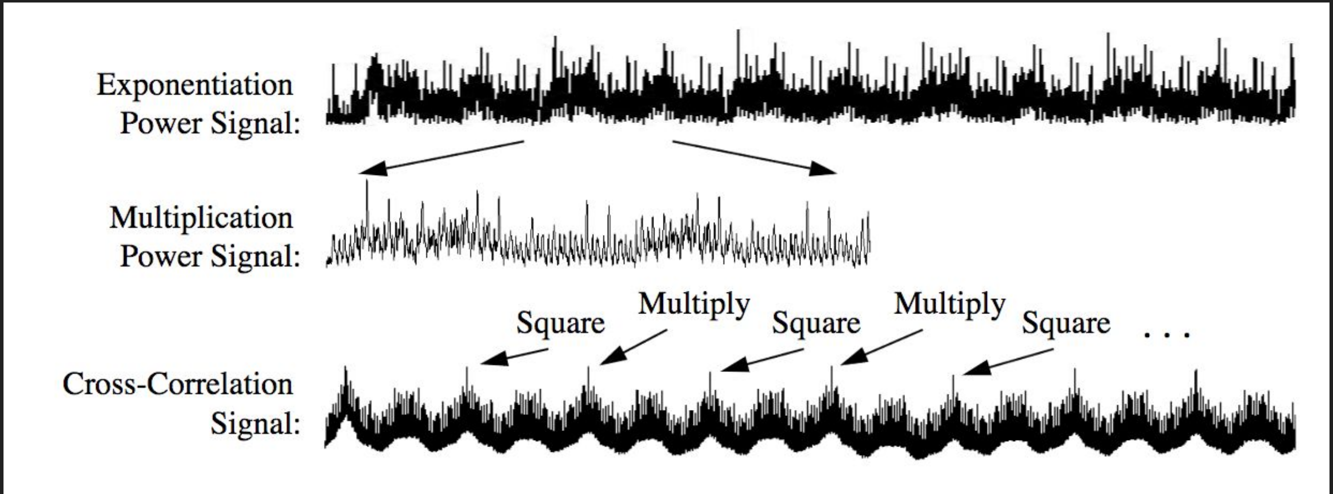
Side-Channel Attacks on Everyday Applications

Taylor Hornby[†‡]

(With thanks to Prof. John Aycock[†])

University of Calgary[†]

Zcash[‡]



T. Messerges et al. *CHES,* 1999.

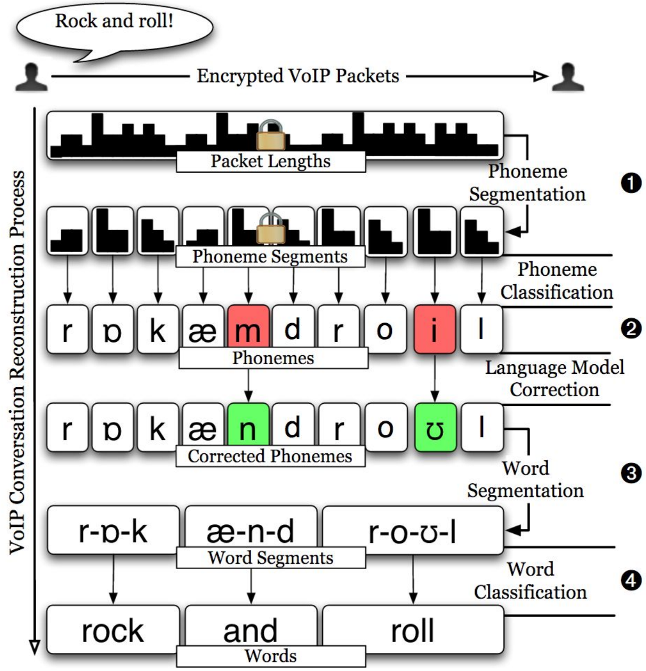
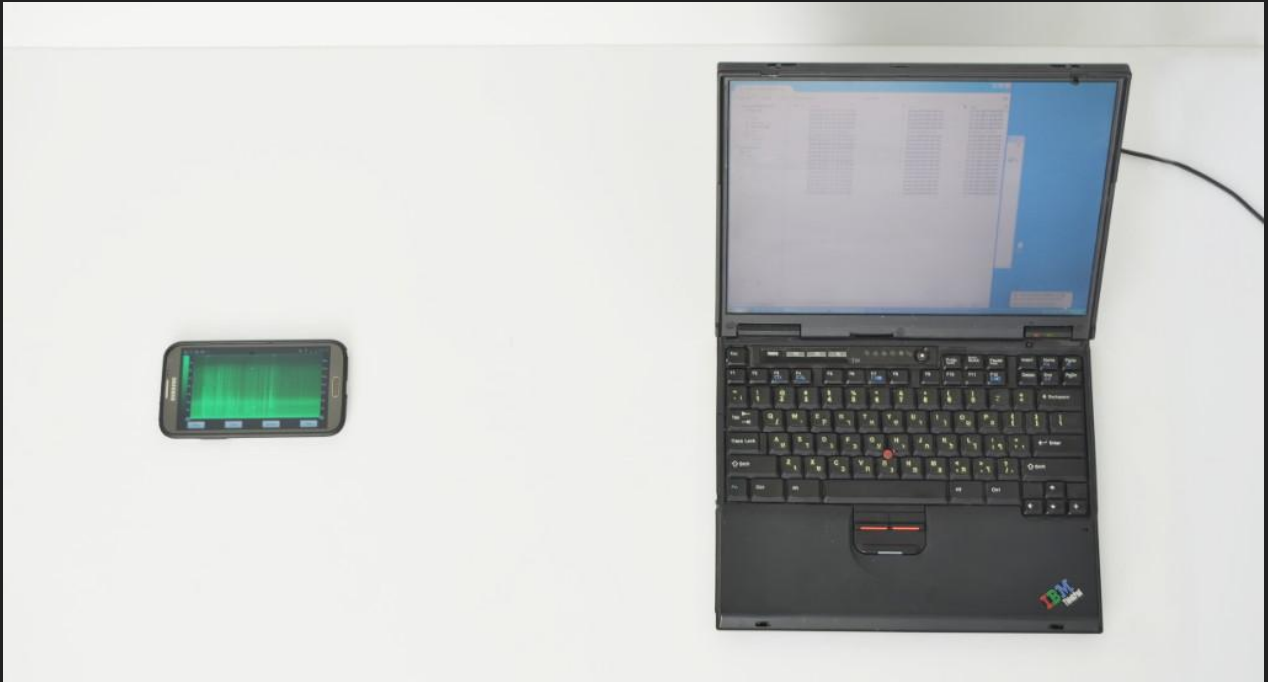


Figure 2. Overall architecture of our approach for reconstructing transcripts of VoIP conversations from sequences of encrypted packet sizes.

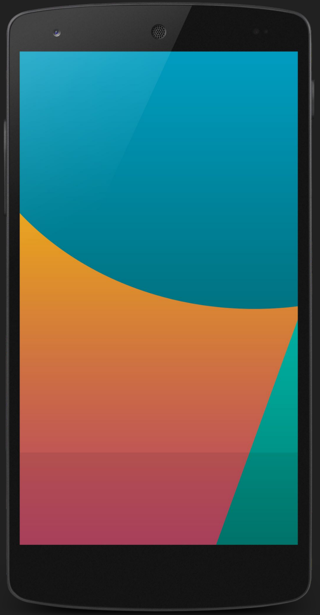
A. White et al. *IEEE S&P*, 2011.

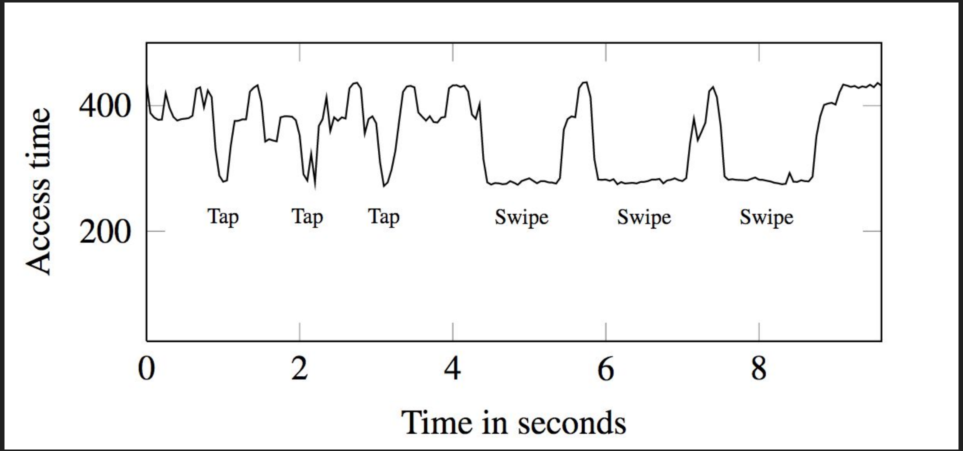


D. Genkin et al. *CRYPTO*, 2014.

Side channels affect more than crypto.

