An Application Agnostic Defense Against the Dark Arts of Cryptojacking

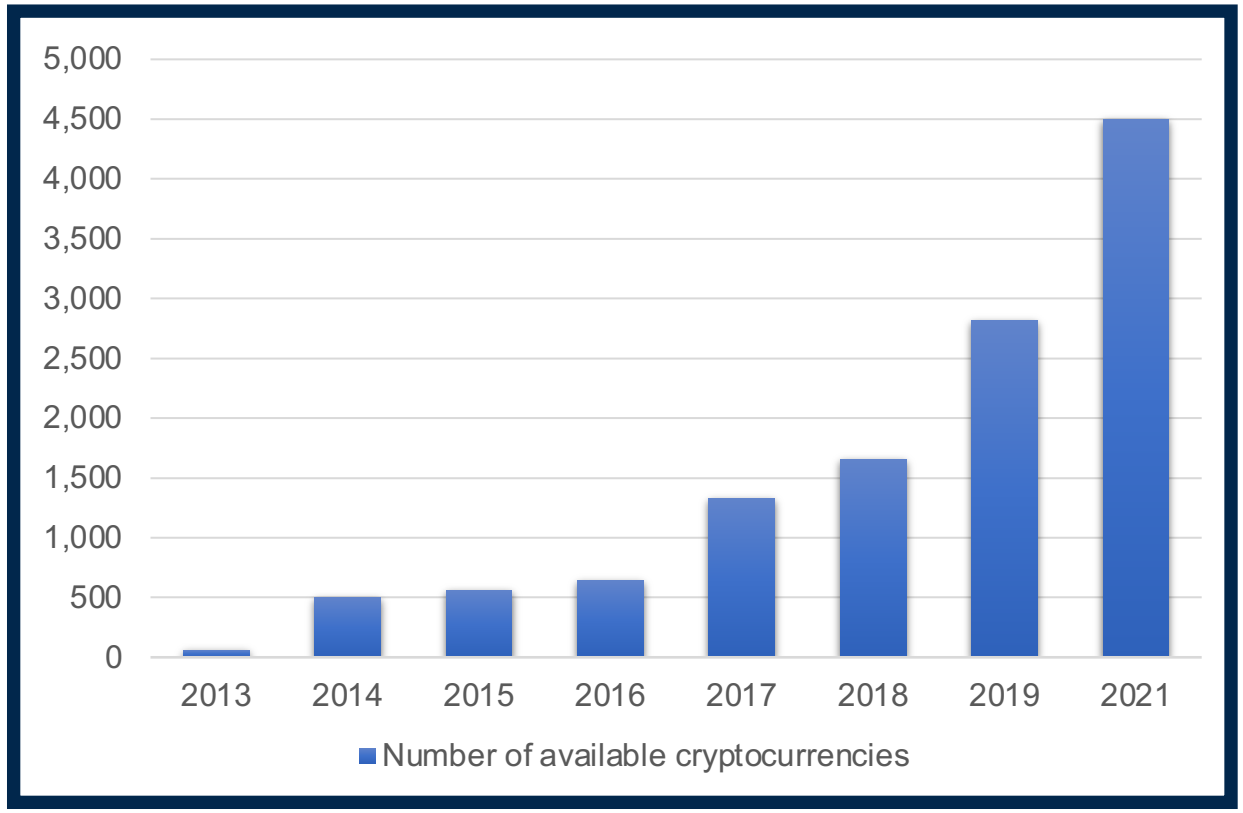
Nada Lachtar

Abdulrahman Abu Elkhail

Anys Bacha

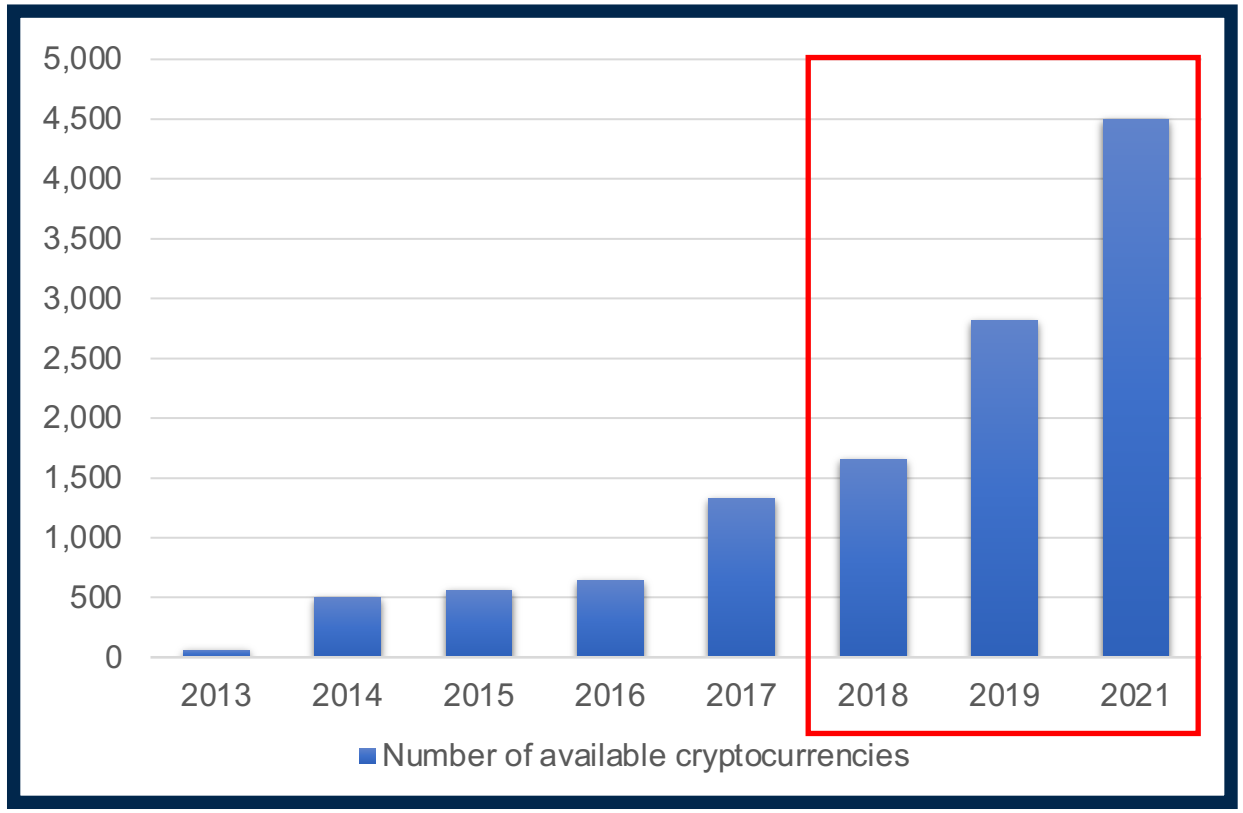
Hafiz Malik

The Emergence of Cryptocurrencies



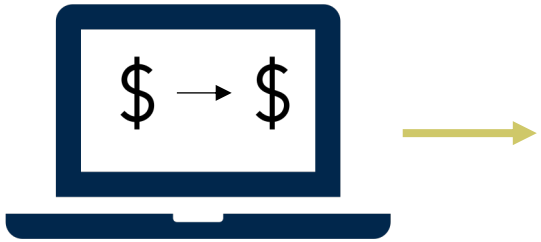
Number of available cryptocurrencies

The Emergence of Cryptocurrencies

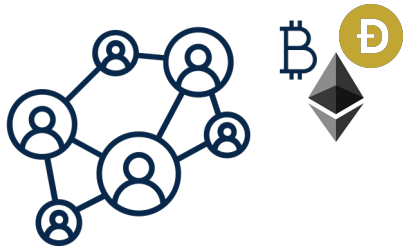


Number of available cryptocurrencies

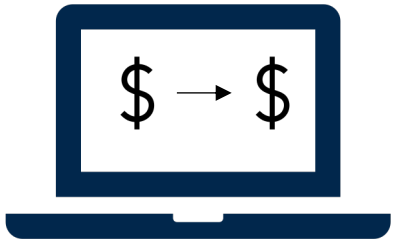
A Decentralized Type of Currency



Customer initiates transaction

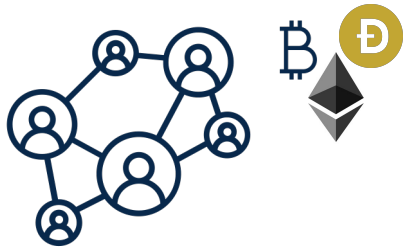


A Decentralized Type of Currency

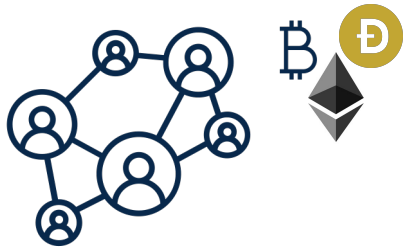


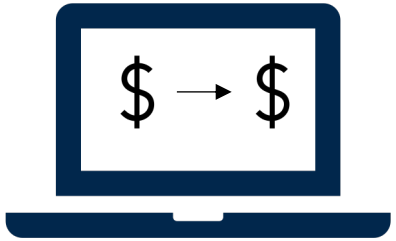
Customer initiates transaction

Broadcast network of P2P nodes (miners)



A Decentralized Type of Currency



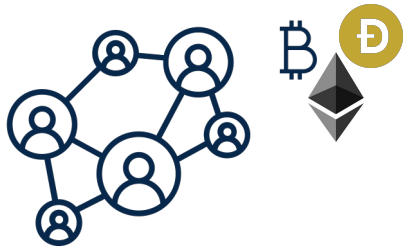


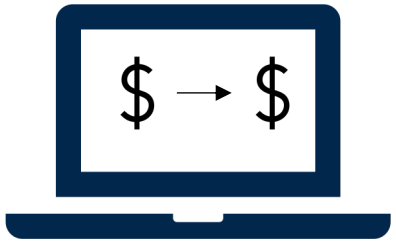
Customer initiates transaction

Broadcast network of P2P nodes (miners)

Mine new block

A Decentralized Type of Currency





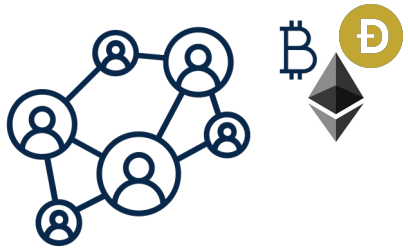
Customer initiates transaction

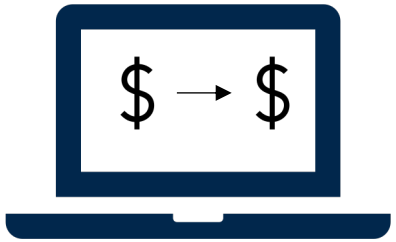
Broadcast network of P2P nodes (miners)

Mine new block

Add newly mined block to blockchain

A Decentralized Type of Currency





Customer initiates transaction

Broadcast network of P2P nodes (miners)

