CLOCKWORK

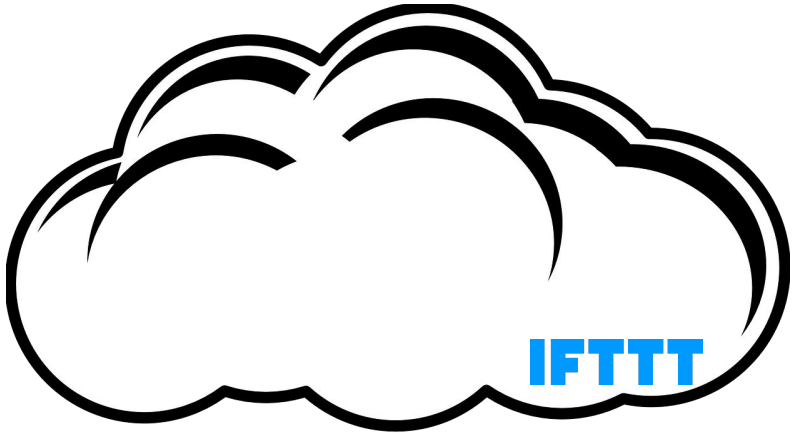
TRACKING REMOTE TIMING ATTACKS

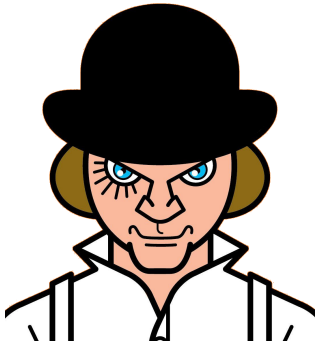
WASP

Inria

Iulia BASTYS Musard BALLIU Tamara REZK Andrei SABELFELD

Remote attacker

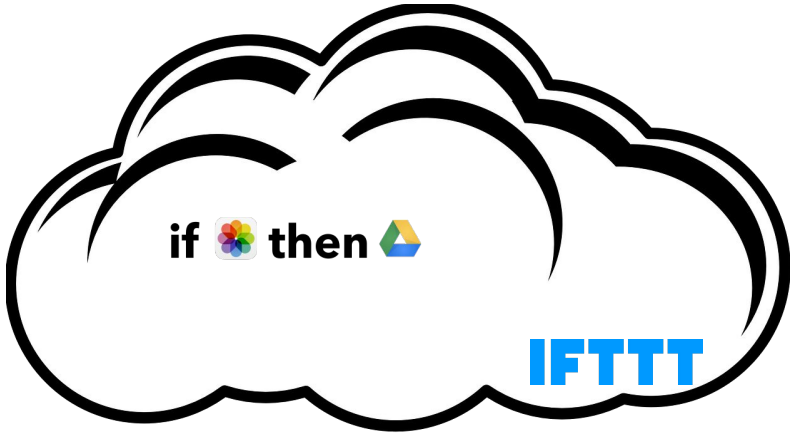


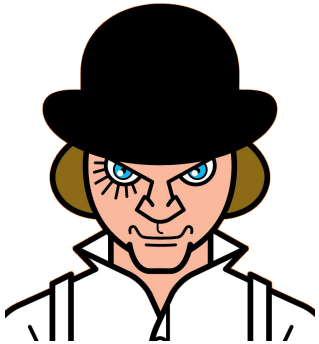


if then JavaScript

*Avatars courtesy of http://costhanzo.com/

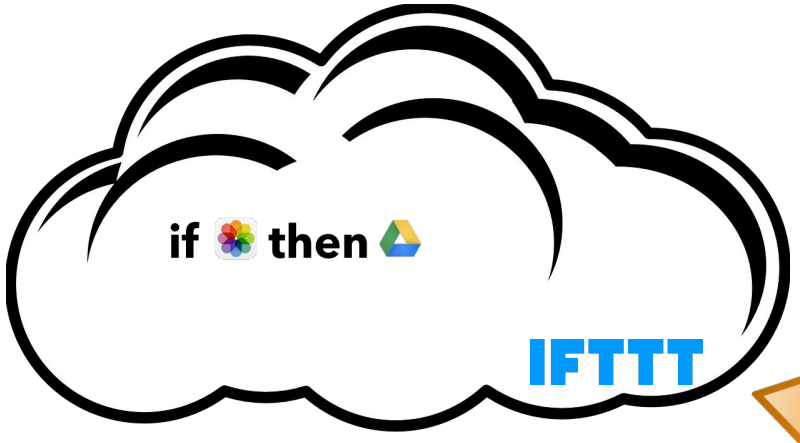
Remote attacker

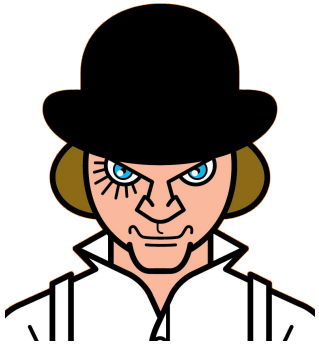


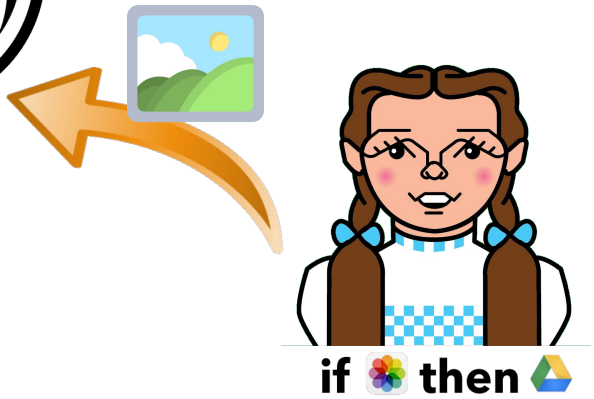


*Avatars courtesy of http://costhanzo.com/