M. Backes, et al. *USENIX Security*, 2010.

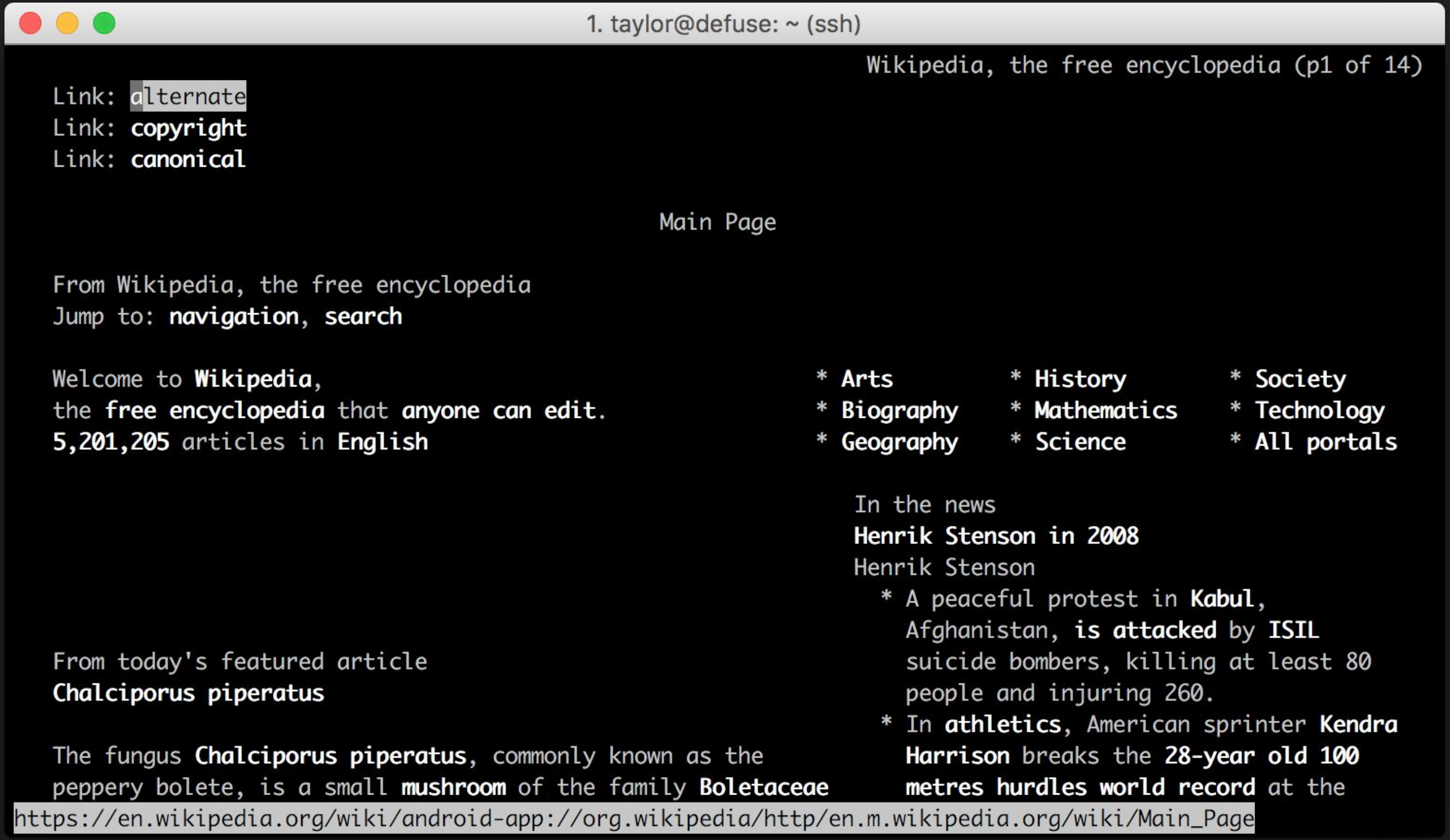




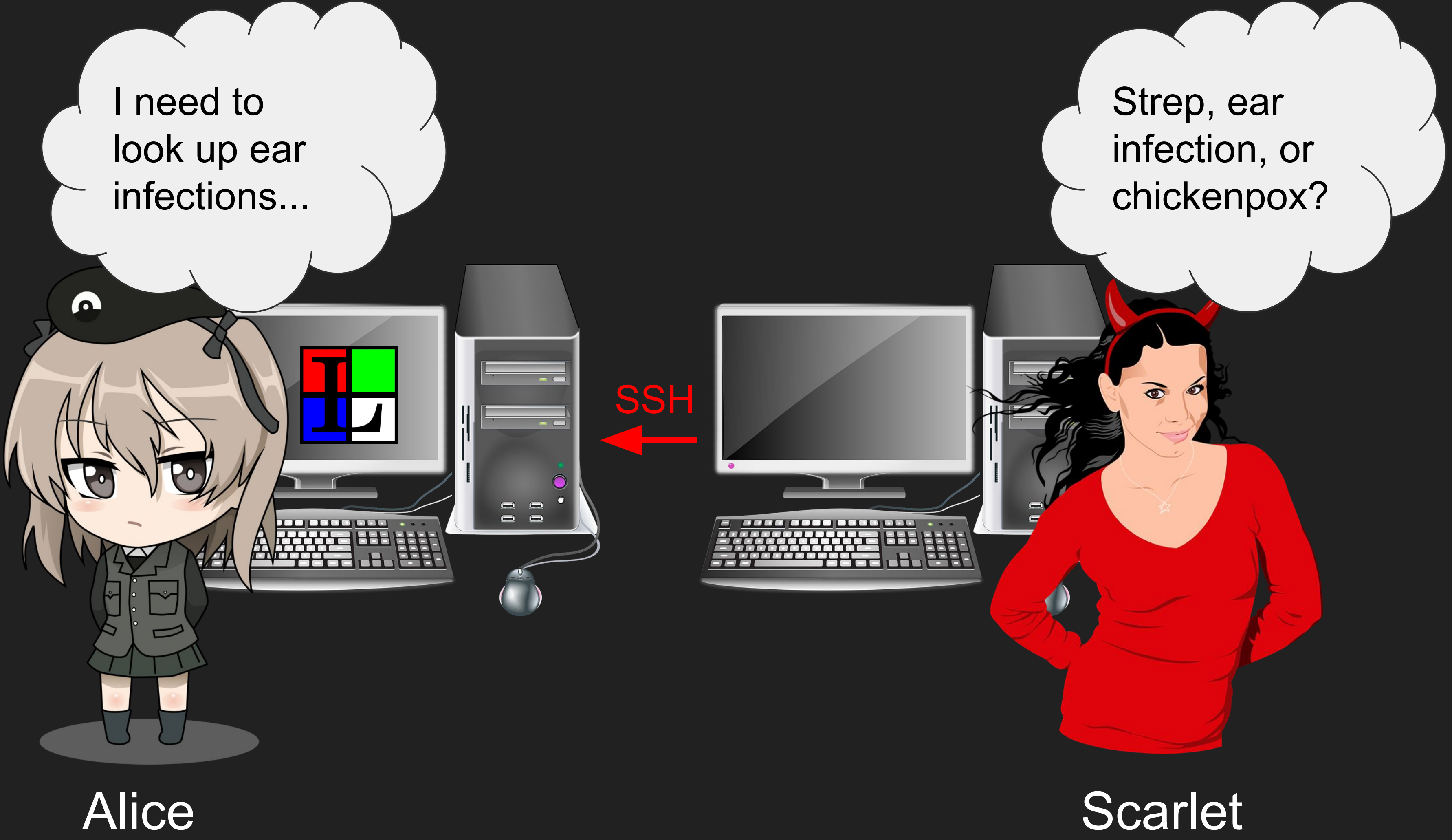
M. Lipp et al. *USENIX Security*, 2016.

A New Attack...

- Continue the “non-crypto” trend.
- Download my code and make better attacks!

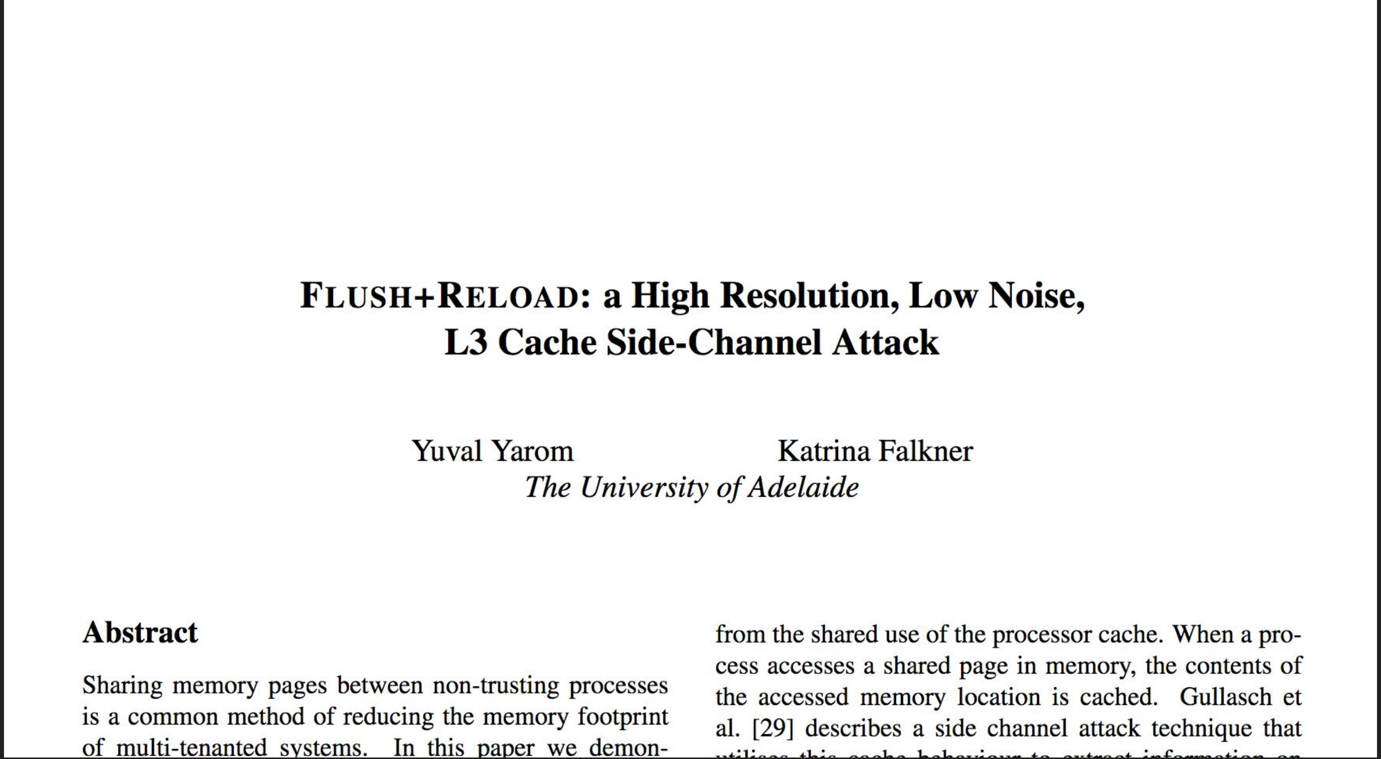


```
                                      Wikipedia, the free encyclopedia (p1 of 14)
  Link: alternate
  Link: copyright
  Link: canonical

                            Main Page

  From Wikipedia, the free encyclopedia
  Jump to: navigation, search

  Welcome to Wikipedia,                     * Arts         * History        * Society
  the free encyclopedia that anyone can edit. * Biography  * Mathematics    * Technology
  5,201,205 articles in English             * Geography    * Science        * All portals

                                              In the news
                                              Henrik Stenson in 2008
                                              Henrik Stenson
                                                * A peaceful protest in Kabul,
                                                  Afghanistan, is attacked by ISIL
  From today's featured article                   suicide bombers, killing at least 80
  Chalciporus piperatus                           people and injuring 260.
                                                * In athletics, American sprinter Kendra
  The fungus Chalciporus piperatus, commonly known as the   Harrison breaks the 28-year old 100
  peppery bolete, is a small mushroom of the family Boletaceae   metres hurdles world record at the
https://en.wikipedia.org/wiki/android-app://org.wikipedia/http/en.m.wikipedia.org/wiki/Main_Page
```


I need to look up ear infections...
Strep, ear infection, or chickenpox?
SSH
Alice
Scarlet

Interesting.
Aha! Ear infection!
SSH
EAR INFECTION
Unprivileged Spy Tool
Alice
Scarlet

Background: Flush+Reload



Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack

Yuval Yarom Katrina Falkner
The University of Adelaide

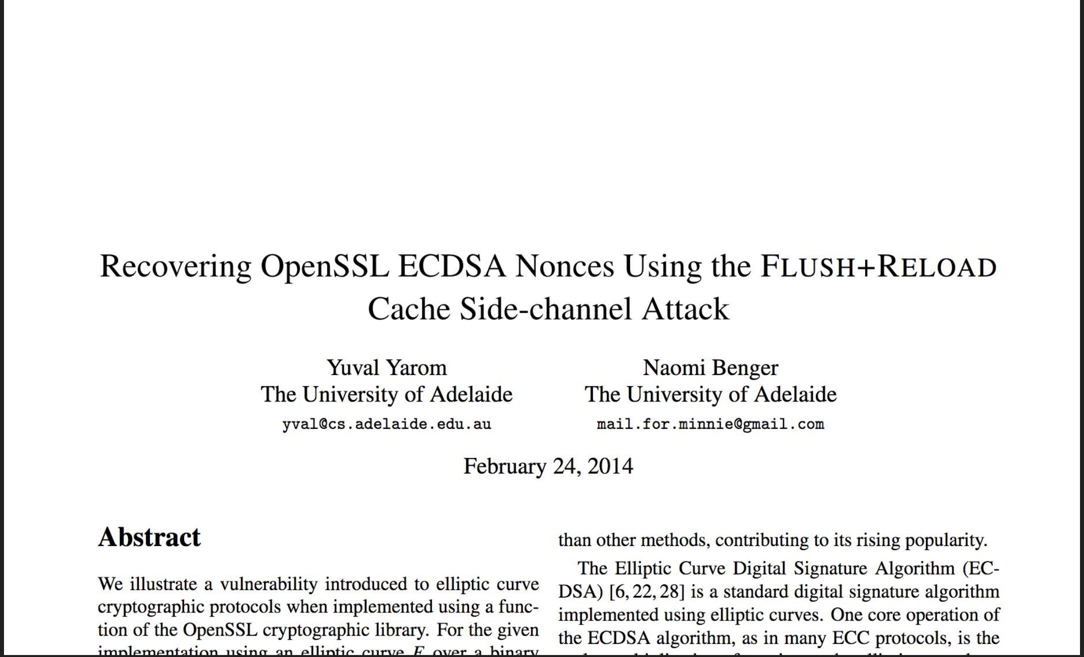
Abstract

Sharing memory pages between non-trusting processes is a common method of reducing the memory footprint of multi-tenanted systems. In this paper we demon-

from the shared use of the processor cache. When a process accesses a shared page in memory, the contents of the accessed memory location is cached. Gullasch et al. [29] describes a side channel attack technique that utilises this cache behaviour to extract information on

Y. Yaram, K. Falkner. *USENIX Security*, 2014.

Flush+Reload is *really good* for breaking crypto...