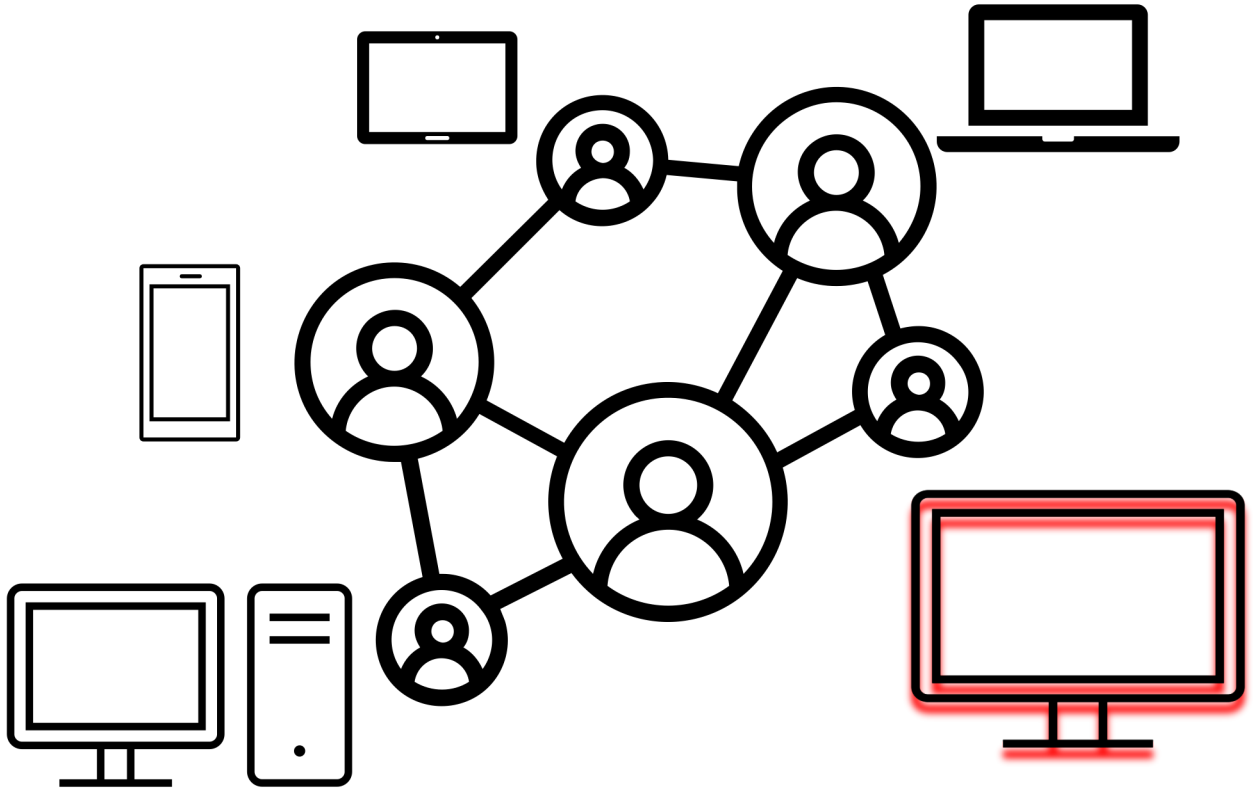
Mine new block

Add newly mined block to blockchain

Complete transaction

The Rise of Cryptojacking Malware

The Rise of Cryptojacking Malware



Attacker

Victim

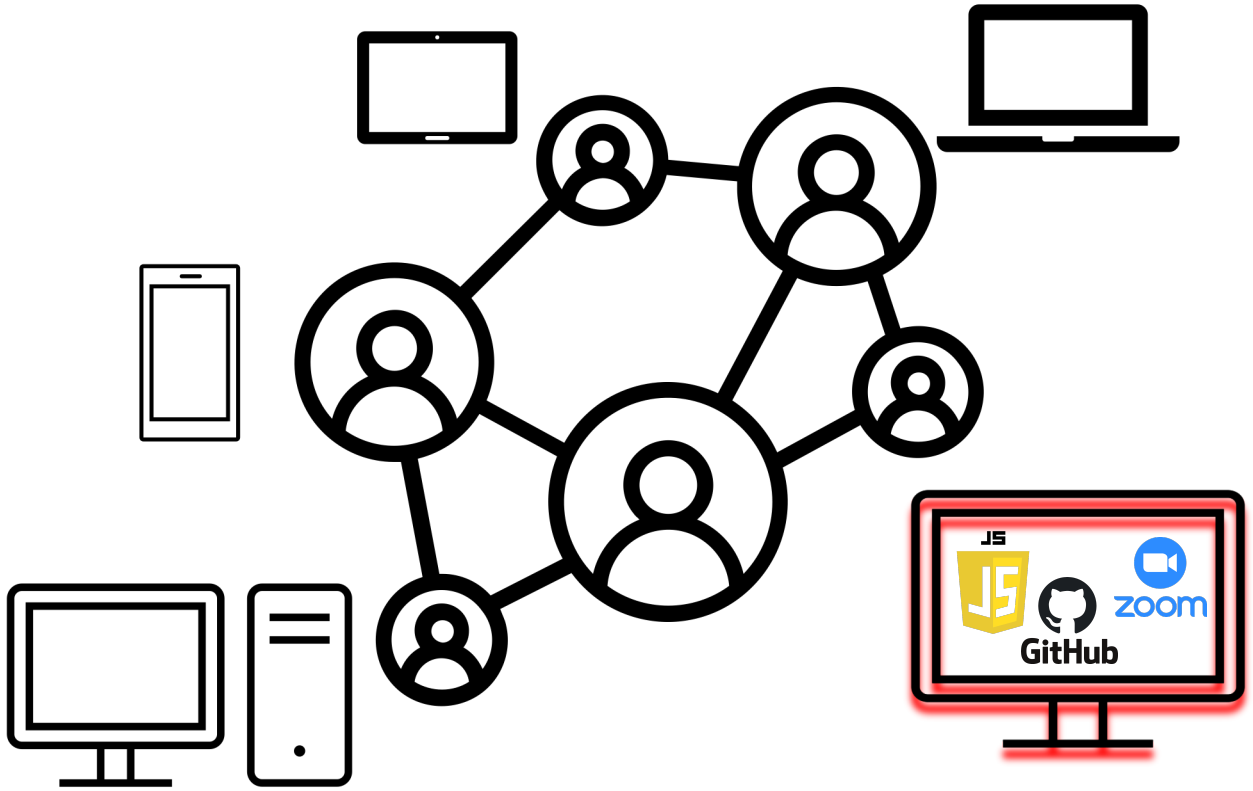
Zcash

MONERO

Dash

VERGE

The Rise of Cryptojacking Malware



Victim

Attacker

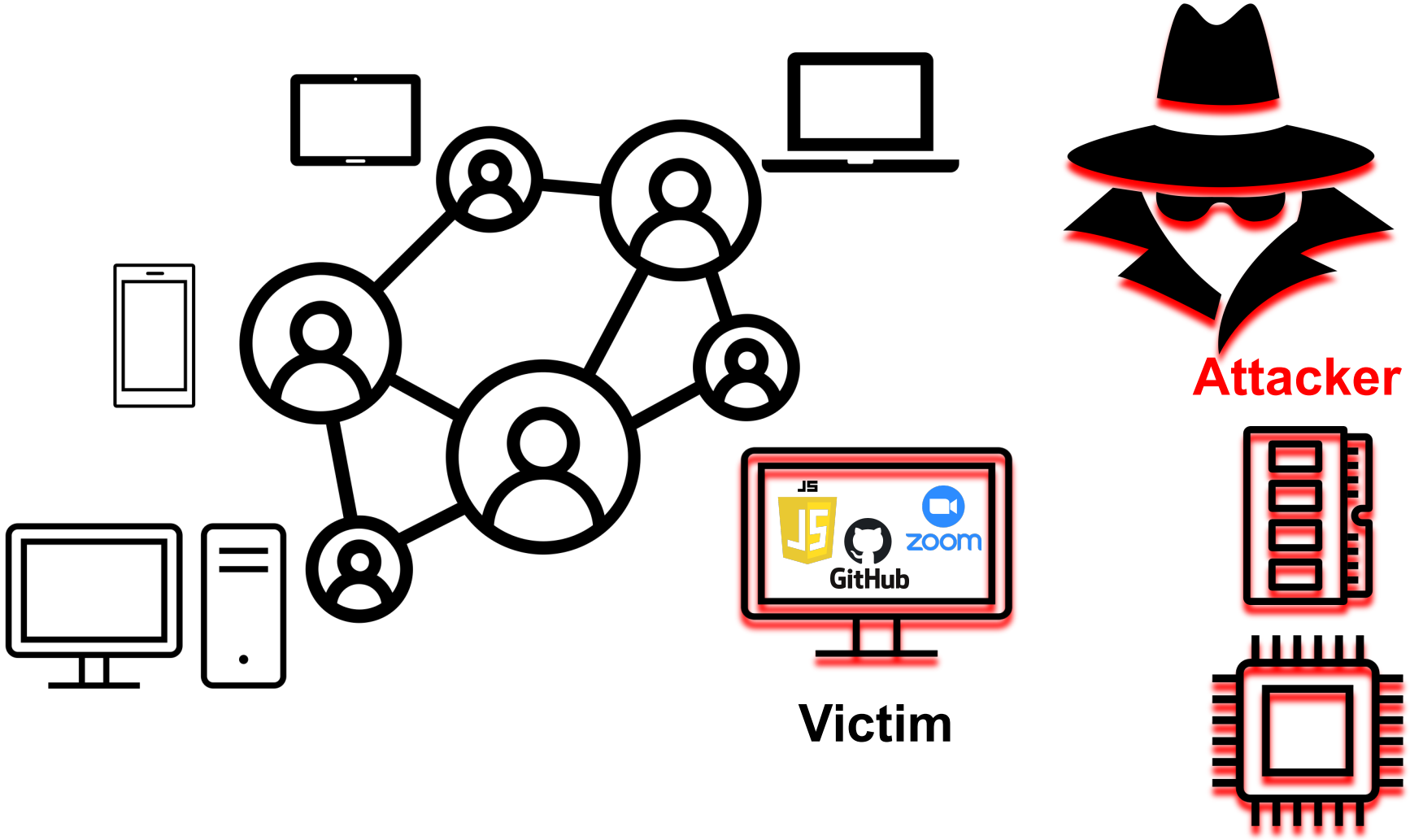
Zcash

MONERO

Dash

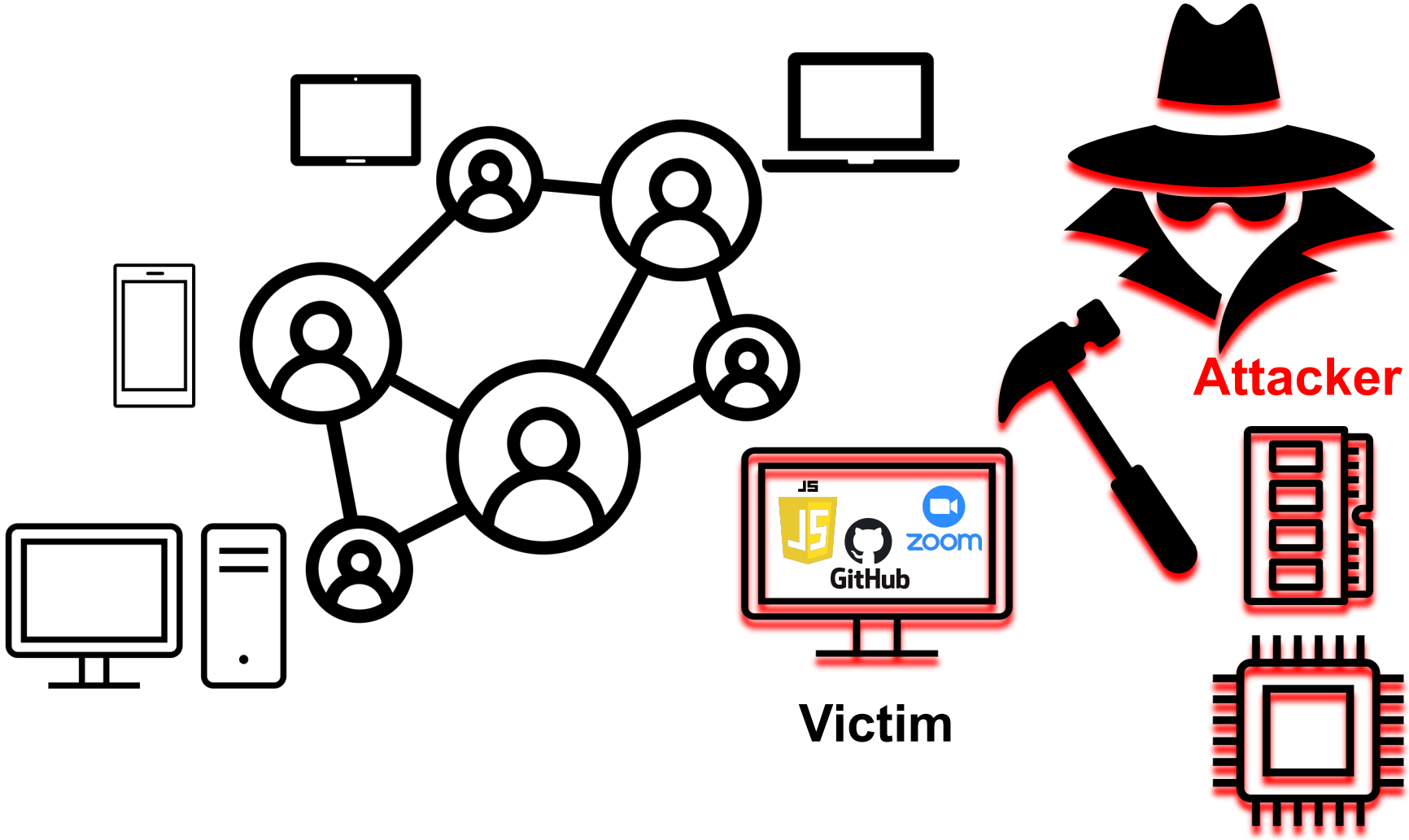
VERGE

The Rise of Cryptojacking Malware



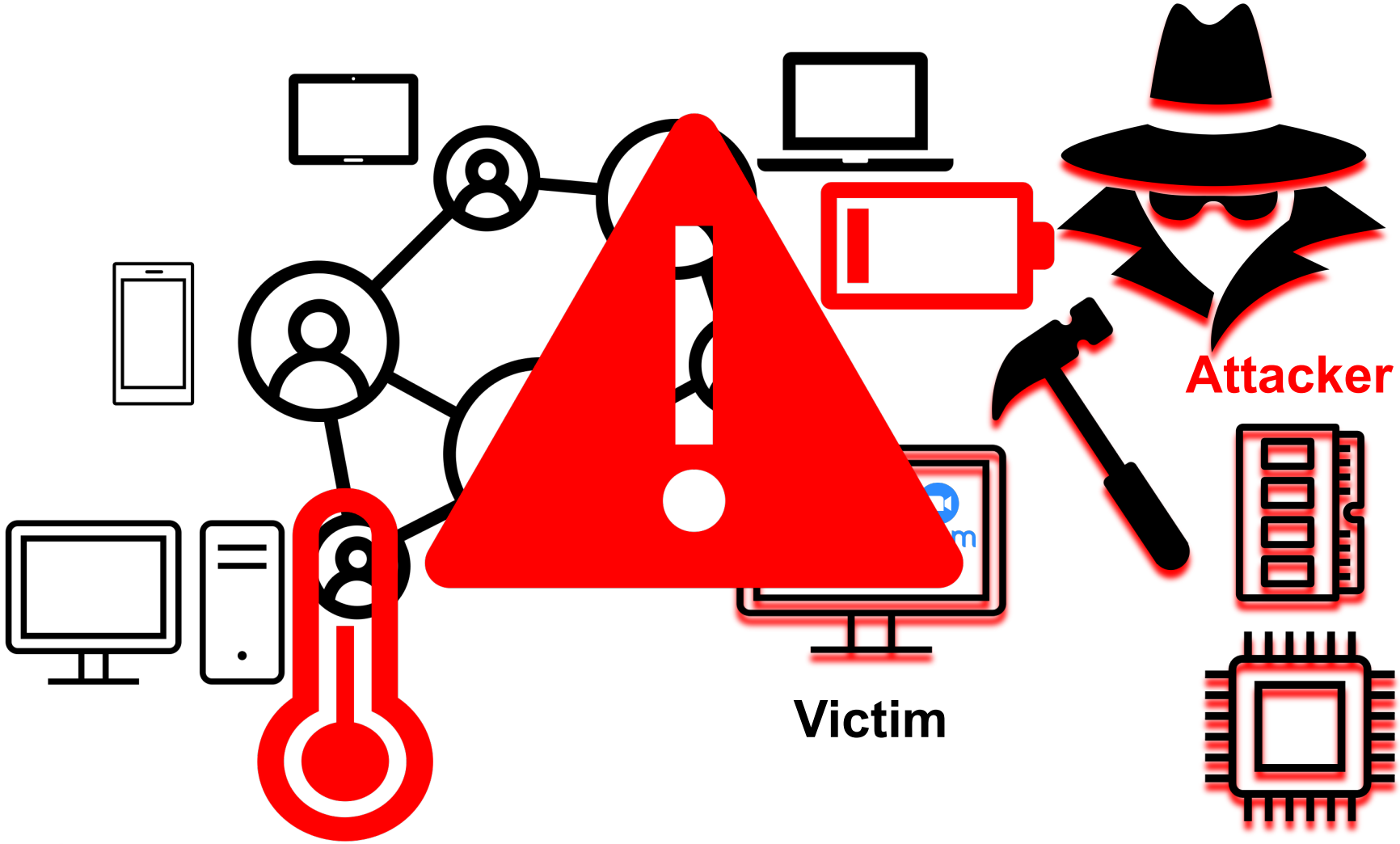
Zcash MONERO Dash VERGE

The Rise of Cryptojacking Malware



Zcash MONERO Dash VERGE

The Rise of Cryptojacking Malware



Zcash MONERO Dash VERGE

Our Work

- Proposes an application agnostic defense that harnesses innovations at the microarchitecture and OS layers

Our Work

- Proposes an application agnostic defense that harnesses innovations at the microarchitecture and OS layers

- Demonstrate that instructions commonly present in cryptographic functions serve as reliable signatures for fingerprinting cryptojacking activity