Remote attacker

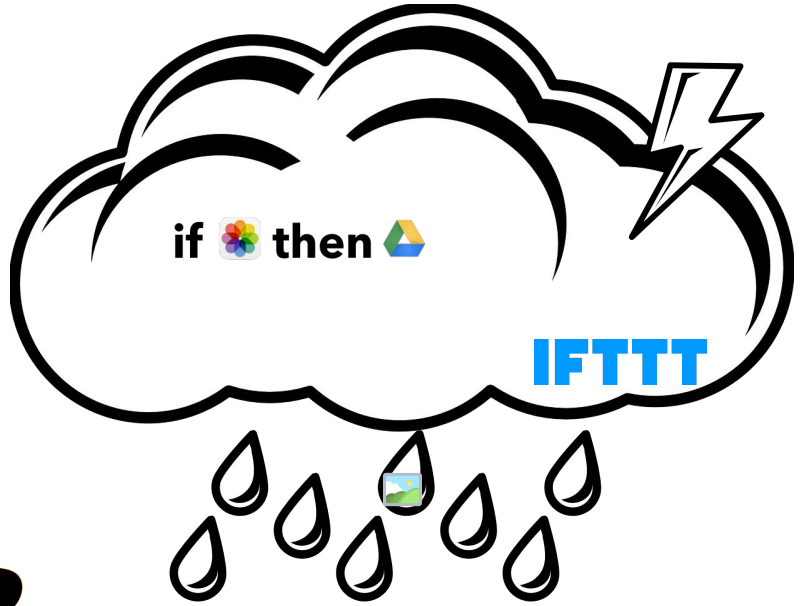


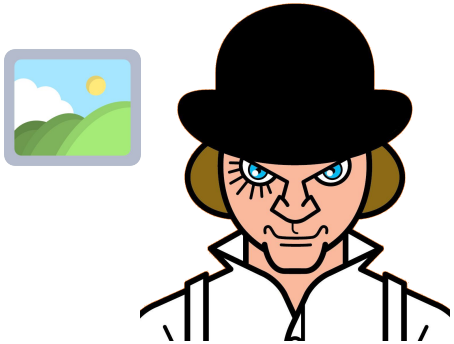


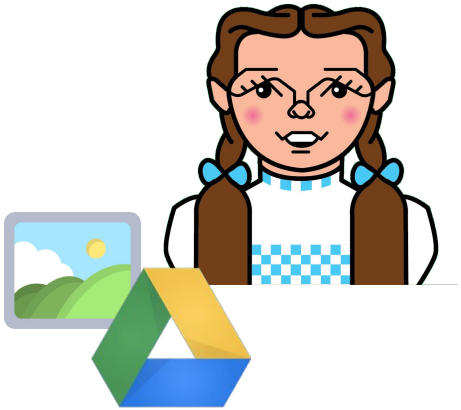


*Avatars courtesy of http://costhanzo.com/

Remote attacker







*Avatars courtesy of http://costhanzo.com/

Remote attacker vs. Local attacker

- writes/knows the program
- doesn't know when the program started
- measures time in between public outputs
- different machines

- writes/knows the program
- knows when the program started
- measures time in between instructions
- same machine