Recovering OpenSSL ECDSA Nonces Using the FLUSH+RELOAD Cache Side-channel Attack

Yuval Yarom
The University of Adelaide
yval@cs.adelaide.edu.au

Naomi Benger
The University of Adelaide
mail.for.minnie@gmail.com

February 24, 2014

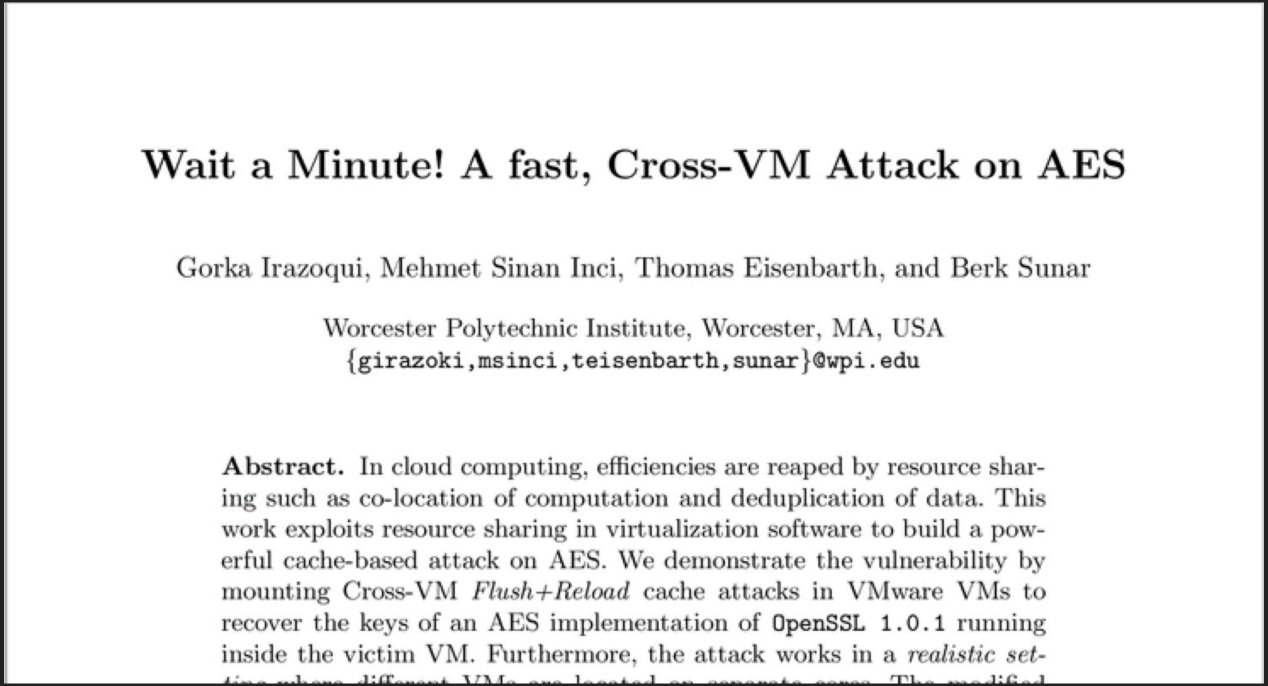
Abstract

We illustrate a vulnerability introduced to elliptic curve cryptographic protocols when implemented using a function of the OpenSSL cryptographic library. For the given implementation using an elliptic curve E over a binary

than other methods, contributing to its rising popularity.

The Elliptic Curve Digital Signature Algorithm (ECDSA) [6, 22, 28] is a standard digital signature algorithm implemented using elliptic curves. One core operation of the ECDSA algorithm, as in many ECC protocols, is the

Y. Yarom, N. Benger. *IACR*, 2014.



Wait a Minute! A fast, Cross-VM Attack on AES

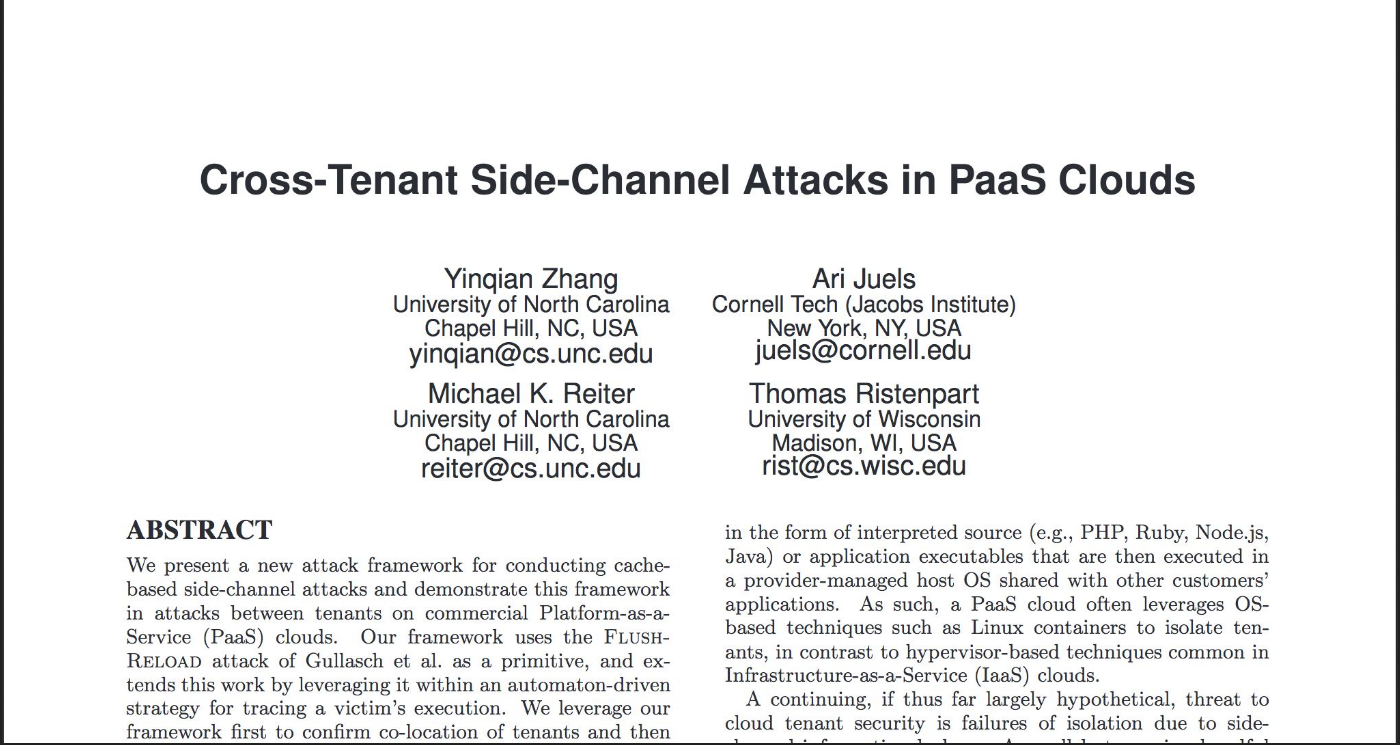
Gorka Irazoqui, Mehmet Sinan Inci, Thomas Eisenbarth, and Berk Sunar

Worcester Polytechnic Institute, Worcester, MA, USA
{girazoki,msinci,teisenbarth,sunar}@wpi.edu

Abstract. In cloud computing, efficiencies are reaped by resource sharing such as co-location of computation and deduplication of data. This work exploits resource sharing in virtualization software to build a powerful cache-based attack on AES. We demonstrate the vulnerability by mounting Cross-VM *Flush+Reload* cache attacks in VMware VMs to recover the keys of an AES implementation of `OpenSSL 1.0.1` running inside the victim VM. Furthermore, the attack works in a *realistic setting* where different VMs are located on separate cores. The modified

G. Irazoqui et al. *RAID*, 2014.

But Flush+Reload can do more...



Cross-Tenant Side-Channel Attacks in PaaS Clouds

Yinqian Zhang
University of North Carolina
Chapel Hill, NC, USA
yinqian@cs.unc.edu

Ari Juels
Cornell Tech (Jacobs Institute)
New York, NY, USA
juels@cornell.edu

Michael K. Reiter
University of North Carolina
Chapel Hill, NC, USA
reiter@cs.unc.edu

Thomas Ristenpart
University of Wisconsin
Madison, WI, USA
rist@cs.wisc.edu

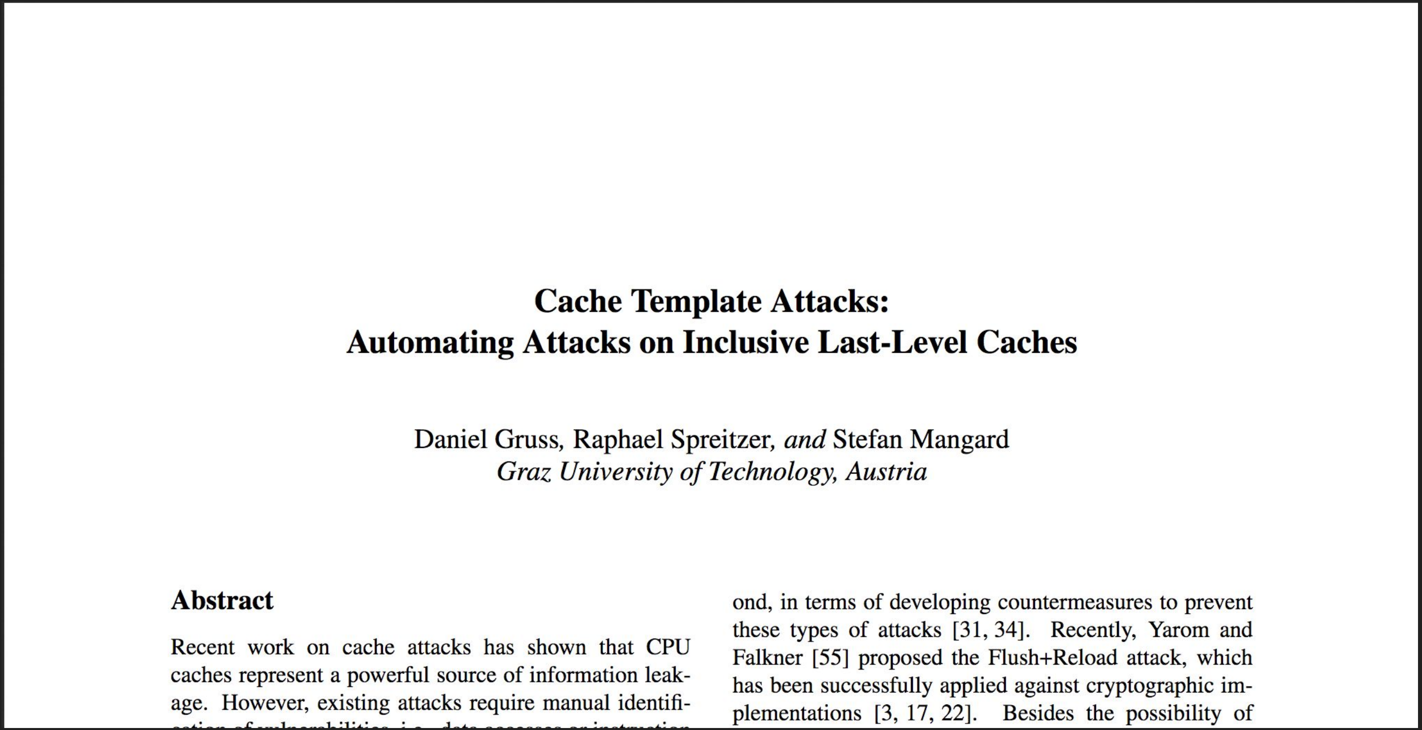
ABSTRACT

We present a new attack framework for conducting cache-based side-channel attacks and demonstrate this framework in attacks between tenants on commercial Platform-as-a-Service (PaaS) clouds. Our framework uses the FLUSH-RELOAD attack of Gullasch et al. as a primitive, and extends this work by leveraging it within an automaton-driven strategy for tracing a victim's execution. We leverage our framework first to confirm co-location of tenants and then

in the form of interpreted source (e.g., PHP, Ruby, Node.js, Java) or application executables that are then executed in a provider-managed host OS shared with other customers' applications. As such, a PaaS cloud often leverages OS-based techniques such as Linux containers to isolate tenants, in contrast to hypervisor-based techniques common in Infrastructure-as-a-Service (IaaS) clouds.

