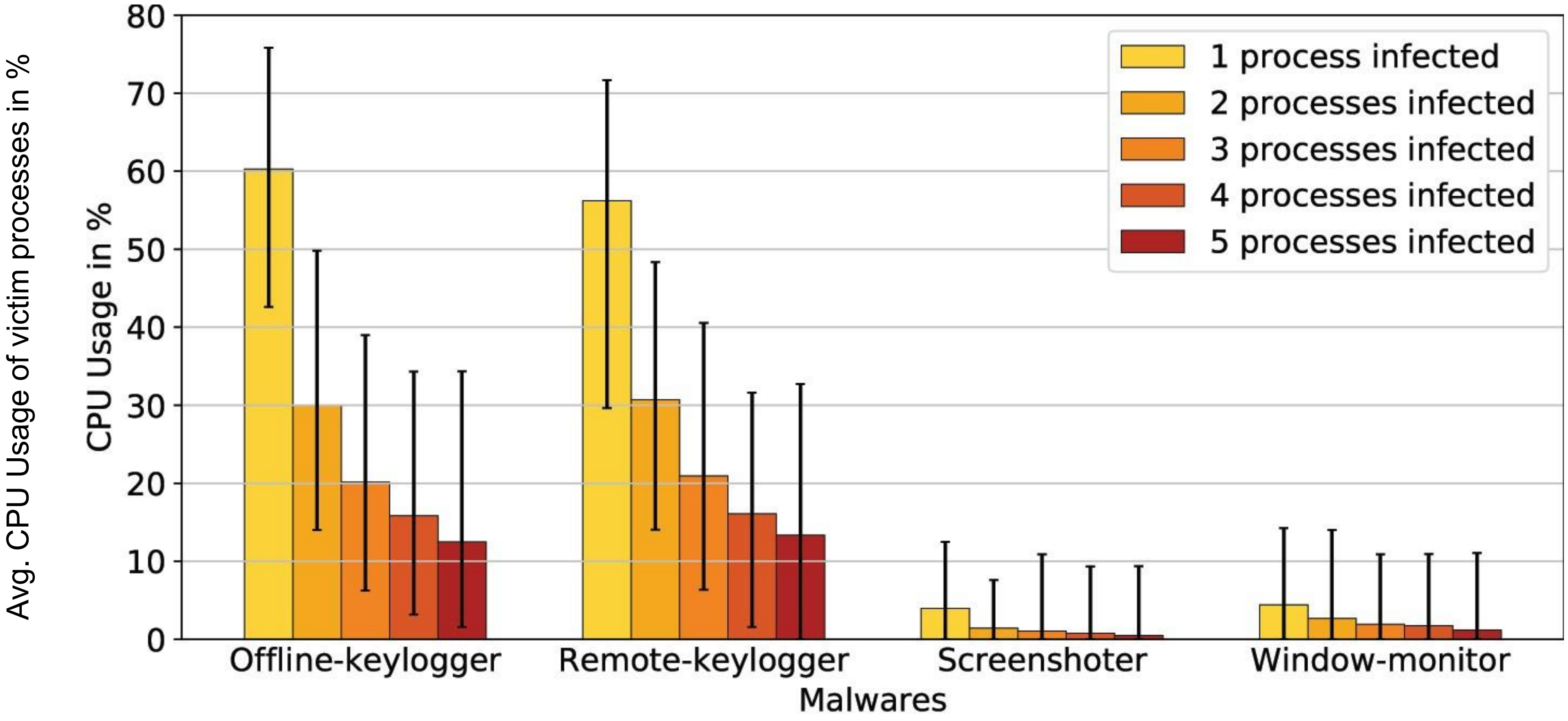
Evaluation & Results

1. BitDefender
2. Norton
3. Kaspersky
4. WEBROOT
5. McAfee
6. ESET
7. Avast
8. AVG
9. Windows Defender
10. Avira

CR

Evaluation & Results

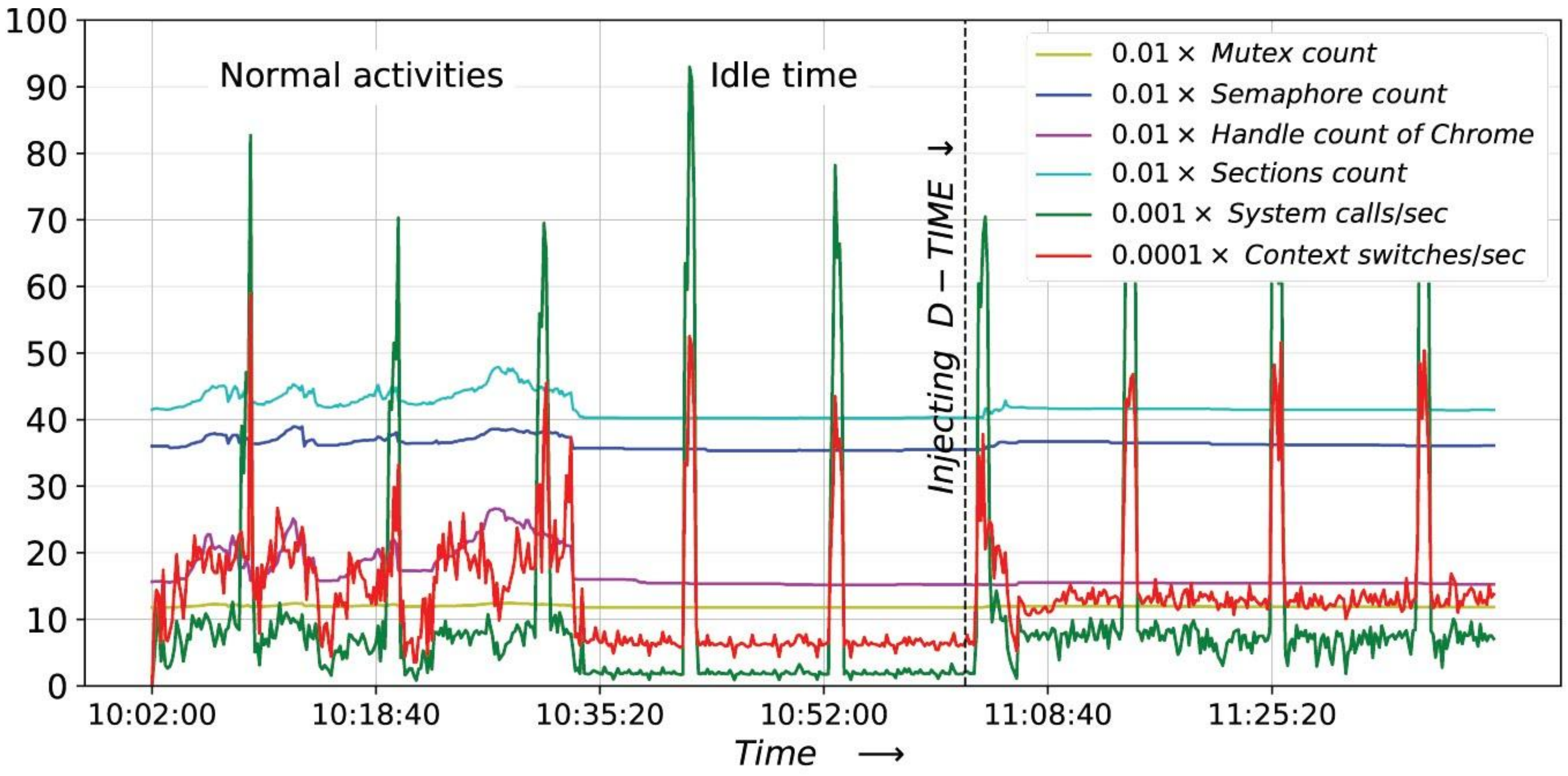
CPU usage for different number of infected processes



CR

Evaluation & Results

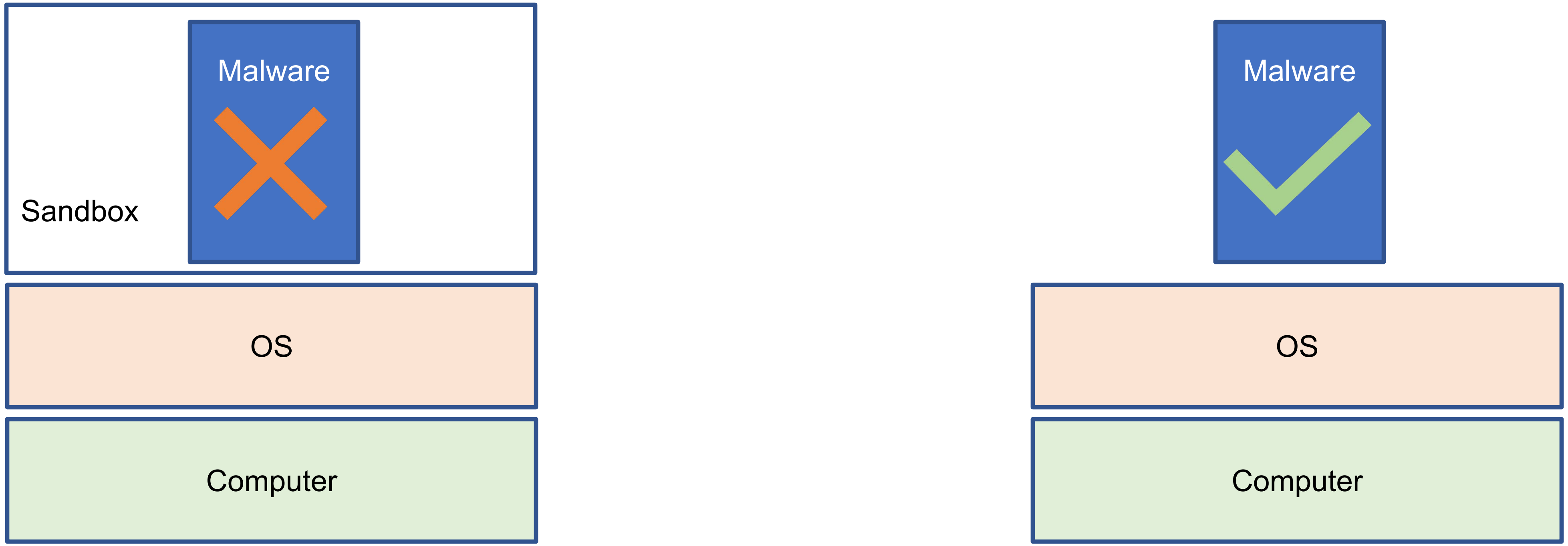
Performance Counters



CR

Evading Sandbox and Simulation Environments

Malware detects the presence of a sandbox and does not execute



https://evasions.checkpoint.com/

Evading Sandbox Evasion Techniques

Malware detects the presence of a sandbox using clues

1. Filesystem(F)
2. Registry (R)
3. Generic OS Queries (OSQ)
4. Global OS Objects (GOS)
5. UI Artifacts (UIA)
6. OS Features (OSF)
7. Processes (P)
8. Network (N)
9. CPU (C)
10. Hardware (HW)
11. Firmware Tables (FW)
12. Hooks (H)

https://evasions.checkpoint.com/

CR