Remote attacker vs. Local attacker

weaker than

- writes/knows the program
- doesn't know when the program started
- measures time in between public outputs
- different machines

- writes/knows the program
- knows when the program started
- measures time in between instructions
- same machine

Classical exfiltration

explicit flow

```
out_L(h)
```

Attacker knowledge: `h`

implicit flow

```
if h then l = 1
else l = 0
out_L(l)
```

`h = true` if `l = 1`

Classical exfiltration …

explicit flow

$out_L(h)$

Attacker knowledge: h

implicit flow

```
if h then l = 1
else l = 0
out_L(l)
```

h = true if l = 1

… addressed in previous work

Exfiltration via remote timing

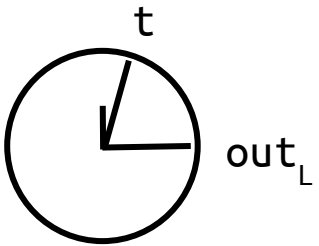
time, branch, I/O

t = 

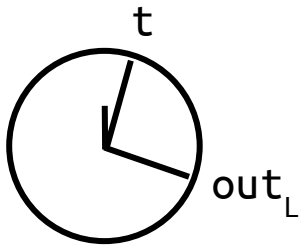
```
if h then h1 = h2
```

$out_L(t)$

h = false



h = true



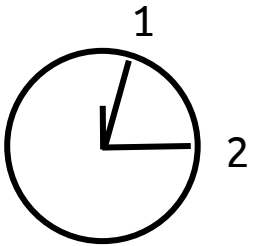
Attacker knowledge: h = true if 

Exfiltration via remote timing

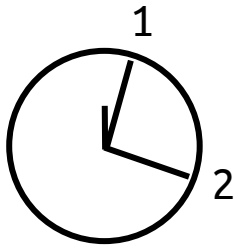
I/O, branch, I/O

```
out_L(1)
if h then h1 = h2
out_L(2)
```

h = false



h = true



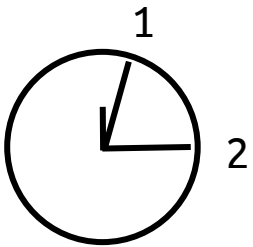
Attacker knowledge: `h = true` if 

Exfiltration via remote timing

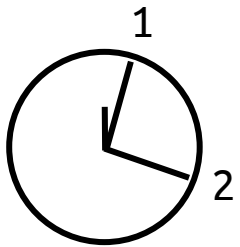
cache

```
if h then h1 = h2
out_L(1)
h1 = h2
out_L(2)
```

h = true



h = false



Attacker knowledge: `h = false` if 

Exfiltration via remote timing

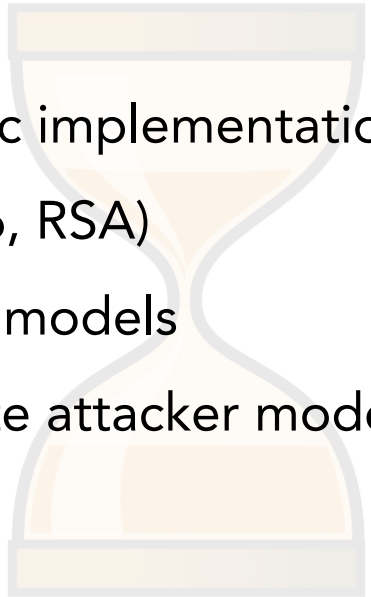
high delay

```
t = 🕐

if h % 2 = seconds(t) % 2 then h = h

else h = h; ...; h = h

out_L(1)
```

Attacker knowledge: `h % 2 = seconds(🕐) % 2`

Constant time security

- popular in cryptographic implementations (e.g. AES, DES, SHA256, RSA)
- useful for local attacker models
- too restrictive for remote attacker models
  - no high branching



Constant-time insecure programs

branch, I/O

```
if h then h1 = h2
out_L(1)
```

I/O, I/O, branch

```
out_L(1)
out_L(2)
if h then h1 = h2
```

Remote secure* programs

branch, I/O

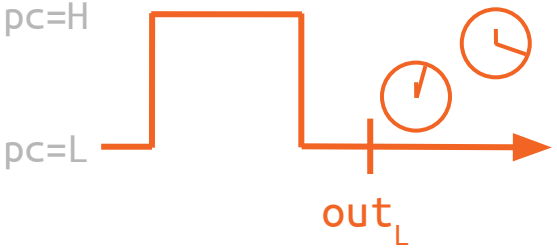
```
if h then h1 = h2
out_L(1)
```

I/O, I/O, branch

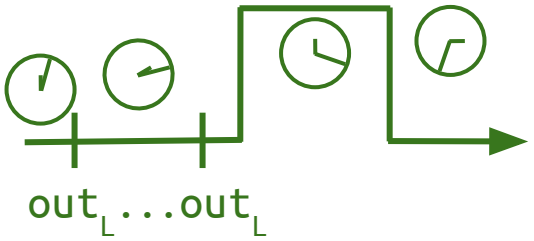
```
out_L(1)
out_L(2)
if h then h1 = h2
```