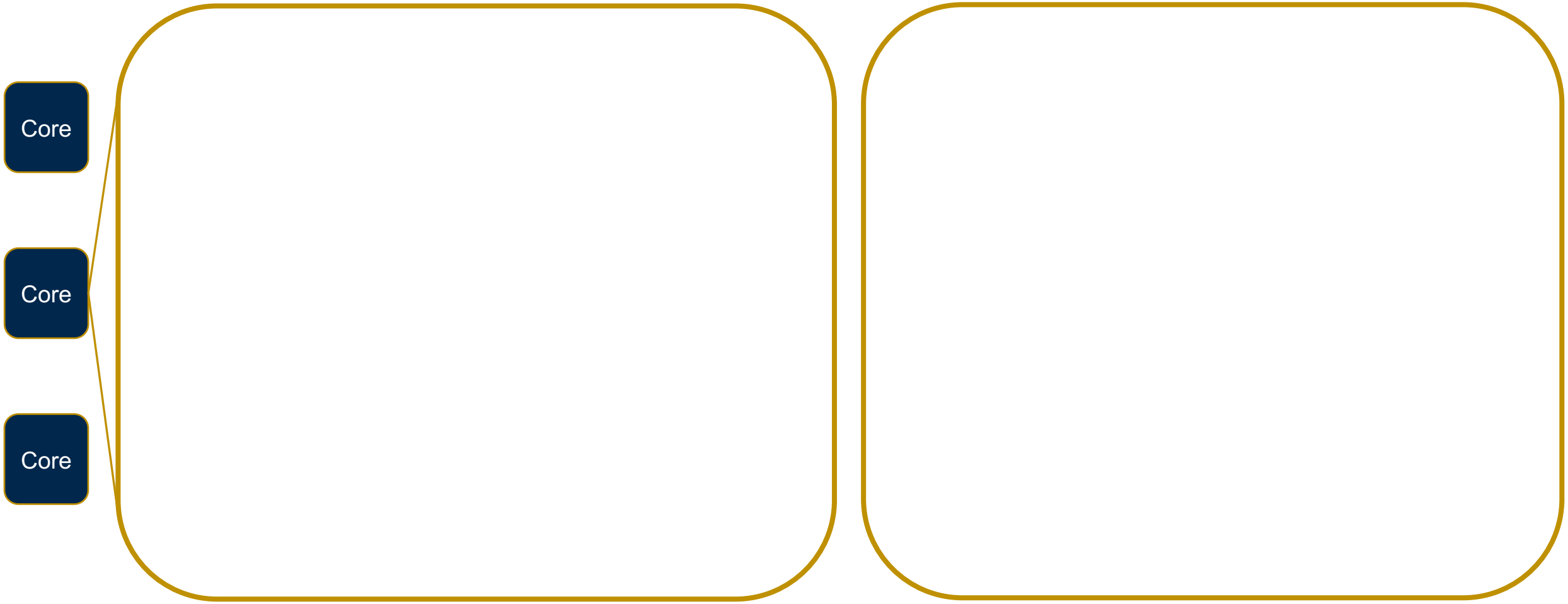
Our Work

- Proposes an application agnostic defense that harnesses innovations at the microarchitecture and OS layers
- Demonstrate that instructions commonly present in cryptographic functions serve as reliable signatures for fingerprinting cryptojacking activity
- Characterize a variety of instruction types across several applications while evaluating their relevance for accurate cryptojacking detection

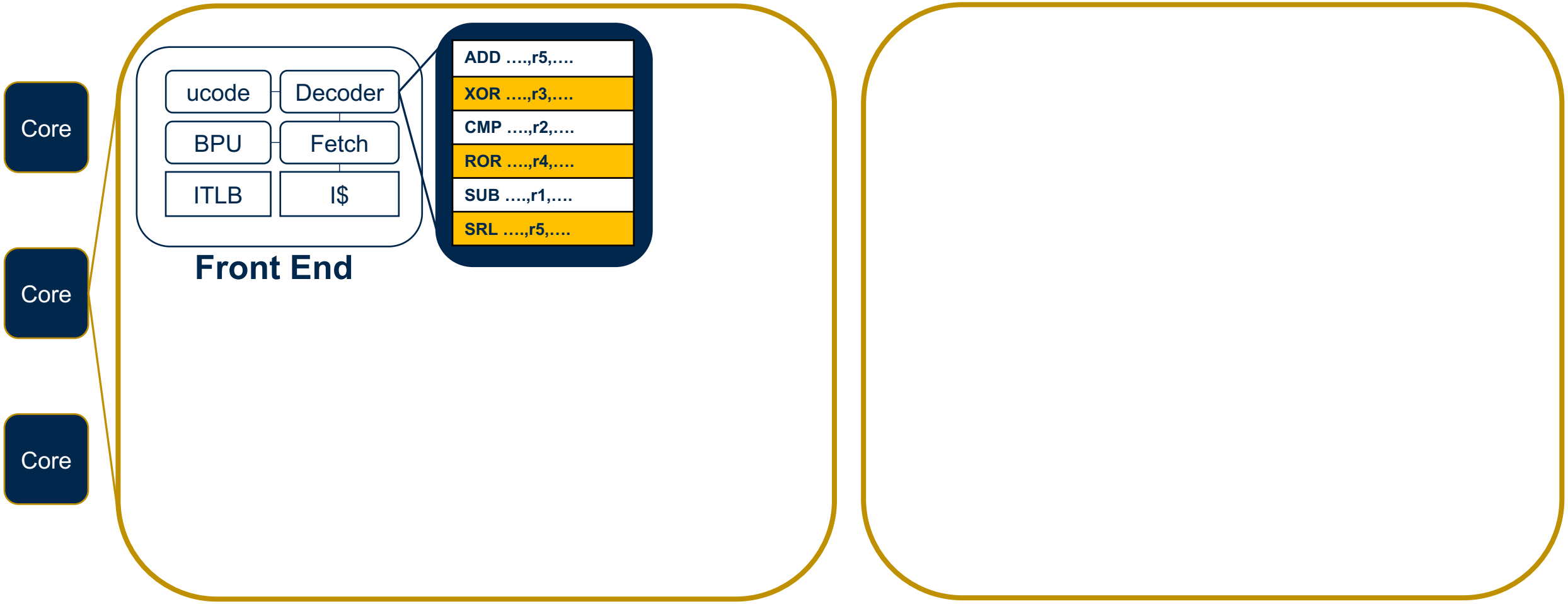
Our Work

- Proposes an application agnostic defense that harnesses innovations at the microarchitecture and OS layers
- Demonstrate that instructions commonly present in cryptographic functions serve as reliable signatures for fingerprinting cryptojacking activity
- Characterize a variety of instruction types across several applications while evaluating their relevance for accurate cryptojacking detection
- Present a design that is robust against code obfuscation, multi-threaded, and throttling attacks while maintaining a low false positive rate