A continuing, if thus far largely hypothetical, threat to cloud tenant security is failures of isolation due to side-

Y. Zhang et al. *CCS*, 2014.



Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches

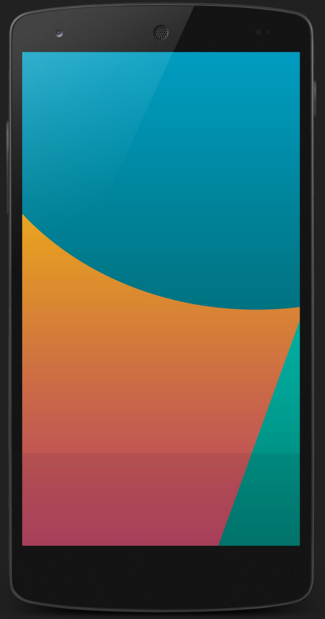
Daniel Gruss, Raphael Spreitzer, *and* Stefan Mangard
Graz University of Technology, Austria

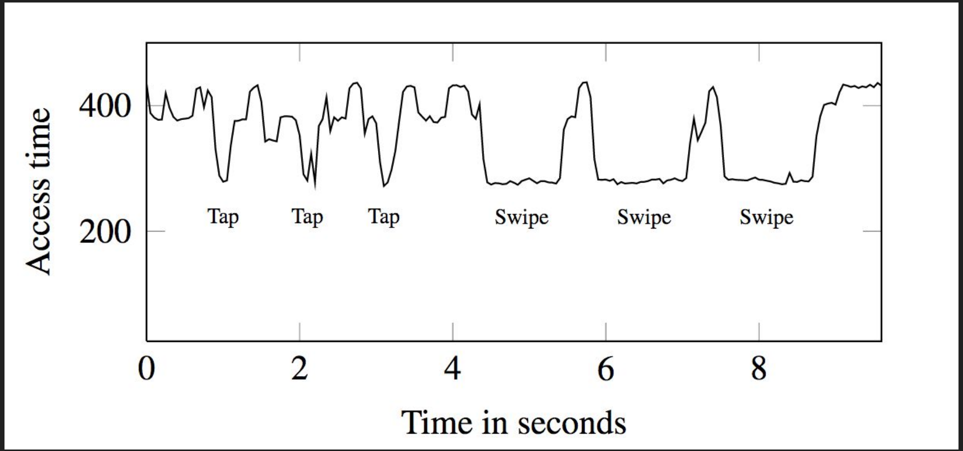
Abstract

Recent work on cache attacks has shown that CPU caches represent a powerful source of information leakage. However, existing attacks require manual identification of vulnerabilities, i.e., data accesses or instruction

ond, in terms of developing countermeasures to prevent these types of attacks [31, 34]. Recently, Yarom and Falkner [55] proposed the Flush+Reload attack, which has been successfully applied against cryptographic implementations [3, 17, 22]. Besides the possibility of

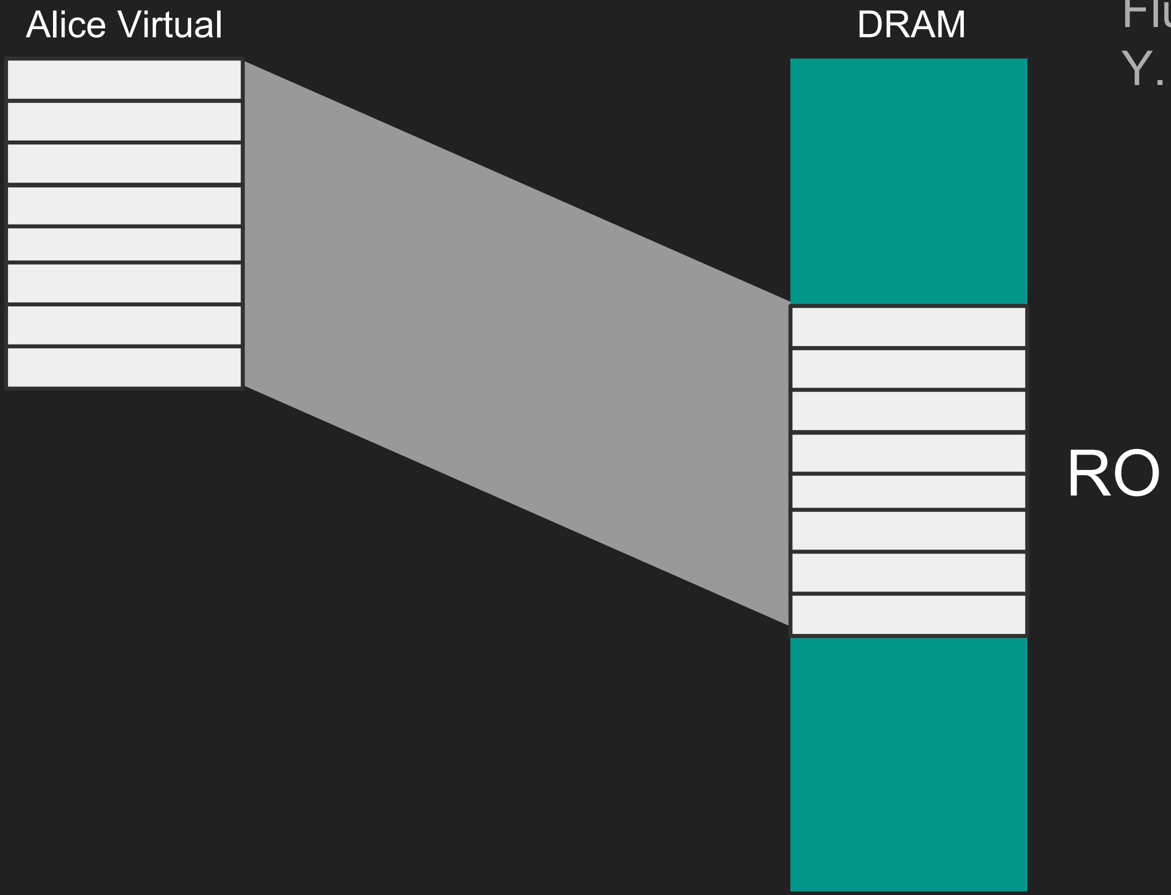
D. Gruss et al. *USENIX Security,* 2015.



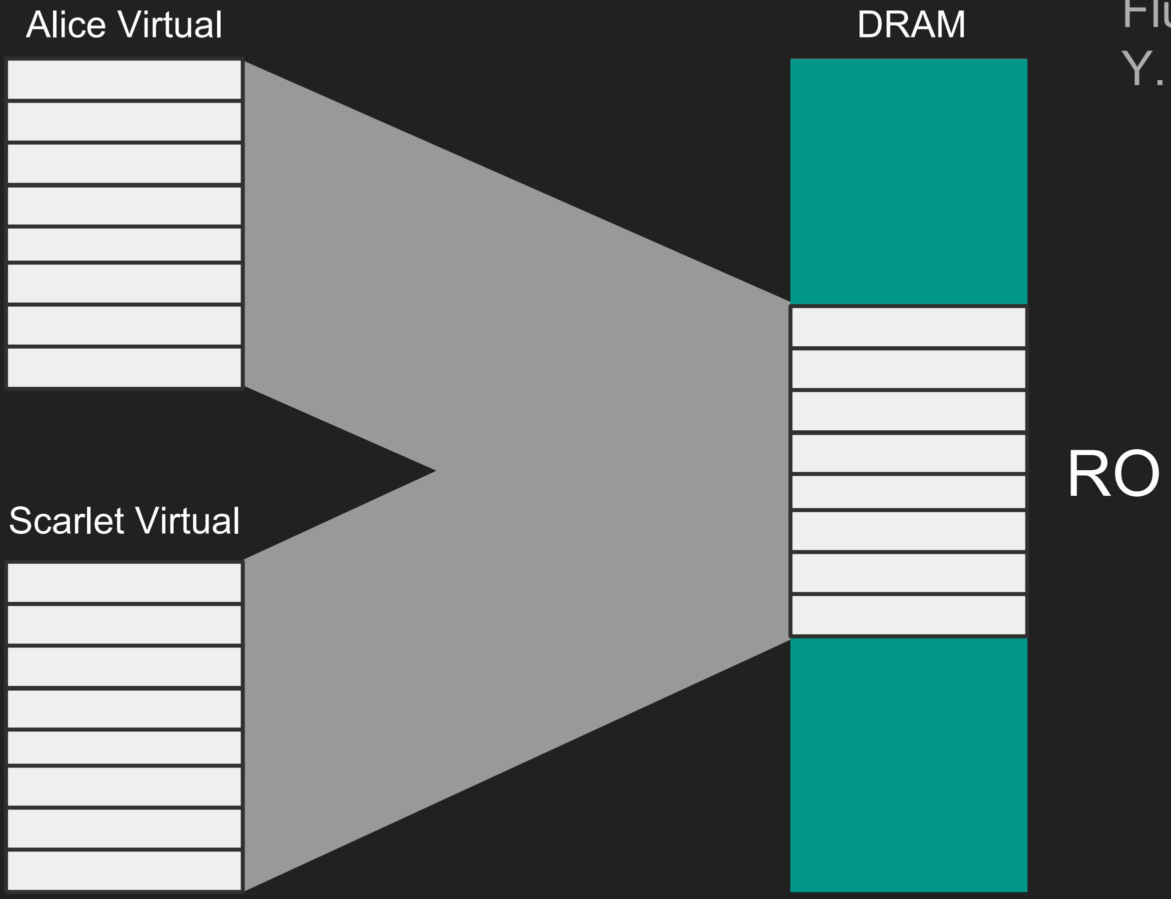


M. Lipp et al. *USENIX Security*, 2016.

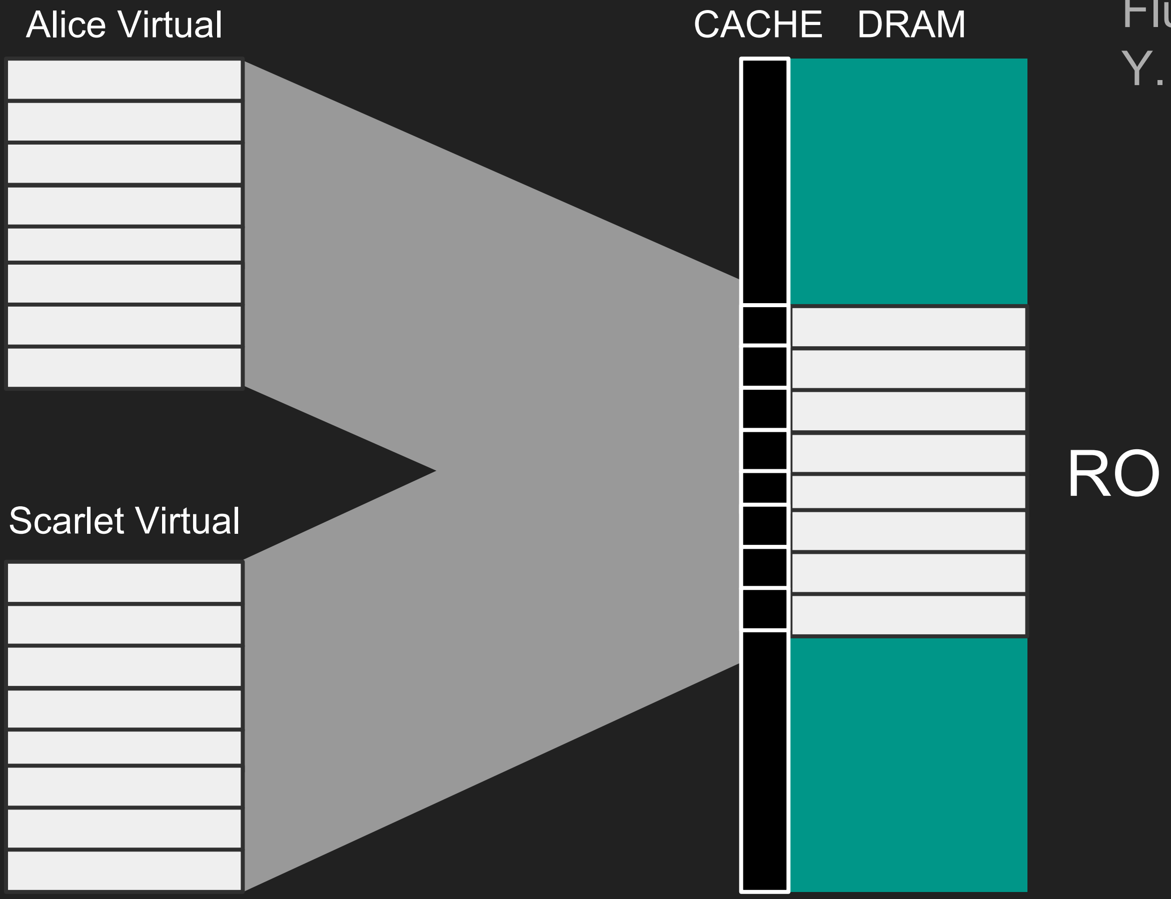
Flush+Reload by
Y. Yaram, K. Falkner.