Attacker knowledge: $h \in \{true, false\}$

*Formal knowledge-based definition in paper



Patterns of remote secure programs*



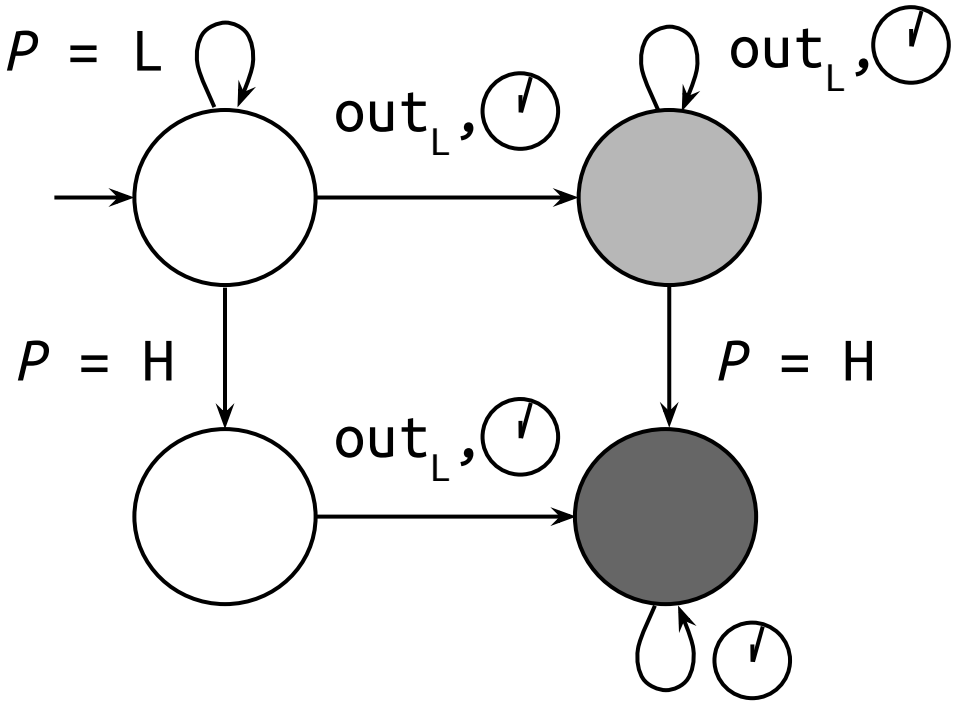
one low output after branching on high if no previous time reads OR low outputs