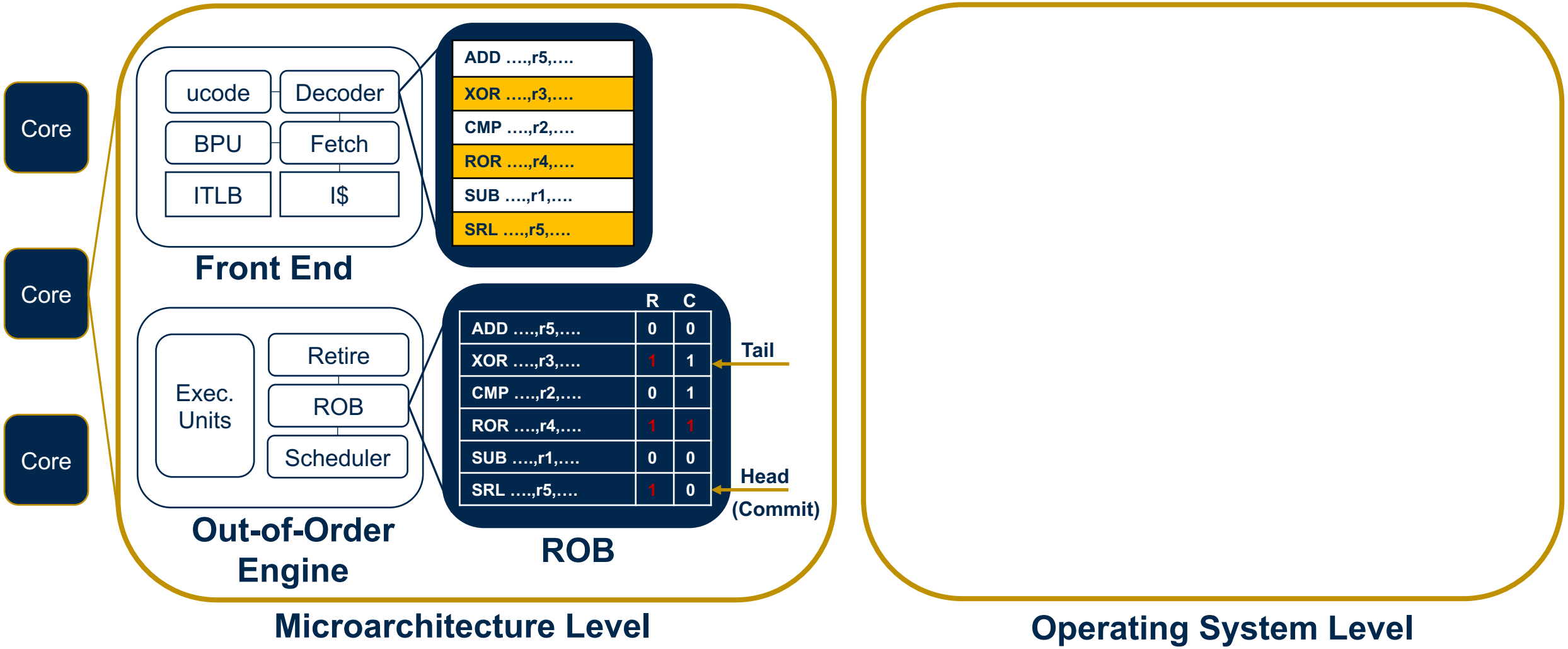
The Cryptojacking Defense System



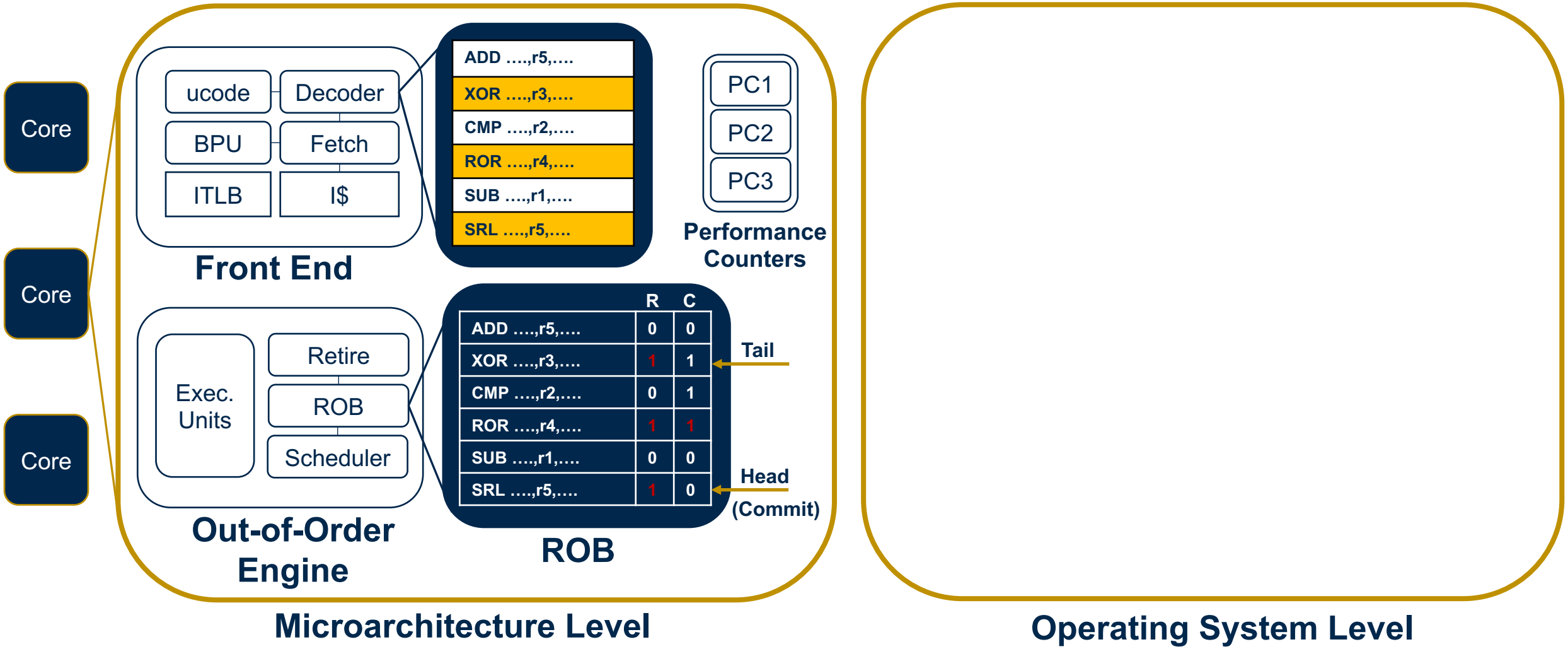
The Cryptojacking Defense System



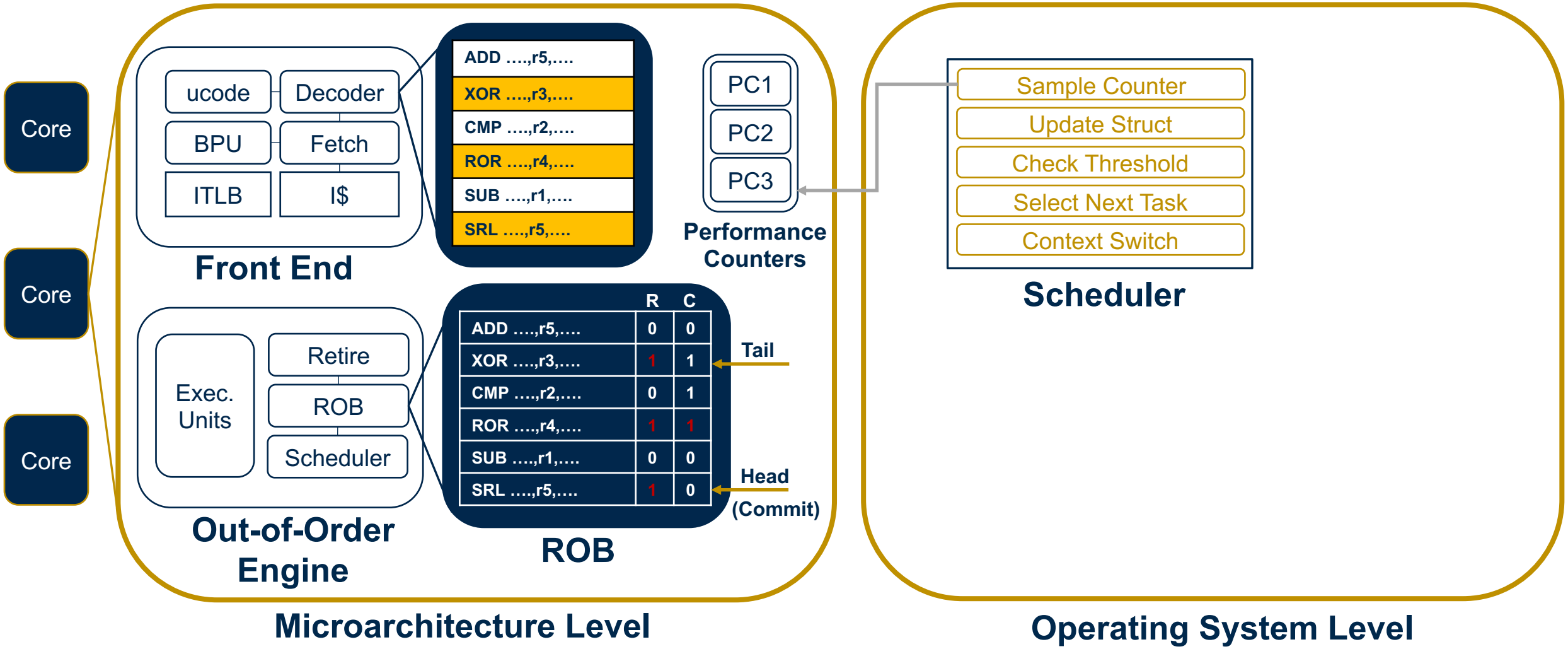
The Cryptojacking Defense System



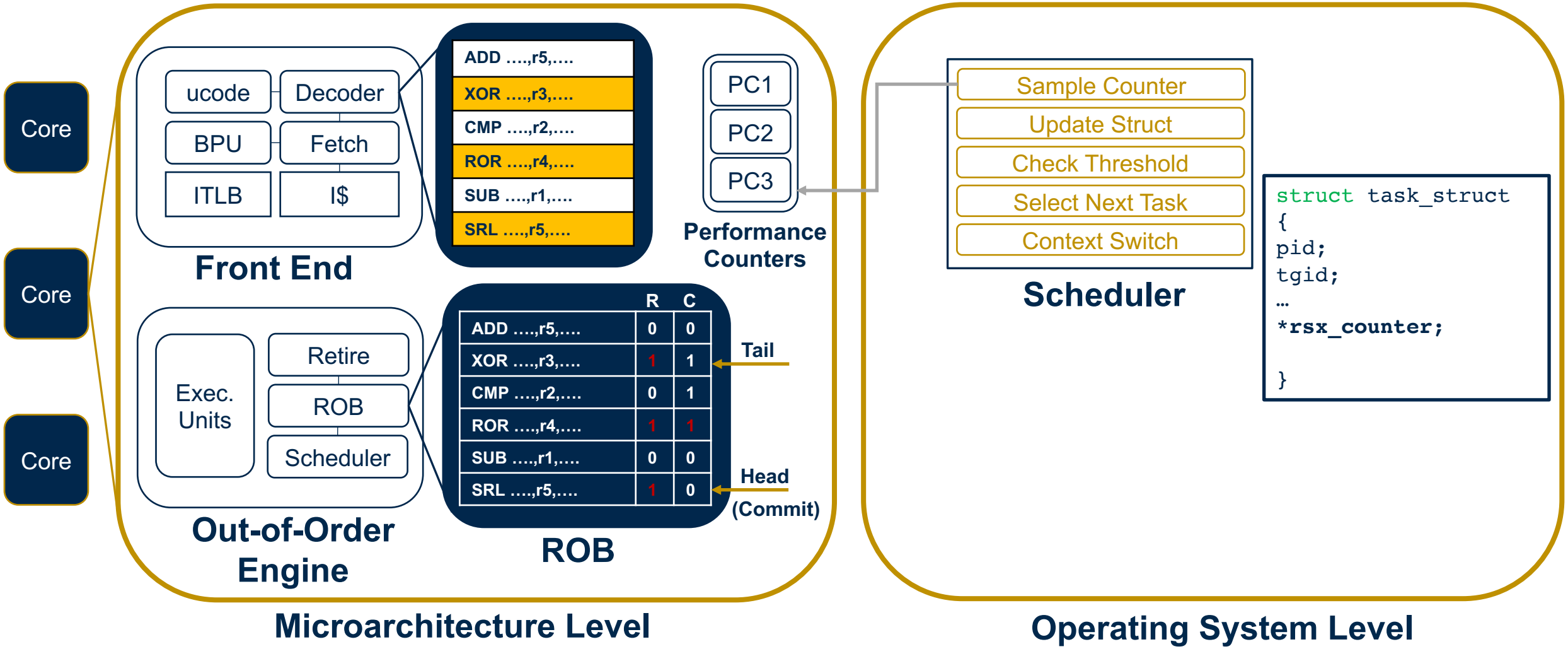
The Cryptojacking Defense System



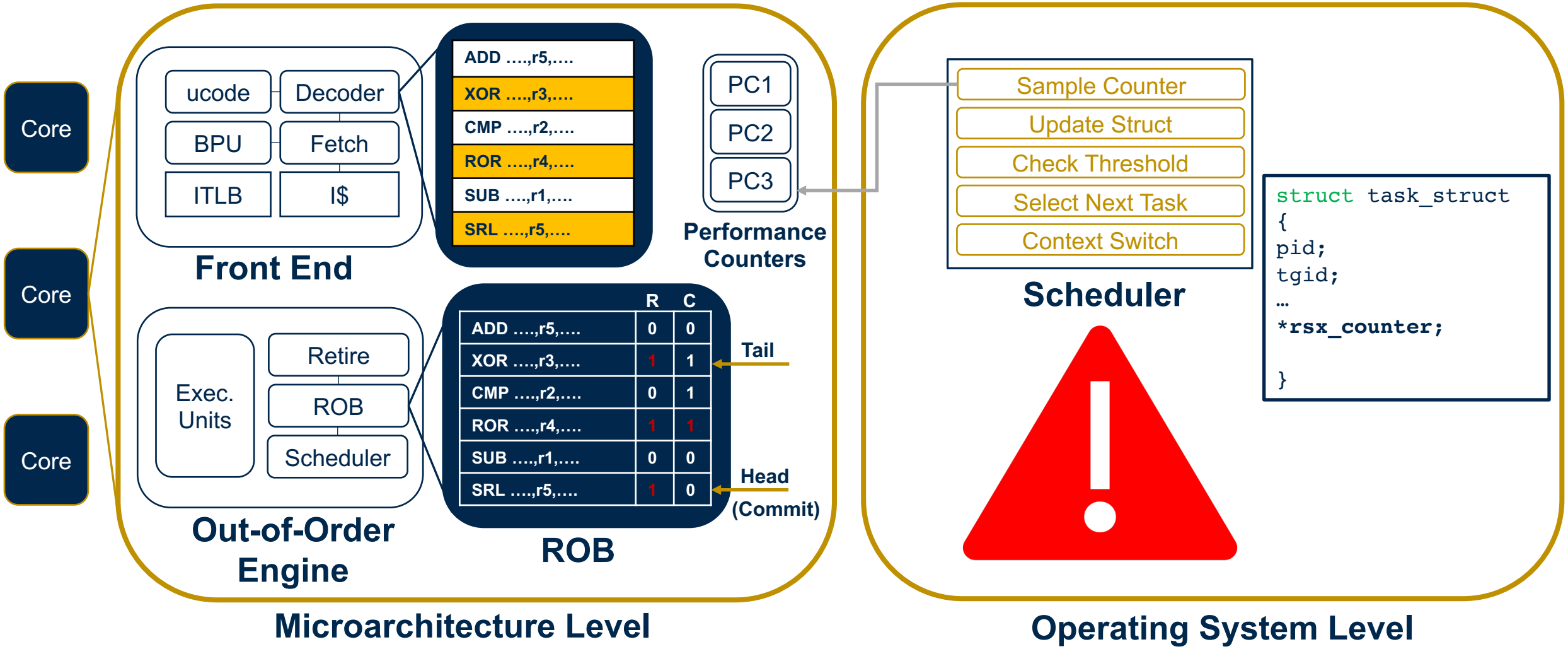
The Cryptojacking Defense System



The Cryptojacking Defense System



The Cryptojacking Defense System



Experimental Framework

- Gem5 Simulator
- Ubuntu OS 16.04
- Linux kernel v4.19.91
- Intel Software Development Emulator