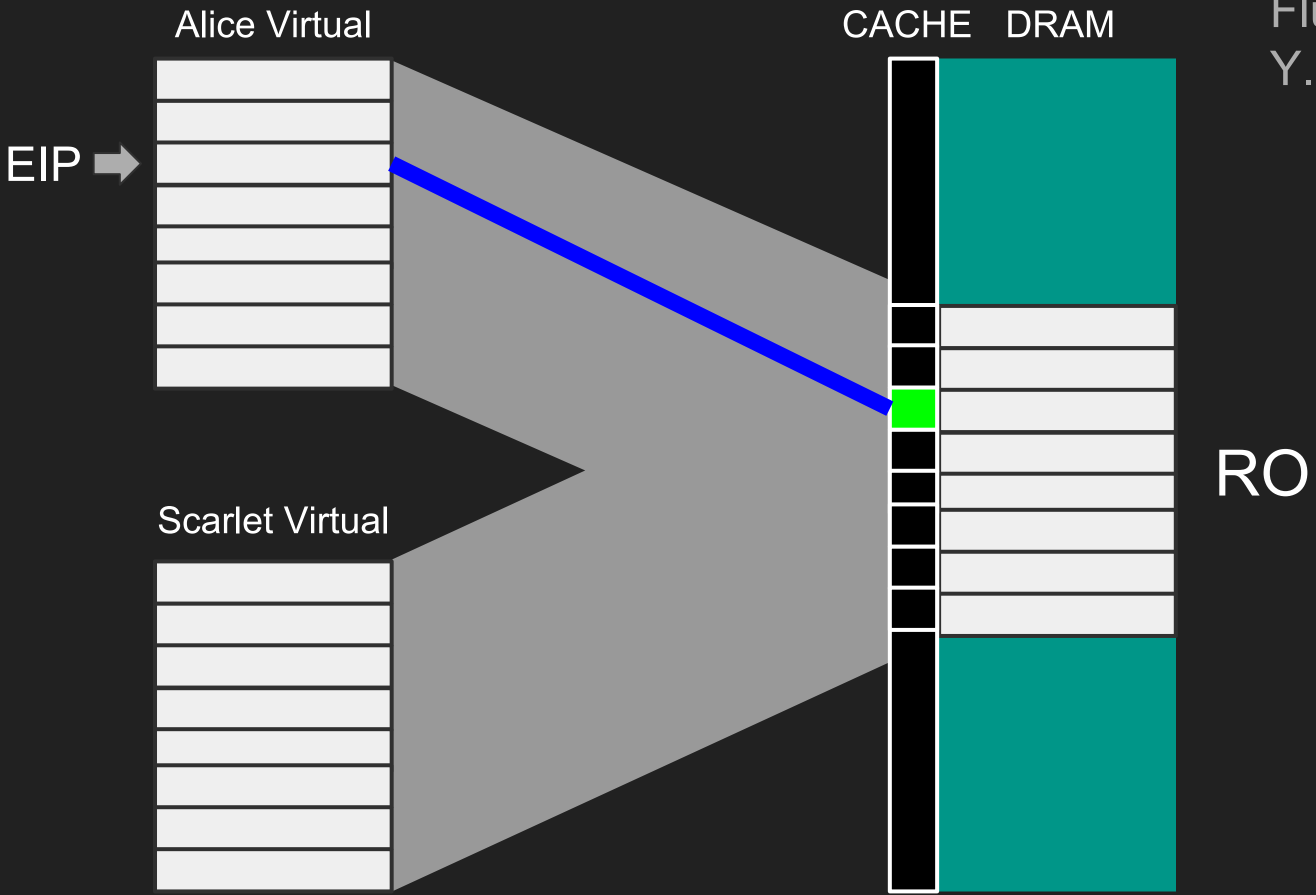
Flush+Reload by
Y. Yaram, K. Falkner.



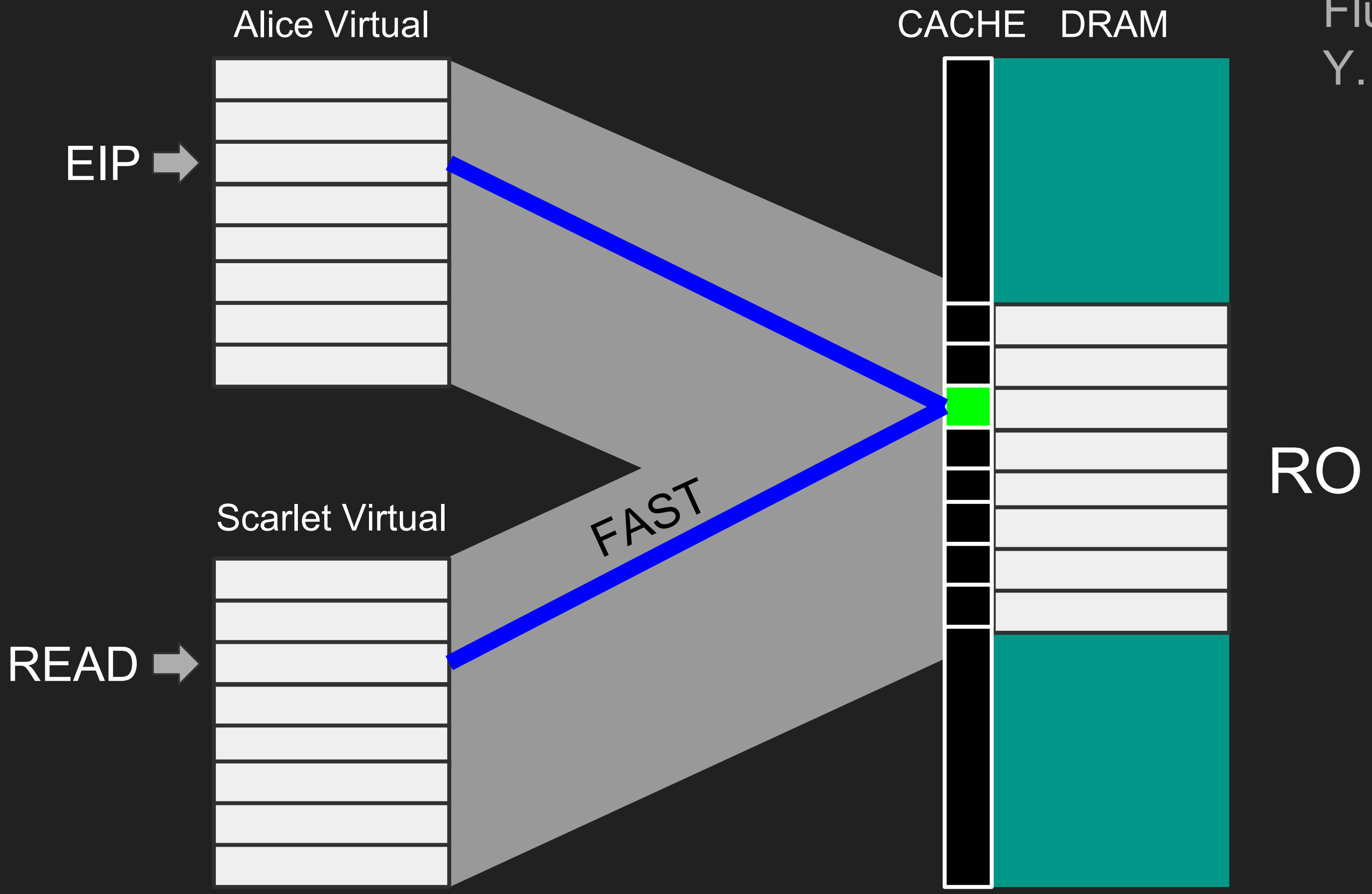
Flush+Reload by
Y. Yaram, K. Falkner.



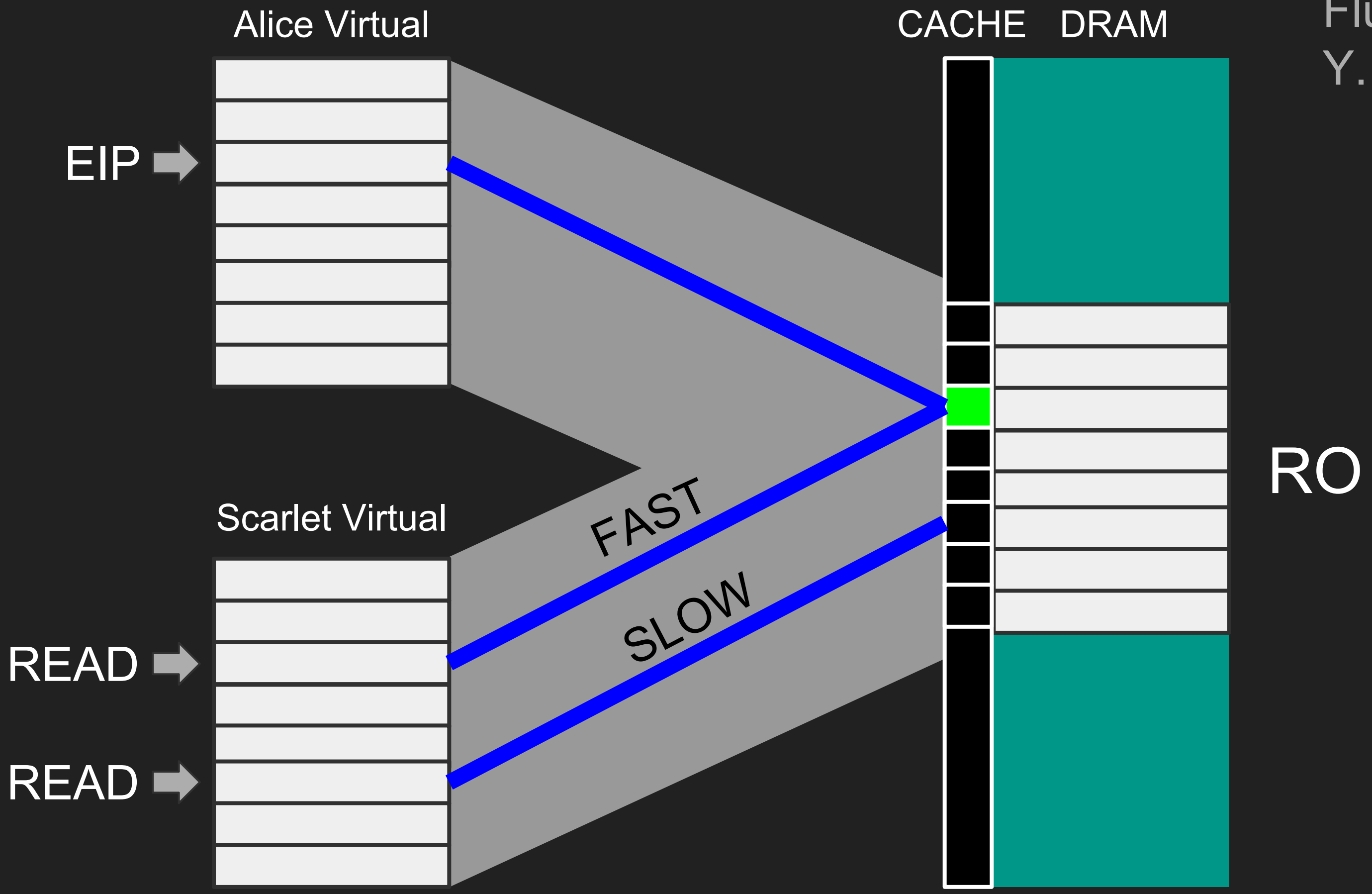
Flush+Reload by
Y. Yaram, K. Falkner.



Flush+Reload by
Y. Yaram, K. Falkner.

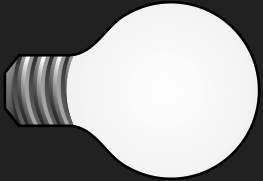


Flush+Reload by
Y. Yaram, K. Falkner.

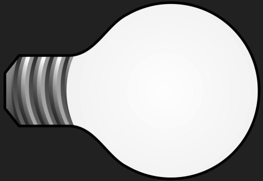


Flush+Reload by
Y. Yaram, K. Falkner.