any low outputs before branching on high; unrestricted time reads

*wrt timing and assuming explicit & implicit flows handled

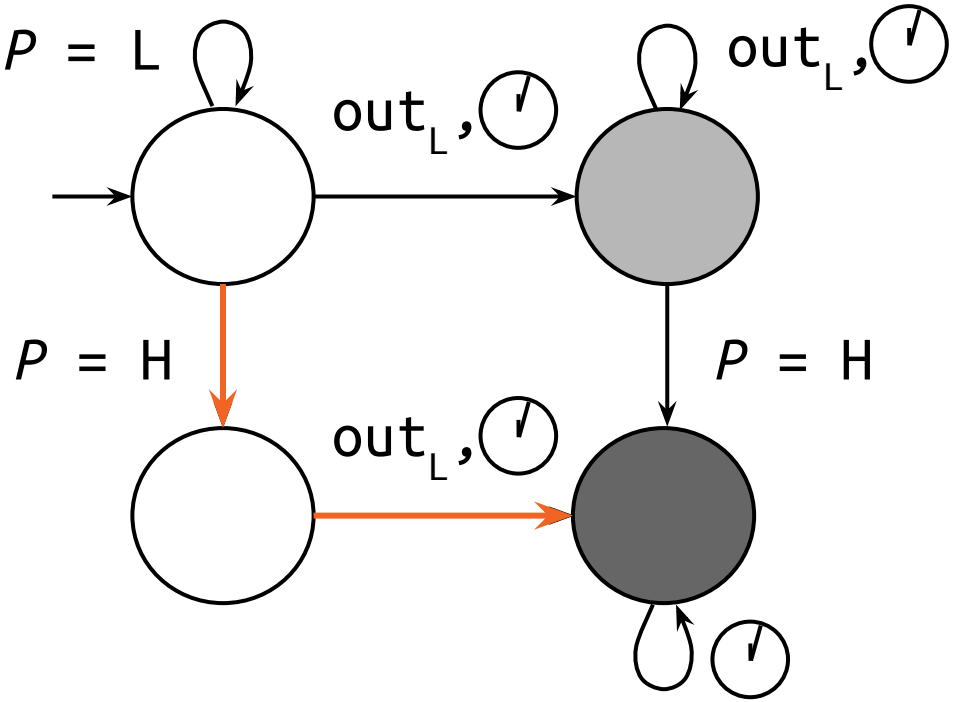
Clockwork: Dynamic monitor for RS

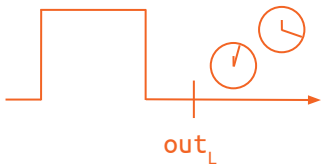
P = highest PC ever on stack



Clockwork: Dynamic monitor for RS

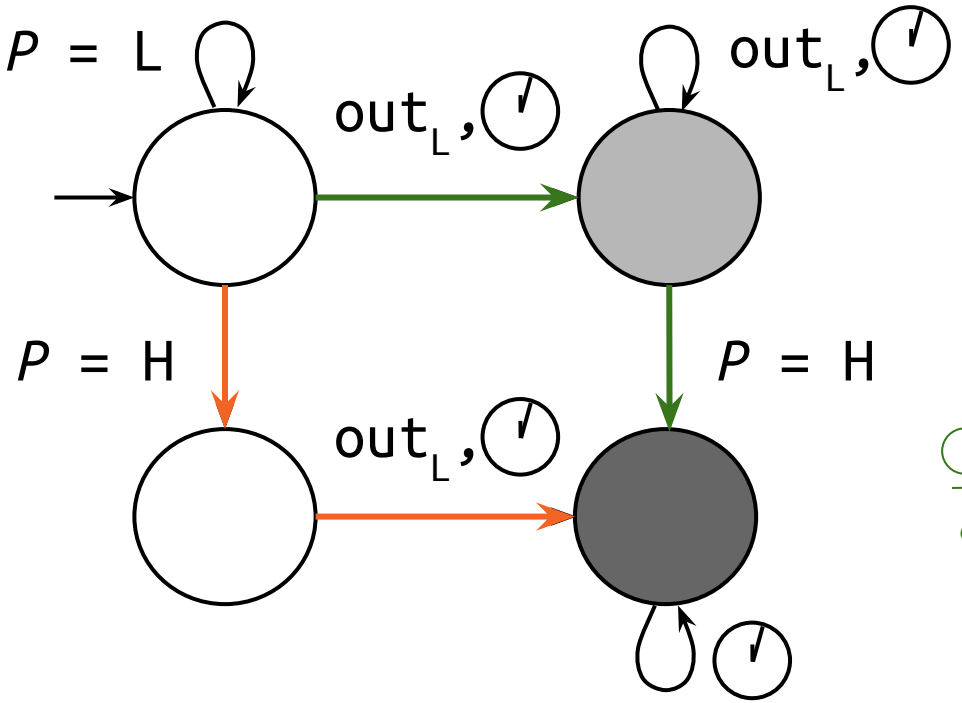
P = highest PC ever on stack

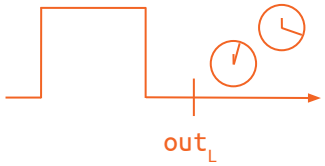


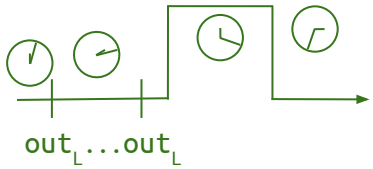


Clockwork: Dynamic monitor for RS

P = highest PC ever on stack







Case studies

- basic code
- exfiltrate GPS location
- cloud-based
- suitable for securing IoT apps

- real-world software
- no remote timing leaks
- client side
- suitable for security testing

IFTTT

Open Verificatum

Conclusion

- Timing attacks under remote execution
- Knowledge-based remote security
- Clockwork - Permissive yet sound dynamic monitor
- JSFlow-based implementation
  - Case studies with IFTTT and Verificatum
- Generalization to arbitrary lattices

Full paper & materials