Experimental Framework

- Gem5 Simulator
- Ubuntu OS 16.04
- Linux kernel v4.19.91
- Intel Software Development Emulator
- SPEC2k6 Benchmark Suite
- 150 Standard User Apps
- Crypto-wallets & Decentralized Apps
- Mining Services

Cryptojacking Evasion Techniques

- Code Obfuscation Attacks

$$p = \sim(\sim p)$$

$$P \vee Q = \overline{\overline{P} \wedge \overline{Q}}$$

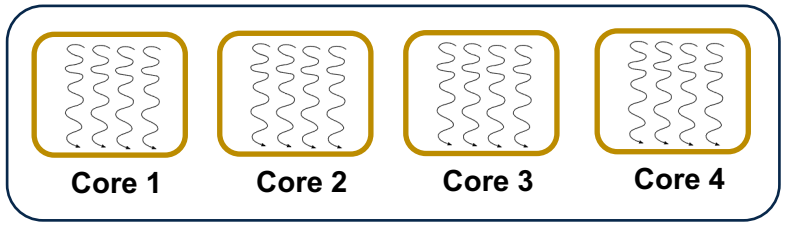
Cryptojacking Evasion Techniques

- Code Obfuscation Attacks

$$p = \sim(\sim p)$$

$$P \vee Q = \overline{\overline{P} \wedge \overline{Q}}$$

- Multi-threaded Attacks



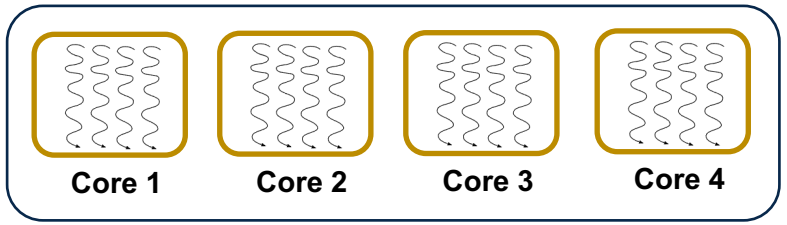
Cryptojacking Evasion Techniques

- Code Obfuscation Attacks

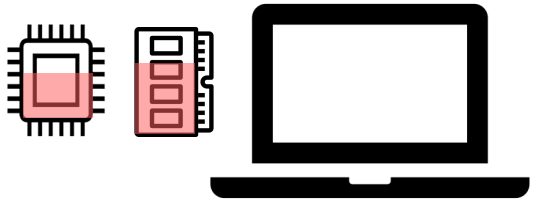
$$p = \sim(\sim p)$$

$$P \vee Q = \overline{\overline{P} \wedge \overline{Q}}$$

- Multi-threaded Attacks



- Throttling Attacks



Mitigating Code Obfuscation Attacks

- Substitute Rotate Instructions

$$R_l^n = S_l^n \vee S_r^{64-n}$$

$$R_r^n = S_r^n \vee S_l^{64-n}$$

Mitigating Code Obfuscation Attacks

- Substitute Rotate Instructions

$$R_l^n = S_l^n \vee S_r^{64-n}$$

$$R_r^n = S_r^n \vee S_l^{64-n}$$

- Track Rotate, Shift, and XOR (RSX) instructions

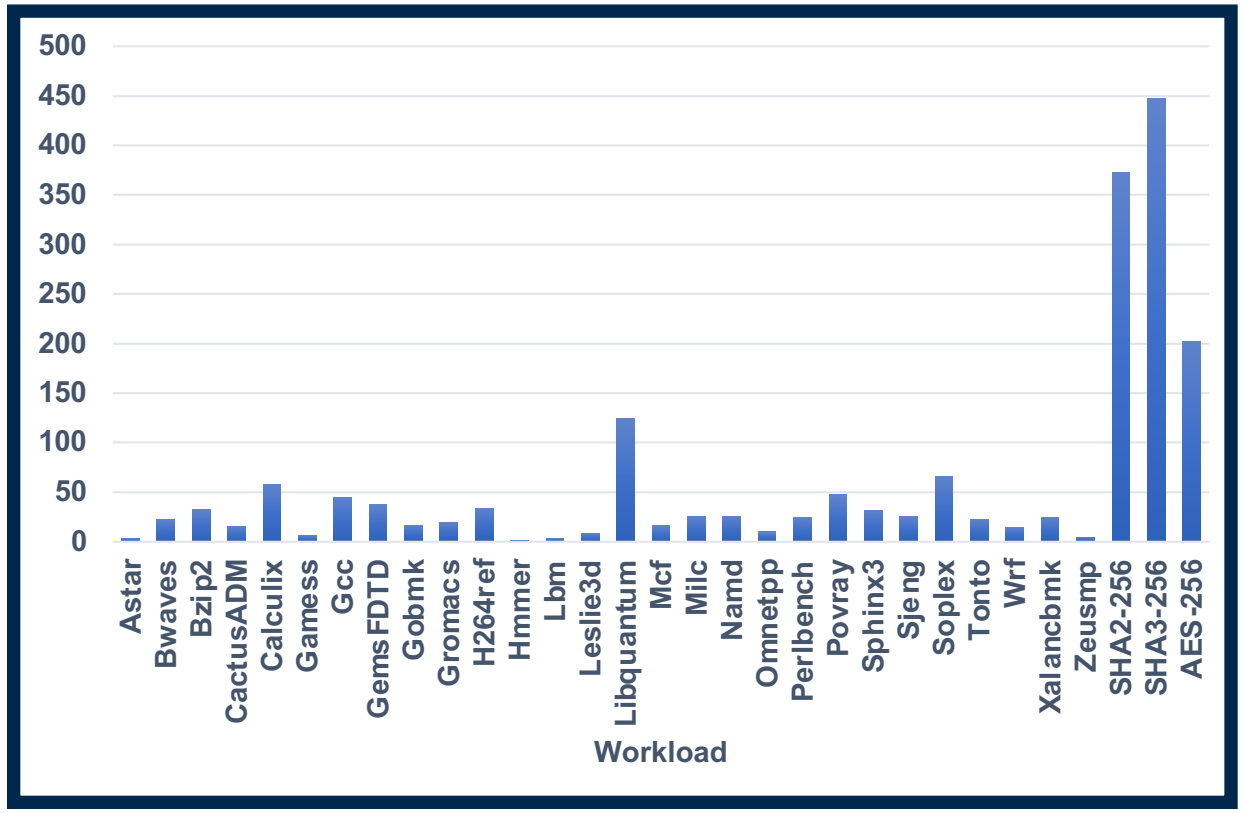
Mitigating Code Obfuscation Attacks

- Substitute Rotate Instructions

$$R_l^n = S_l^n \vee S_r^{64-n}$$

$$R_r^n = S_r^n \vee S_l^{64-n}$$

- Track Rotate, Shift, and XOR (RSX) instructions



Total number of Rotate, Shift, and XOR instructions (Millions)

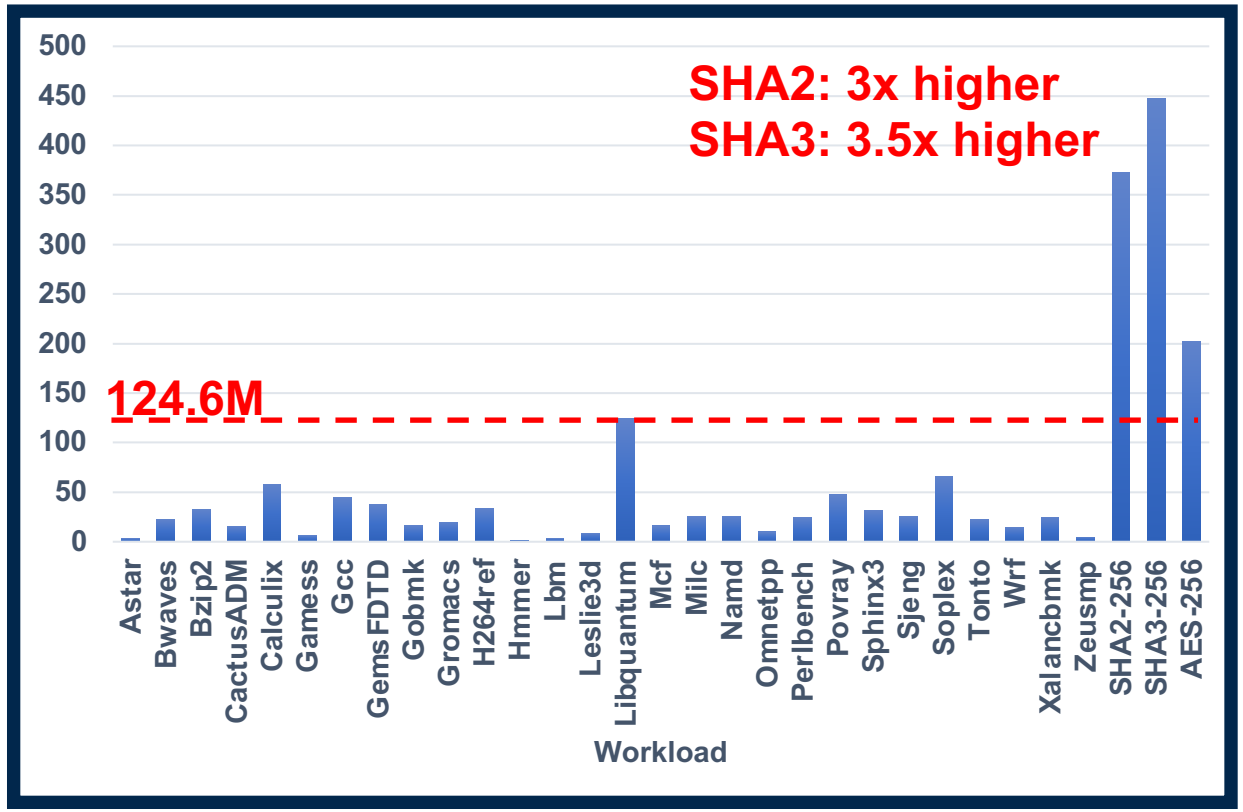
Mitigating Code Obfuscation Attacks

- Substitute Rotate Instructions

$$R_l^n = S_l^n \vee S_r^{64-n}$$

$$R_r^n = S_r^n \vee S_l^{64-n}$$

- Track Rotate, Shift, and XOR (RSX) instructions



Total number of Rotate, Shift, and XOR instructions (Millions)