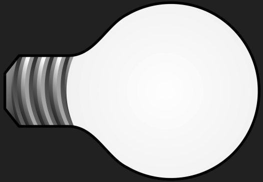
```
foo() {
   ...
}
```



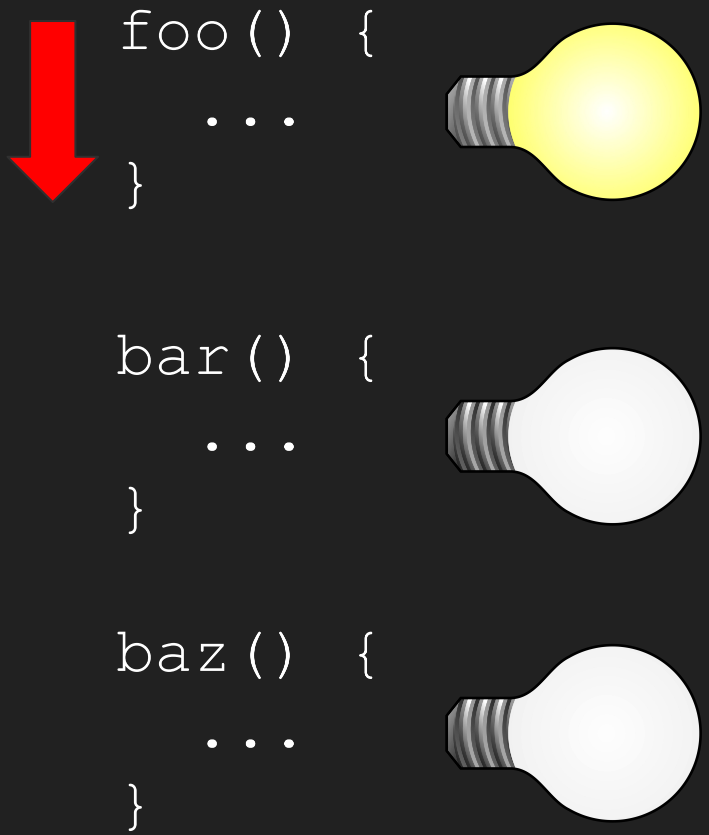
```
bar() {
   ...
}
```



```
baz() {
   ...
}
```



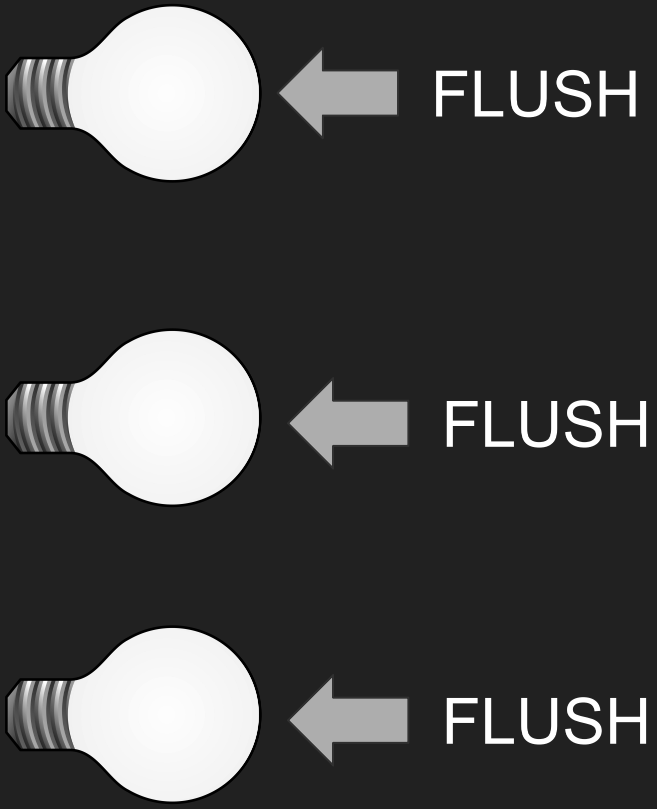
Flush+Reload by
Y. Yaram, K. Falkner.



```
foo() {
    ...
}

bar() {
    ...
}

baz() {
    ...
}
```

Flush+Reload by
Y. Yaram, K. Falkner.

```
foo() {
   ...
}

bar() {
   ...
}

baz() {
   ...
}
```

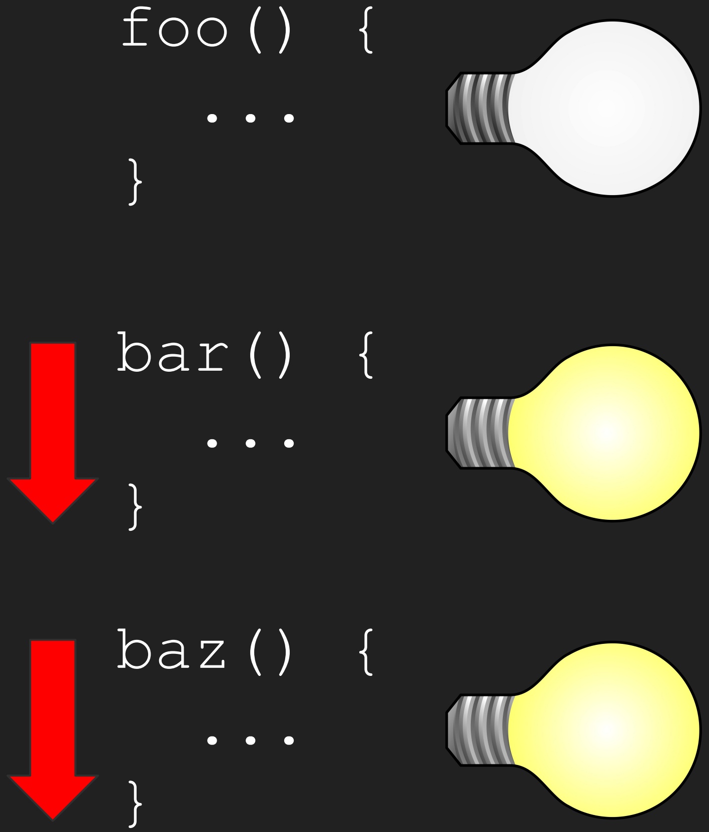


FLUSH

FLUSH

FLUSH

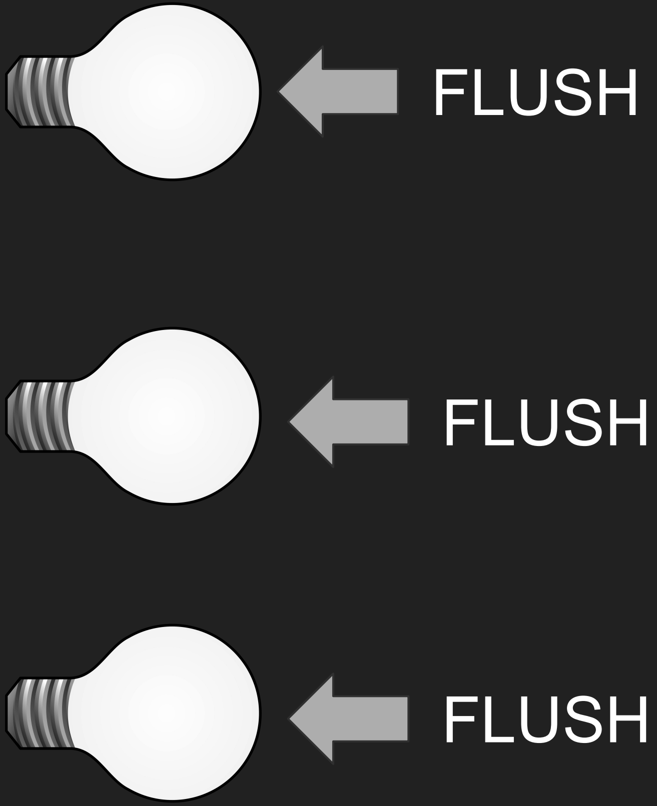
Flush+Reload by
Y. Yaram, K. Falkner.



```
foo() {
    ...
}

bar() {
    ...
}

baz() {
    ...
}
```

Flush+Reload by
Y. Yaram, K. Falkner.

```
foo() {
    ...
}


bar() {
    ...
}


baz() {
    ...
}
```

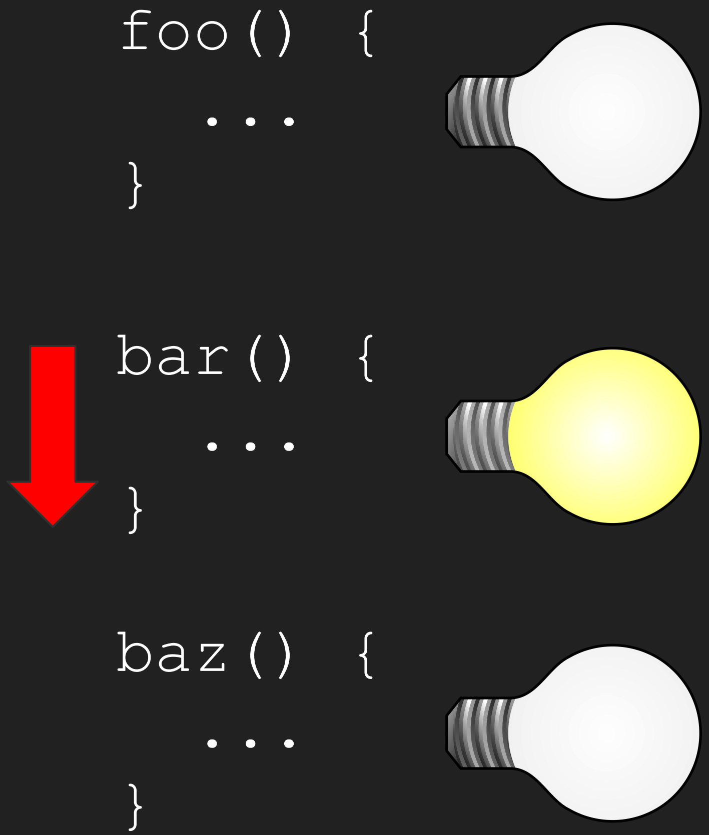


FLUSH

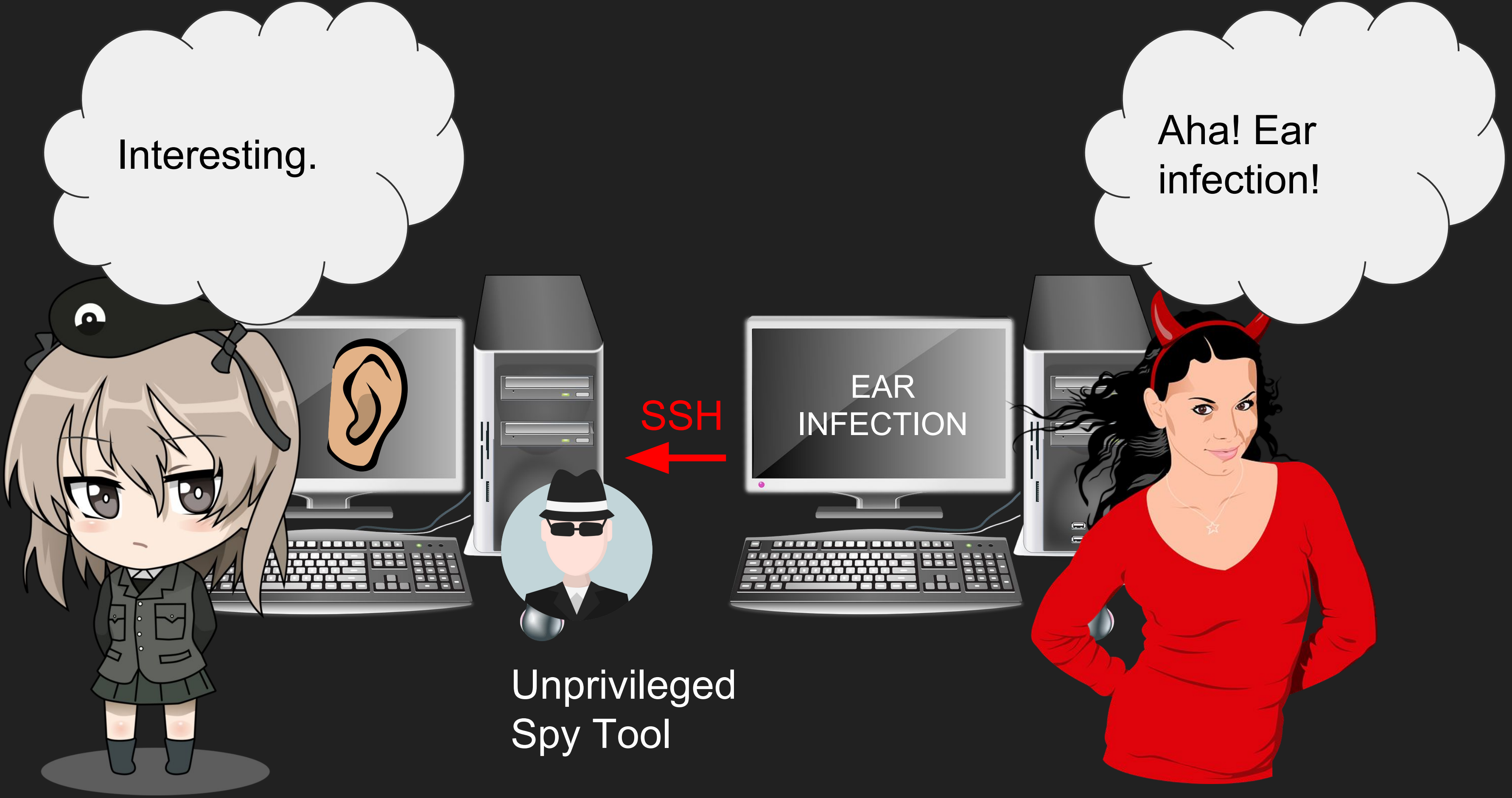
FLUSH

FLUSH

Flush+Reload by
Y. Yaram, K. Falkner.



```
foo() {
    ...
}

bar() {
    ...
}

baz() {
    ...
}
```



Unprivileged Spy Tool

Alice

Scarlet

Probe Links’s HTML-parsing code:

- parse_html()
- html_stack_dup()
- html_h()
- html_span()

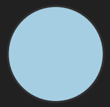
```
BDBCBCABABABACBABABCBABACBABCACABCBCACACABCABABCABCACABCBCABACACADBABDBCABDBCACB
CABDBCABDBCABDBCABCABCBCBCABCACABDCBDBCBABABDCBDBCABDACBDBCBCBABABCBCABCACBCBCBA
CBABACBACBABDBCABDBCABDBCABCBCBCBCABCABCBCABDABDCBCACBCACACBCABDABDBCABDBCABDBCB
CABCABDBCABDBCABCABDBCABDBCABCABDBCABDBCABCABDBCABDBCABCABCBDBCABDBCABABDBCACBCA
BCBCABCABDBDBCBCABCABDABDBCBCABCABCBCABCABCABDBCABDACBDBCABDBCABACBDBCABDBCABDCA
BDBCBCABCACBCABCABDBCABCABDABCBCABDBCBCBABABCBCABCABCBCBCBABACABABACBABDACBDBCAB
DBCABDBCBCABCBCABCBCABCABCBCABDBDBCBCABCABCABCACABDACBDCACBDCACBDBCBCACBCBCABDBC
ABDBCACBCABDBCABDBCBCABCABDBCABCABDCABDCABCACBCBCABDADBCBCABDBCABCABCACBCACBCACA
BDABDBCABCBCABCABDBCBCACBCBCABCABCABCABCBDBCABDBCABACBDBCABDBCABDBCABDBCABCABCAC
BCABABCBCACBABCABDBDBCBCBCBACBABACBABCABCABCBCBCBCABABACBACBABCABDABCACBDABCABDA
BCBCABACABCBCABABCABCBCABCBCABDABCBACACACABACABDBDBCABDBCABDBABCABCBCABDBACABDBA
BCABACABACABDABCBACABCABDBCBACABDBACABDBABABCABCABCABABDBCBCABDABCABCBCBABCBCBAC
ABDBCABCABCBCABDABCABCABCABCABACABCBCABD
```

Goal: Recognize this as the *Ear Infection* Wikipedia page.

Attack Stages:

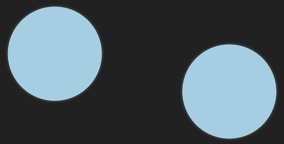
1. Training
2. Spying
3. Identification

Stage 1: Training



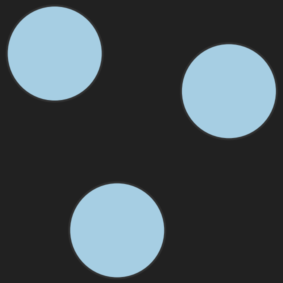
- ● Strep throat
- 🟢 Ear infection
- 🔵 Chickenpox

Stage 1: Training



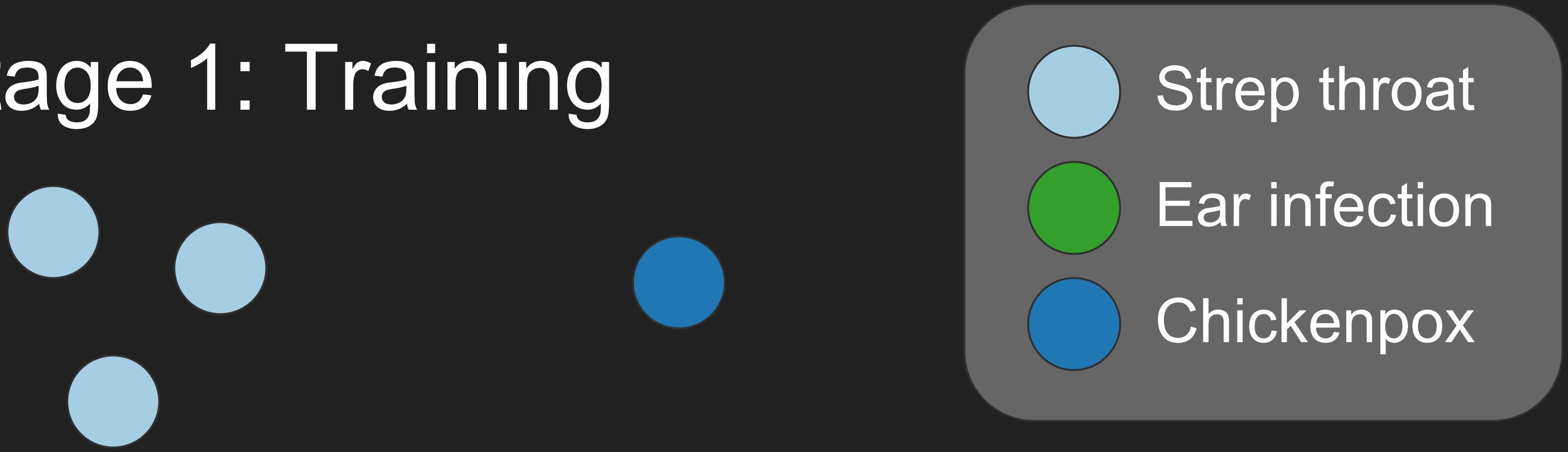
- ● Strep throat
- ● Ear infection
- ● Chickenpox