Mitigating Code Obfuscation Attacks

- Substitute Exclusive-OR instructions

$$A \oplus B = A\bar{B} \vee \bar{A}B$$

Mitigating Code Obfuscation Attacks

- Substitute Exclusive-OR instructions

$$A \oplus B = A\bar{B} \vee \bar{A}B$$

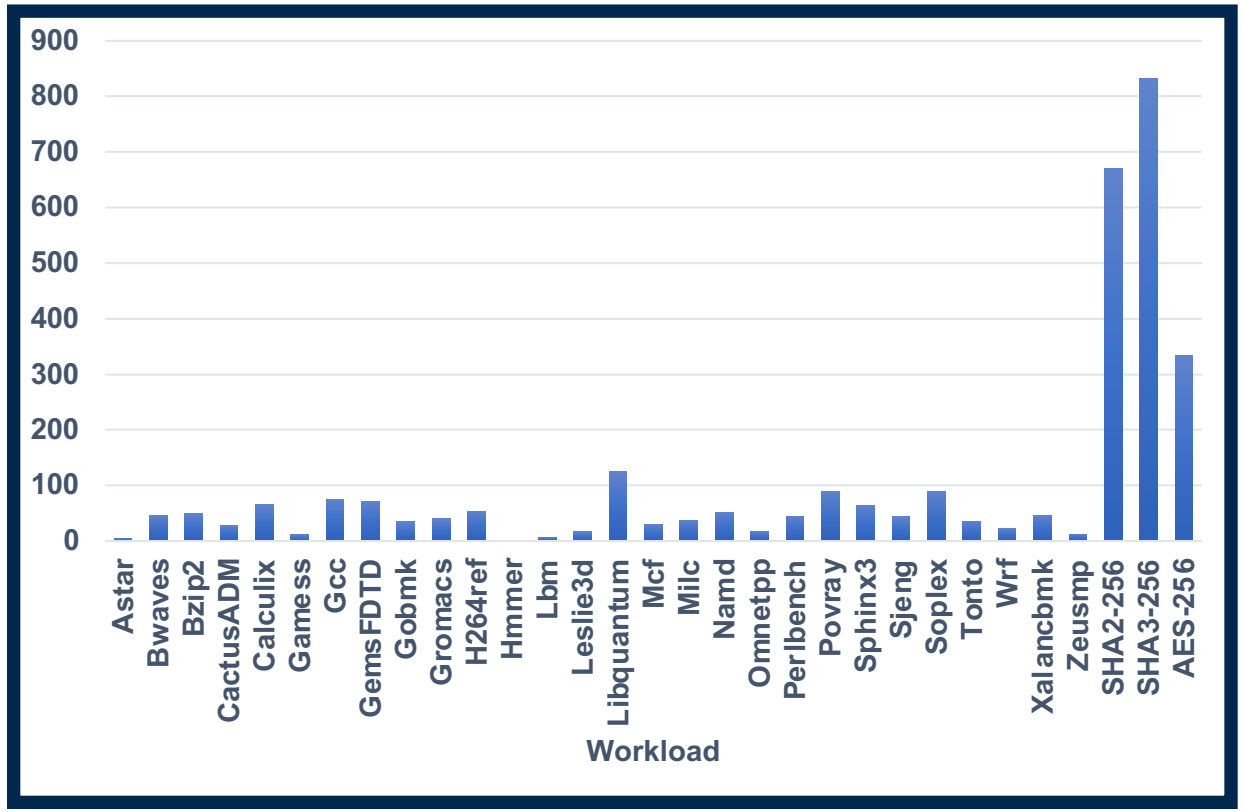
- Track Rotate, Shift, XOR, and OR (RSXO) instructions

Mitigating Code Obfuscation Attacks

- Substitute Exclusive-OR instructions

$$A \oplus B = A\bar{B} \vee \bar{A}B$$

- Track Rotate, Shift, XOR, and OR (RSXO) instructions



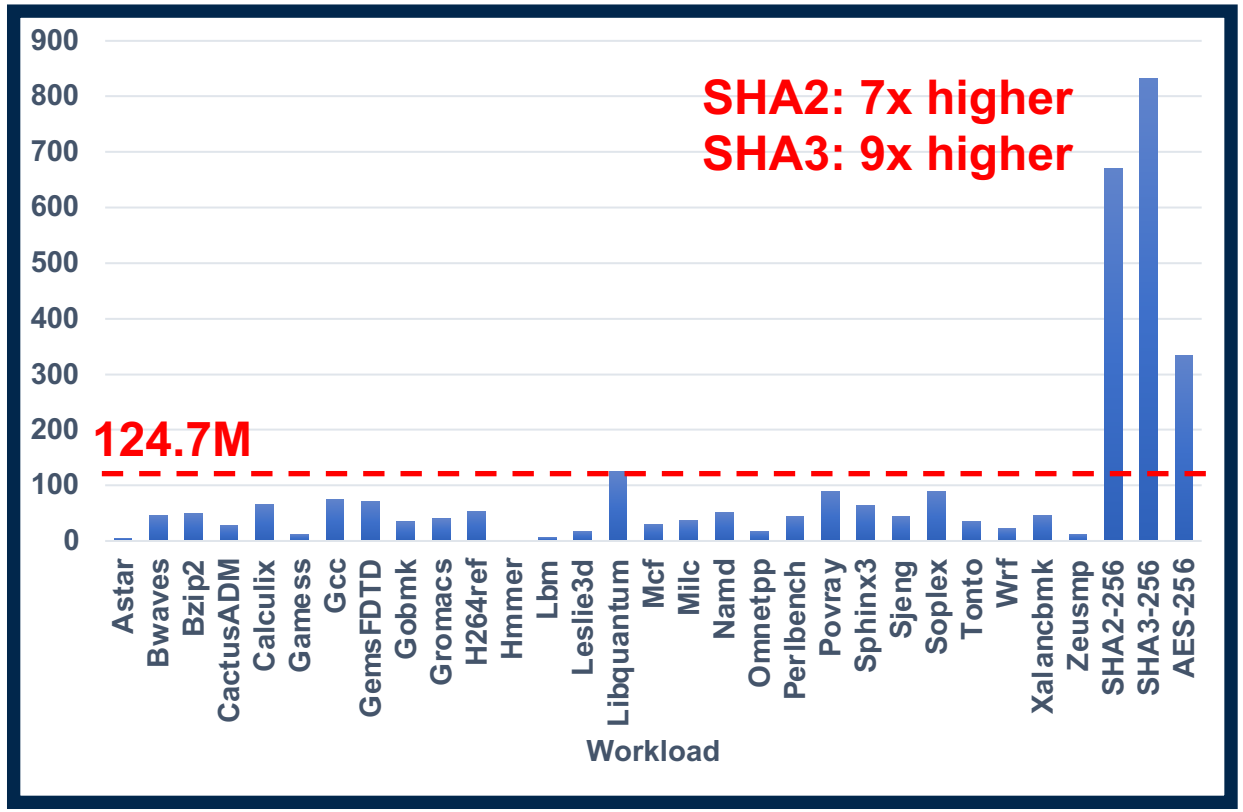
Total number of Rotate, Shift, XOR, and OR instructions (Millions)

Mitigating Code Obfuscation Attacks

- Substitute Exclusive-OR instructions

$$A \oplus B = A\bar{B} \vee \bar{A}B$$

- Track Rotate, Shift, XOR, and OR (RSXO) instructions



Total number of Rotate, Shift, XOR, and OR instructions (Millions)

Mitigating Code Obfuscation Attacks

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
     threads: 1,
     autoThreads: false,
     throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

Mitigating Code Obfuscation Attacks

- Distribute workload across multiple threads

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

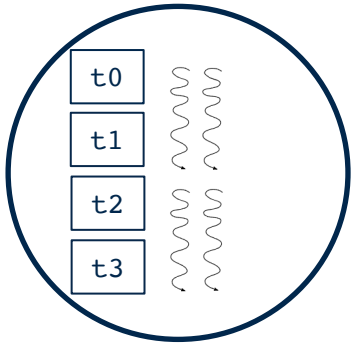
Mitigating Code Obfuscation Attacks

- Distribute workload across multiple threads
- Allocate threads to different cores

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

Mitigating Multi-threaded Attacks



Multi-threaded Program

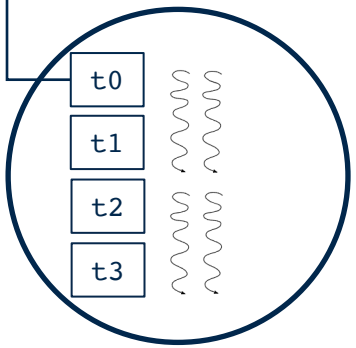
```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

Mitigating Multi-threaded Attacks

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```



Multi-threaded Program

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

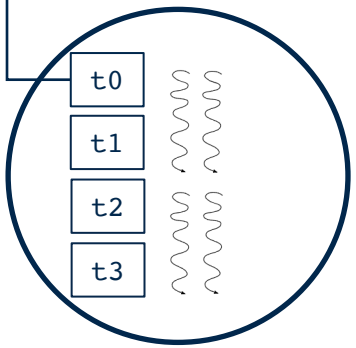
Malicious Code

Mitigating Multi-threaded Attacks

```
struct tgid_rsx_t
{
rsx_count;
tcount;
}
```

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```



Multi-threaded Program

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

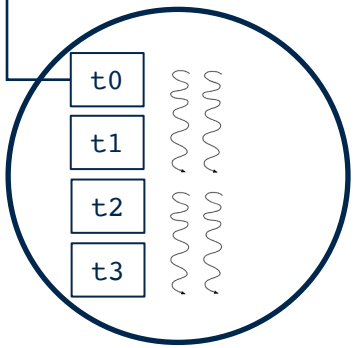
Malicious Code

Mitigating Multi-threaded Attacks

```
struct tgid_rsx_t
{
rsx_count;
tcount;
}
```

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```



Multi-threaded Program

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

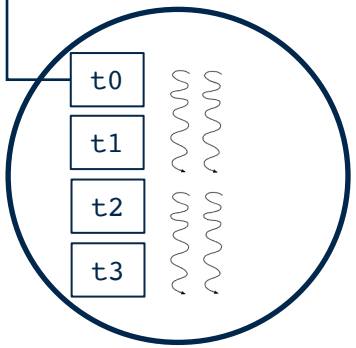
Malicious Code

Mitigating Multi-threaded Attacks

```
struct tgid_rsx_t
{
rsx_count;
tcount;
}
```

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```



Multi-threaded Program

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

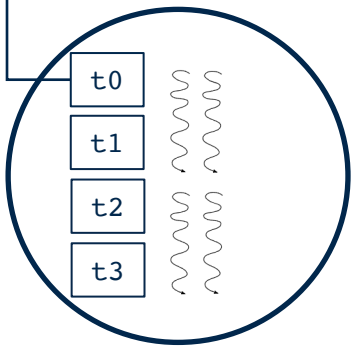
Malicious Code

Mitigating Multi-threaded Attacks

```
struct tgid_rsx_t
{
rsx_count;
tcount;
}
```

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```



Multi-threaded Program

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

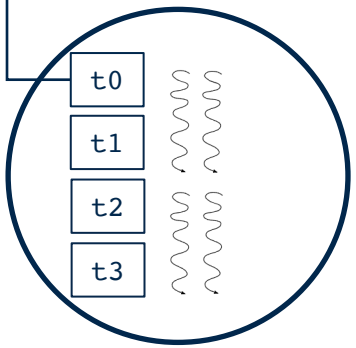
Malicious Code

Mitigating Multi-threaded Attacks

```
struct tgid_rsx_t
{
rsx_count;
tcount;
}
```

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```



Multi-threaded Program

```
long _do_fork(){
…
If (p->tgid != parent->tgid) {
create_tgid_crypt_struct(p);
}
else {
p->rsx_ptr = parent->rsx_ptr;
}
wake_up_new_task(p);
…
}
```

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

Mitigating Multi-threaded Attacks

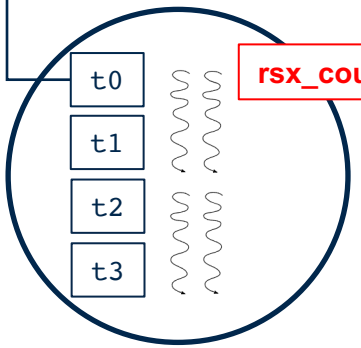
```
struct tgid_rsx_t
{
rsx_count;
tcount;
}
```

rsx_count = 100

tcount = 1

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```



rsx_count = 100

Multi-threaded Program

```
long _do_fork(){
…
If (p->tgid != parent->tgid) {
create_tgid_crypt_struct(p);
}
else {
p->rsx_ptr = parent->rsx_ptr;
}
wake_up_new_task(p);
…
}
```

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

Mitigating Multi-threaded Attacks

```
struct tgid_rsx_t
{
rsx_count;
tcount;
}
```

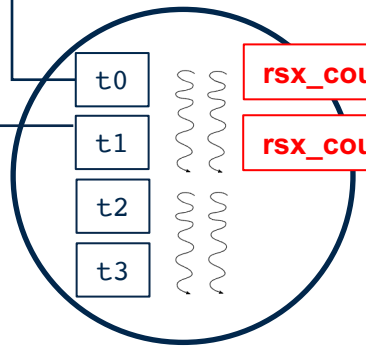
rsx_count = 300

tcount = 2

```
struct task_struct
{
```

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```



rsx_count = 100

rsx_count = 200

Multi-threaded Program