Stage 1: Training

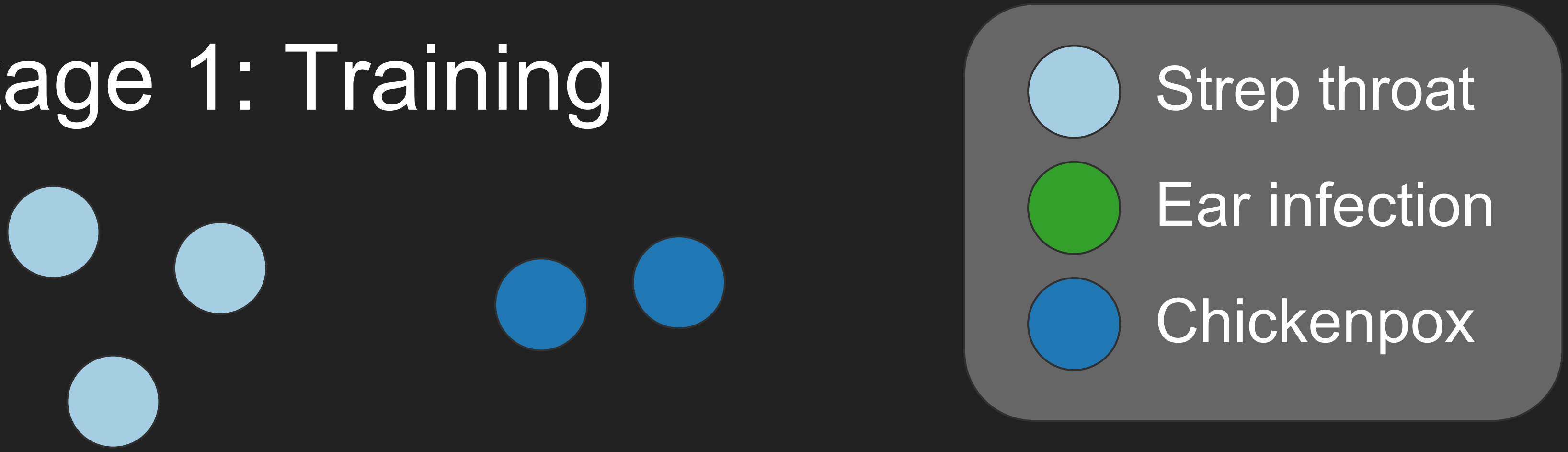


- ● Strep throat
- ● Ear infection
- ● Chickenpox

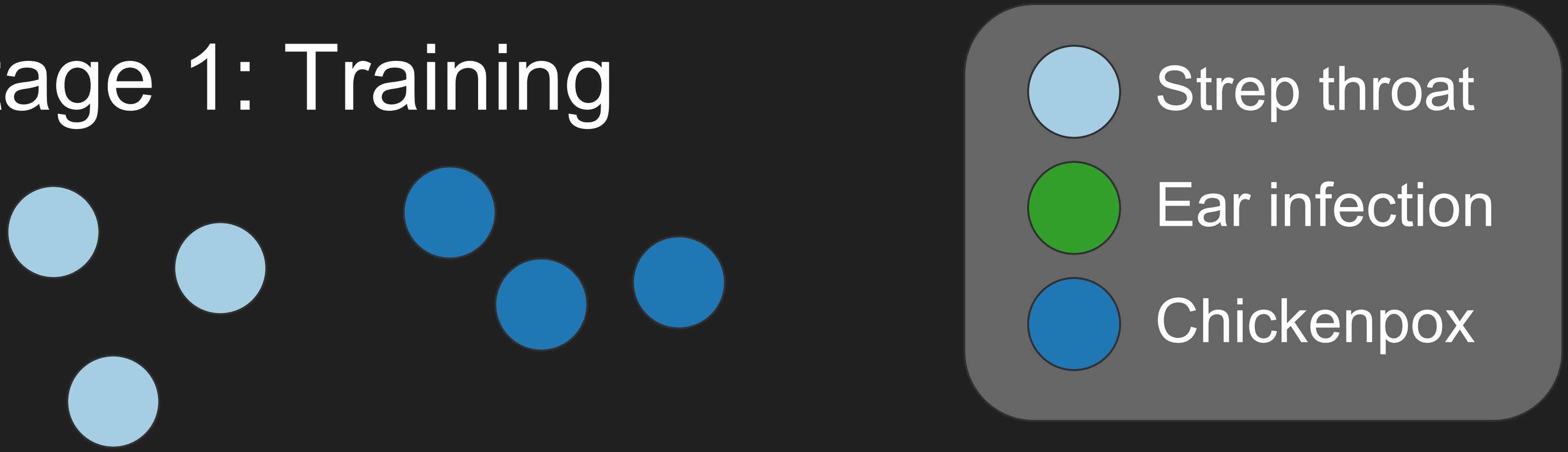
Stage 1: Training



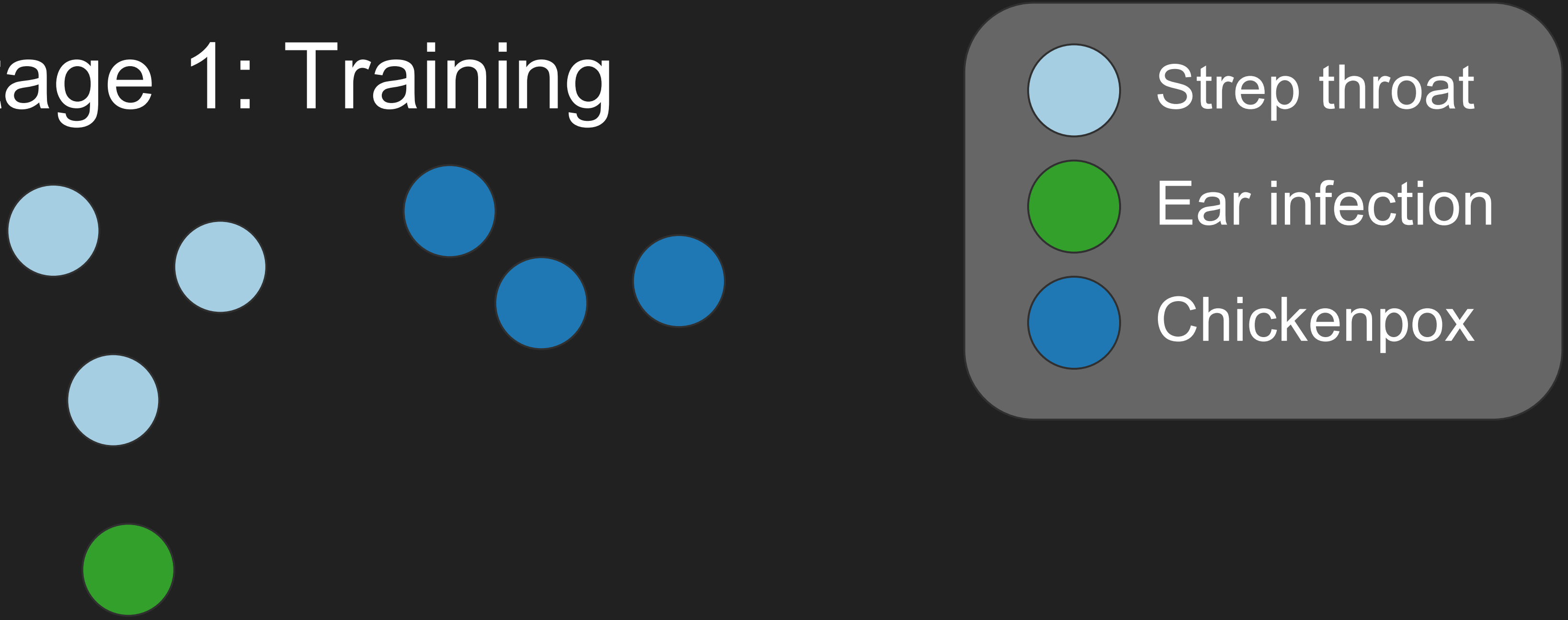
Stage 1: Training



Stage 1: Training



Stage 1: Training

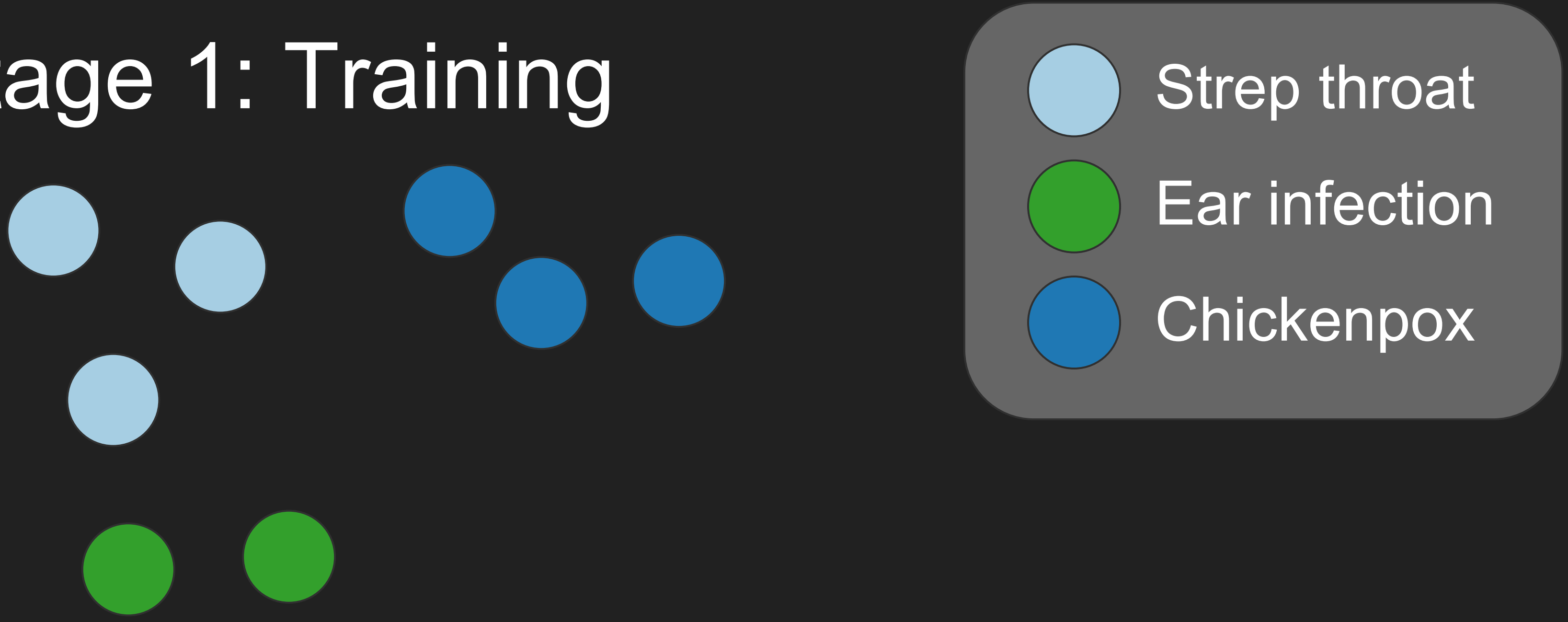


● Strep throat

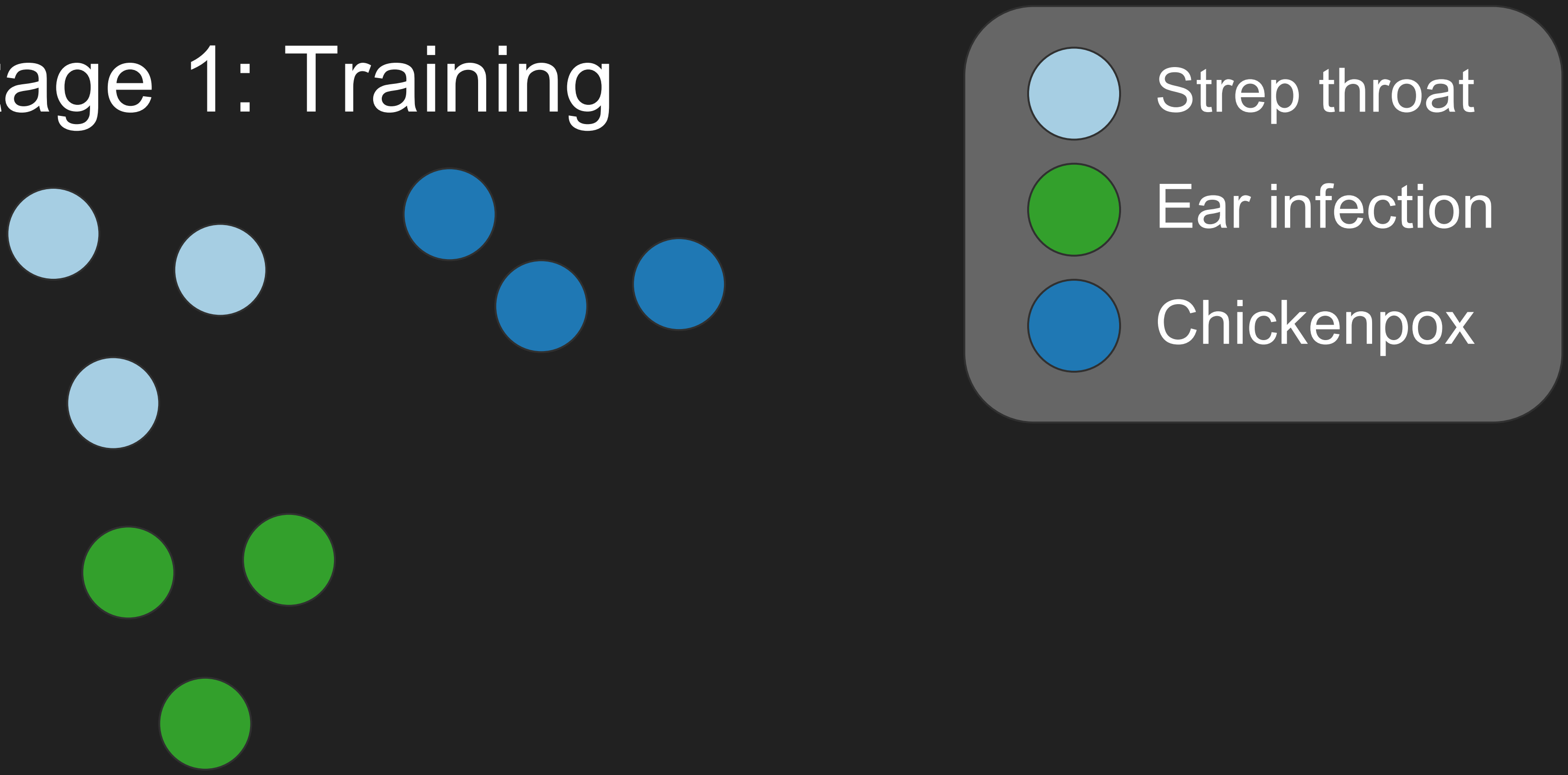
● Ear infection

● Chickenpox

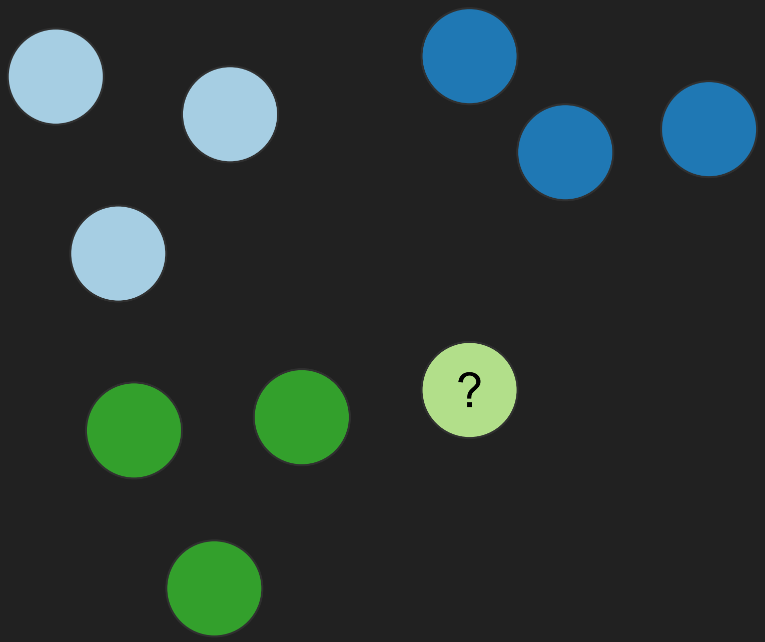
Stage 1: Training



Stage 1: Training

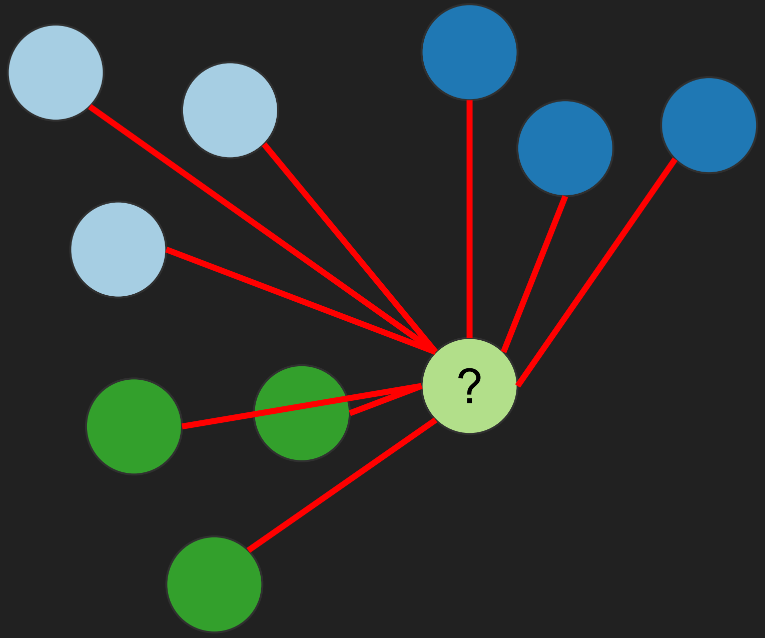


Stage 2: Spying



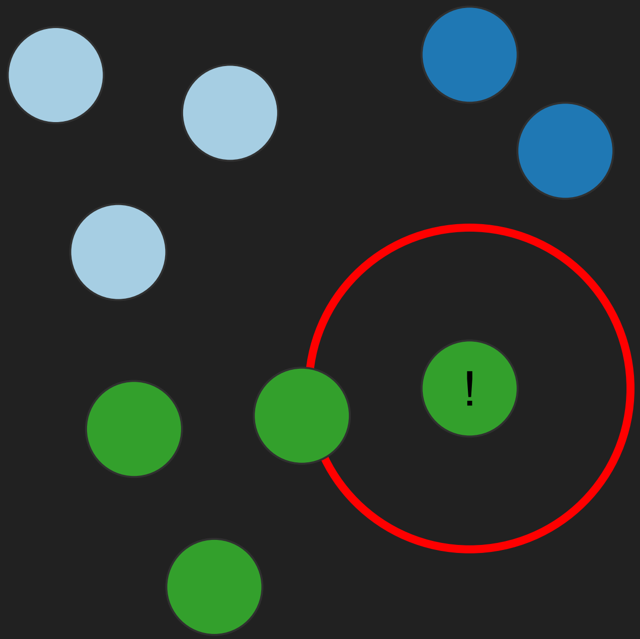
- ● Strep throat
- ● Ear infection
- ● Chickenpox

Stage 3: Identification



- Strep throat
- Ear infection
- Chickenpox

Stage 3: Identification



- ● Strep throat
- ● Ear infection
- ● Chickenpox

Output: “Ear infection”

>90% Success

(100 pages)

It’s demo time.

https://defuse.ca/BH2016

Q&A

https://defuse.ca/BH2016