```
long _do_fork(){
…
If (p->tgid != parent->tgid) {
create_tgid_crypt_struct(p);
}
else {
p->rsx_ptr = parent->rsx_ptr;
}
wake_up_new_task(p);
…
}
```

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

Mitigating Multi-threaded Attacks

```
struct tgid_rsx_t
{
rsx_count;
tcount;
}
```

rsx_count = 600

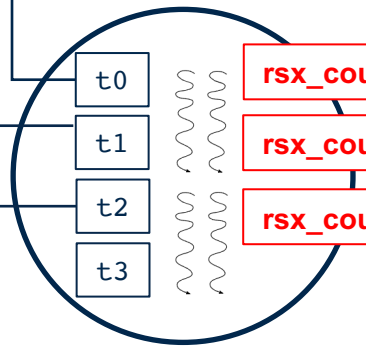
tcount = 3

```
struct task_struct
{
```

```
struct task_struct
{
```

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```



rsx_count = 100

rsx_count = 200

rsx_count = 300

Multi-threaded Program

```
long _do_fork(){
…
If (p->tgid != parent->tgid) {
create_tgid_crypt_struct(p);
}
else {
p->rsx_ptr = parent->rsx_ptr;
}
wake_up_new_task(p);
…
}
```

```
ipt>
miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

Mitigating Multi-threaded Attacks

```
struct tgid_rsx_t
{
rsx_count;
tcount;
}
```

rsx_count = 1000

tcount = 4

```
struct task_struct
{
```

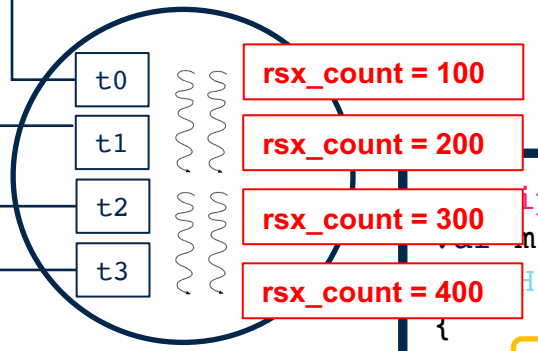
```
struct task_struct
{
```

```
struct task_struct
{
```

```
struct task_struct
{
pid;
tgid;
…
tgid_rsx_t *rsx_ptr;

}
```

```
long _do_fork(){
…
If (p->tgid != parent->tgid) {
create_tgid_crypt_struct(p);
}
else {
p->rsx_ptr = parent->rsx_ptr;
}
wake_up_new_task(p);
…
}
```



rsx_count = 100

rsx_count = 200

rsx_count = 300

rsx_count = 400

Multi-threaded Program

```
ipt>
miner =
Hive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
    threads: 4,
    autoThreads: false,
    throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

Mitigating Throttling Attacks

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
     threads: 4,
     autoThreads: false,
     throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

Mitigating Throttling Attacks

- Use limited amount of the system's power

```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
     threads: 4,
     autoThreads: false,
     throttle: 0.3,
});
    miner.start();
</script >
```

Malicious Code

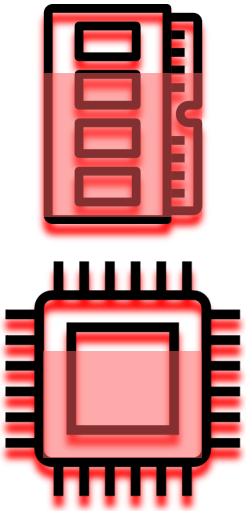
Mitigating Throttling Attacks

- Use limited amount of the system's power

- Common throttling rate is 30%

70%

processing power

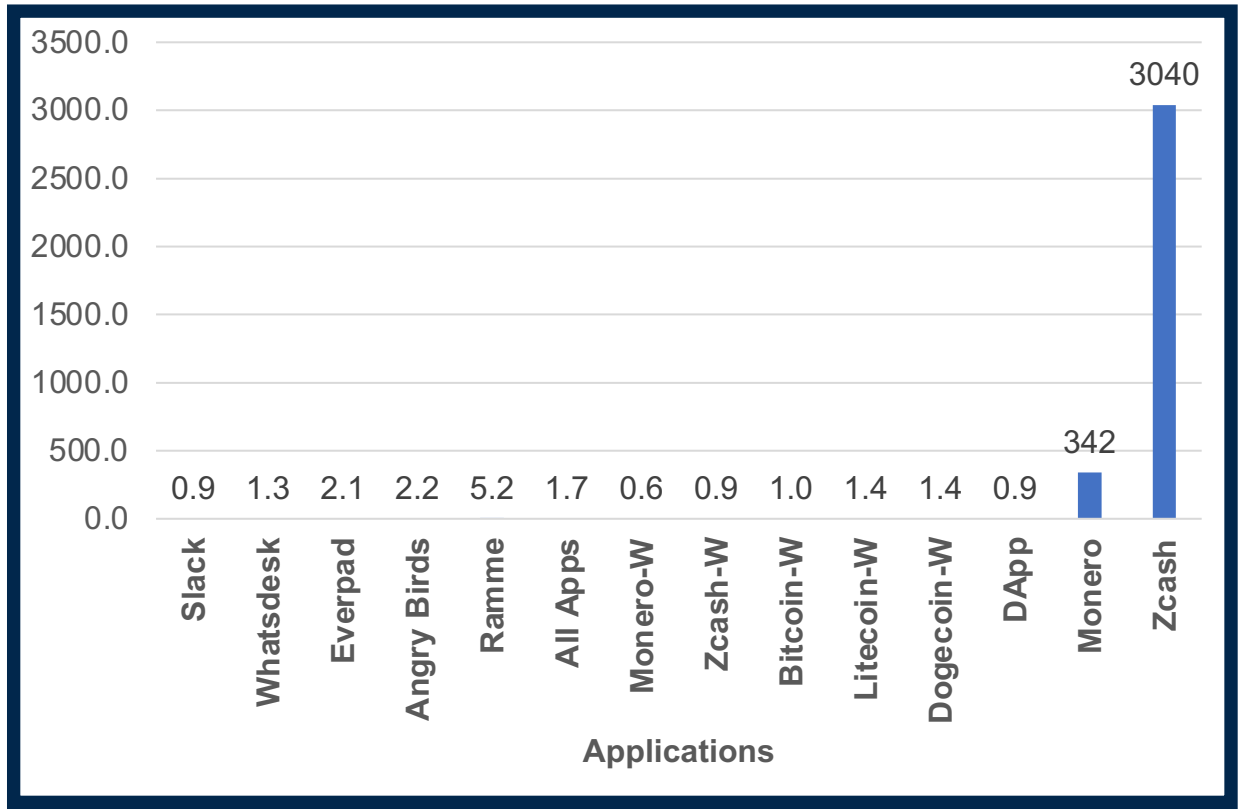


```
<script>
var miner =
CoinHive.Anonymous('SK_OBKc7fs7W3bJHMsAmdHzJ',
{
      threads: 4,
      autoThreads: false,
      throttle: 0.3,
});
     miner.start();
</script >
```

Malicious Code

Mitigating Throttling Attacks

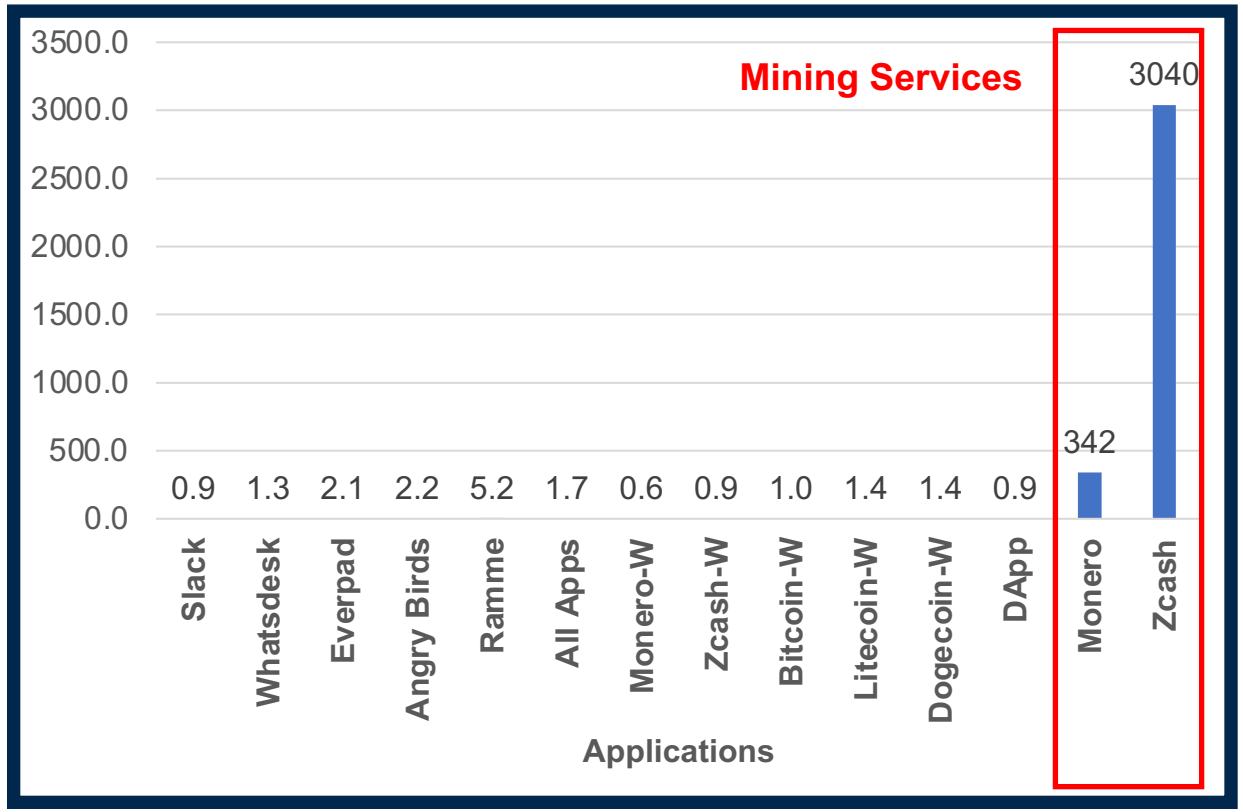
- Ramme has the highest RSX count (5.2B) upon tested apps
- Monero mining service has 65x higher RSX count than Ramme
- Zcash mining service has $3x10^3$ higher RSX count than Ramme
- Non-mining Crypto Apps have an avg. of 1B RSX count



Total number of Rotate, Shift, XOR instructions (Billions)

Mitigating Throttling Attacks

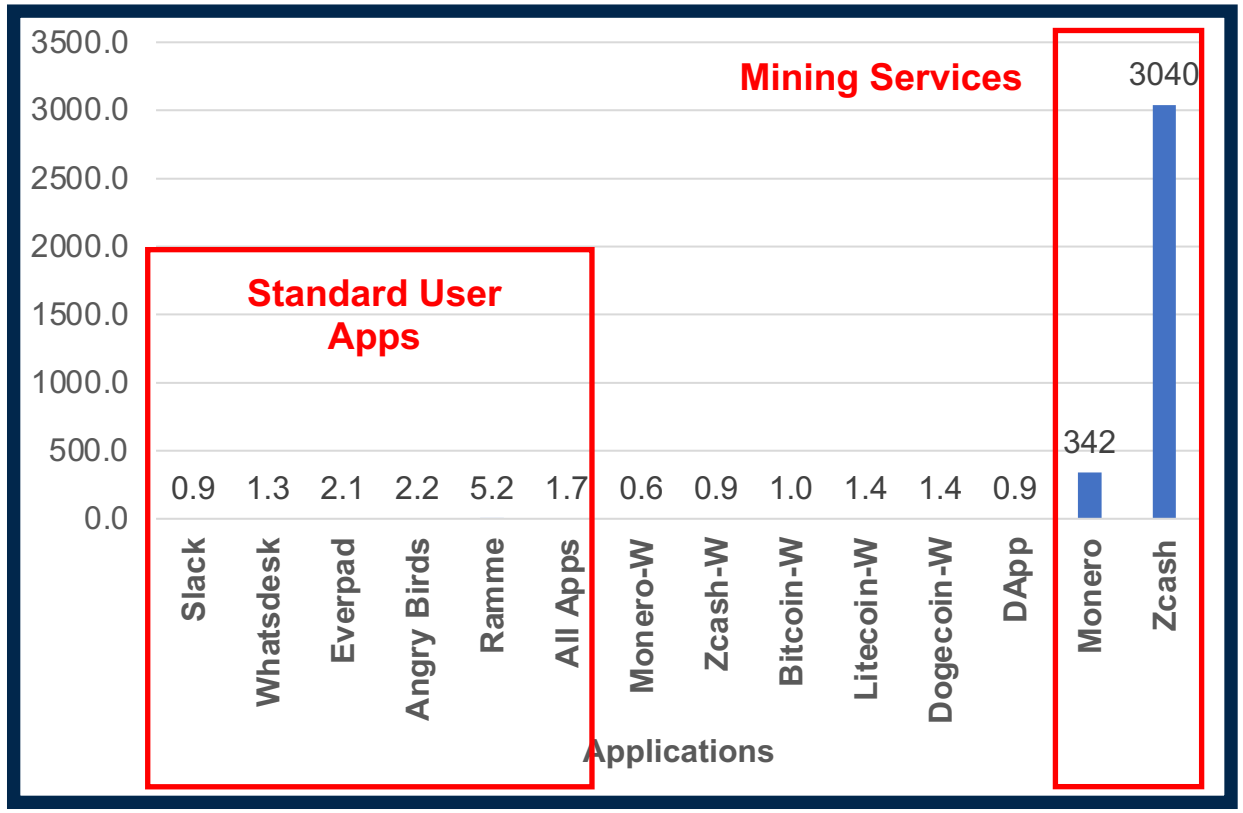
- Ramme has the highest RSX count (5.2B) upon tested apps
- Monero mining service has 65x higher RSX count than Ramme
- Zcash mining service has $3x10^3$ higher RSX count than Ramme
- Non-mining Crypto Apps have an avg. of 1B RSX count



Total number of Rotate, Shift, XOR instructions (Billions)

Mitigating Throttling Attacks

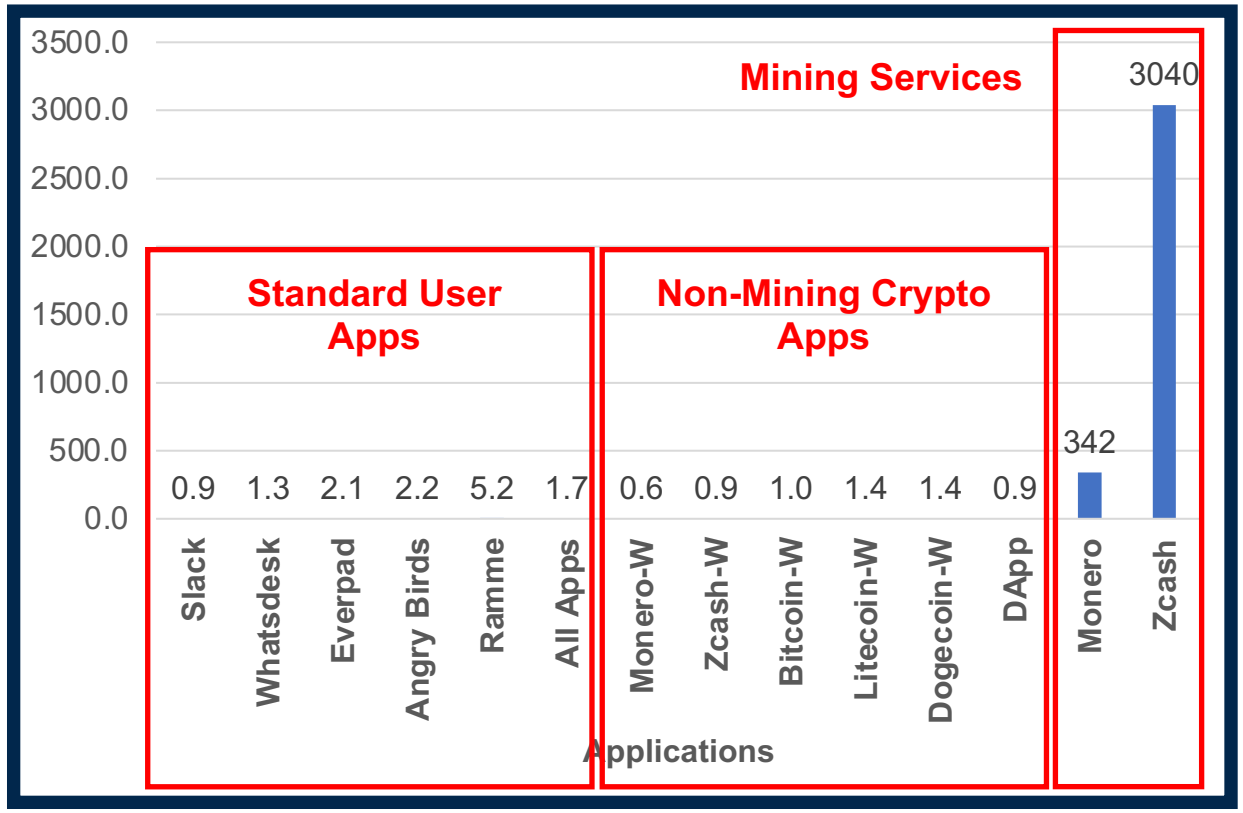
- Ramme has the highest RSX count (5.2B) upon tested apps
- Monero mining service has 65x higher RSX count than Ramme
- Zcash mining service has $3x10^3$ higher RSX count than Ramme
- Non-mining Crypto Apps have an avg. of 1B RSX count



Total number of Rotate, Shift, XOR instructions (Billions)

Mitigating Throttling Attacks

- Ramme has the highest RSX count (5.2B) upon tested apps
- Monero mining service has 65x higher RSX count than Ramme
- Zcash mining service has $3x10^3$ higher RSX count than Ramme
- Non-mining Crypto Apps have an avg. of 1B RSX count

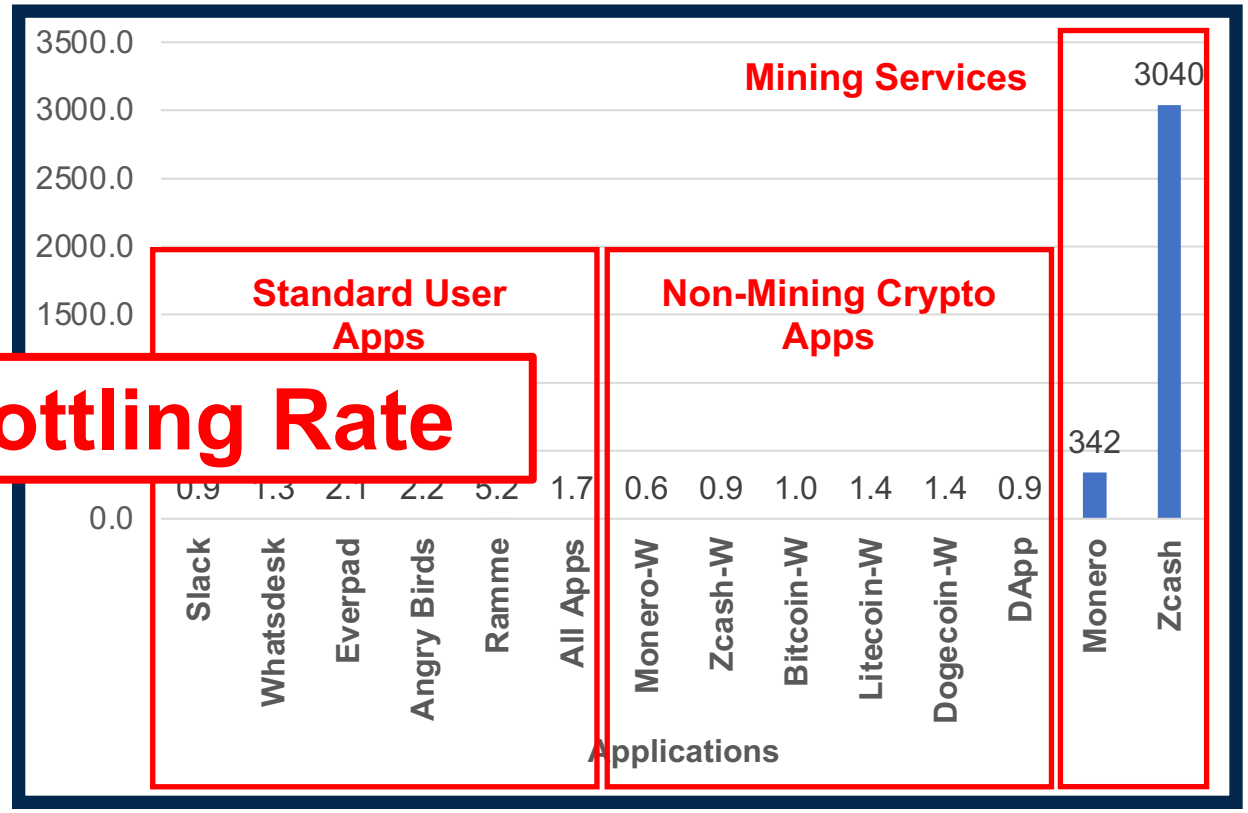


Total number of Rotate, Shift, XOR instructions (Billions)

Mitigating Throttling Attacks

- Ramme has the highest RSX count (5.2B) upon tested apps
- Monero mining service has 65x higher RSX count
- Zcash mining service has $3x10^3$ higher RSX count than Ramme
- Non-mining Crypto Apps have an avg. of 1B RSX count

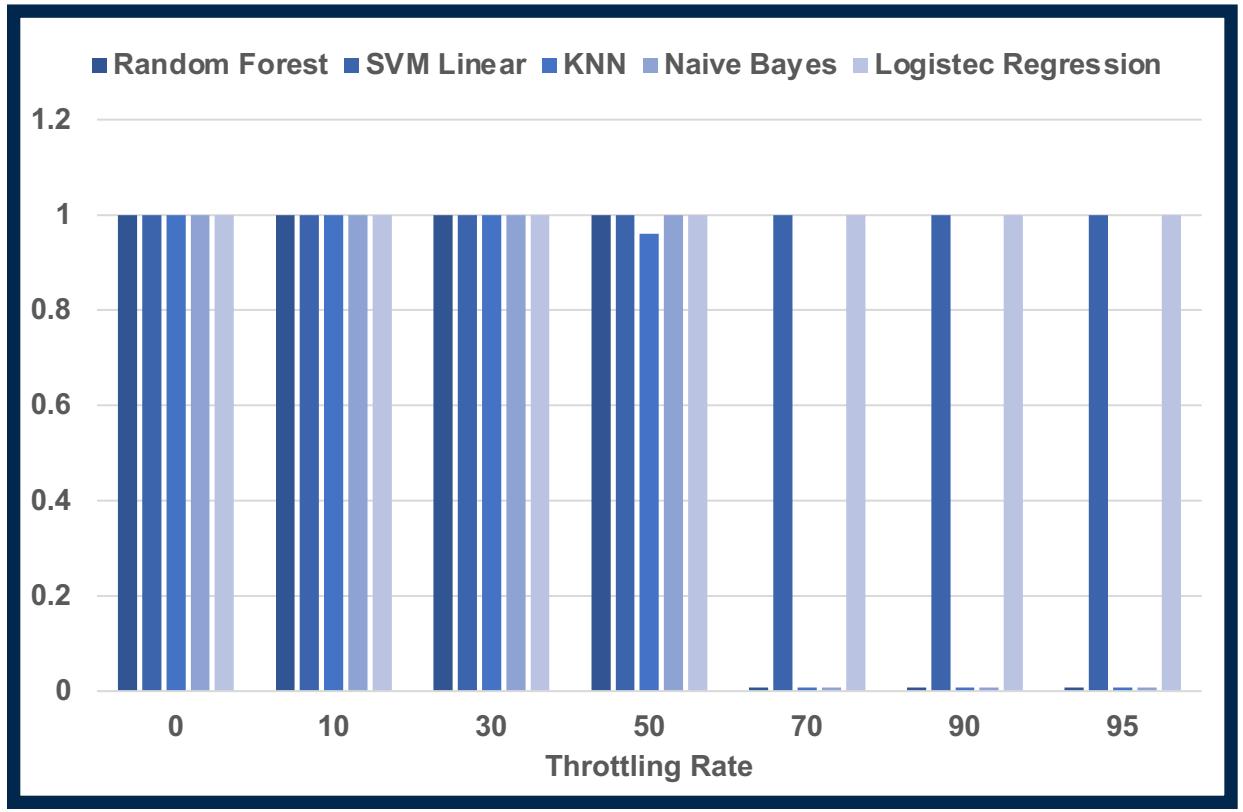
> 50% Throttling Rate



Total number of Rotate, Shift, XOR instructions (Billions)

Mitigating Throttling Attacks

- Principle Components Analysis
- Reduce feature set from 527 to 11
- SVM and Logistic Regression: 100% detection rate
- SVM – Less than 2% False Positives
- Logistic Regression – 40% False Positives



Detection rate as a function of throttling rate

Application Agnostic Solution

Code Obfuscation Attacks

Multi-threaded Attacks

Throttling Attacks

100%

True Positive Rate

100%

True Positive Rate

<2%

False Positive Rate

100%

True Positive Rate

<2%

False Positive Rate

<1%

Overhead

Thank you!

Nada Lachtar
nlachtar@umich.edu