Malware Defense:
New Trends and Approaches

Dawn Song

UC Berkeley

Viruses

Worms

Botnets

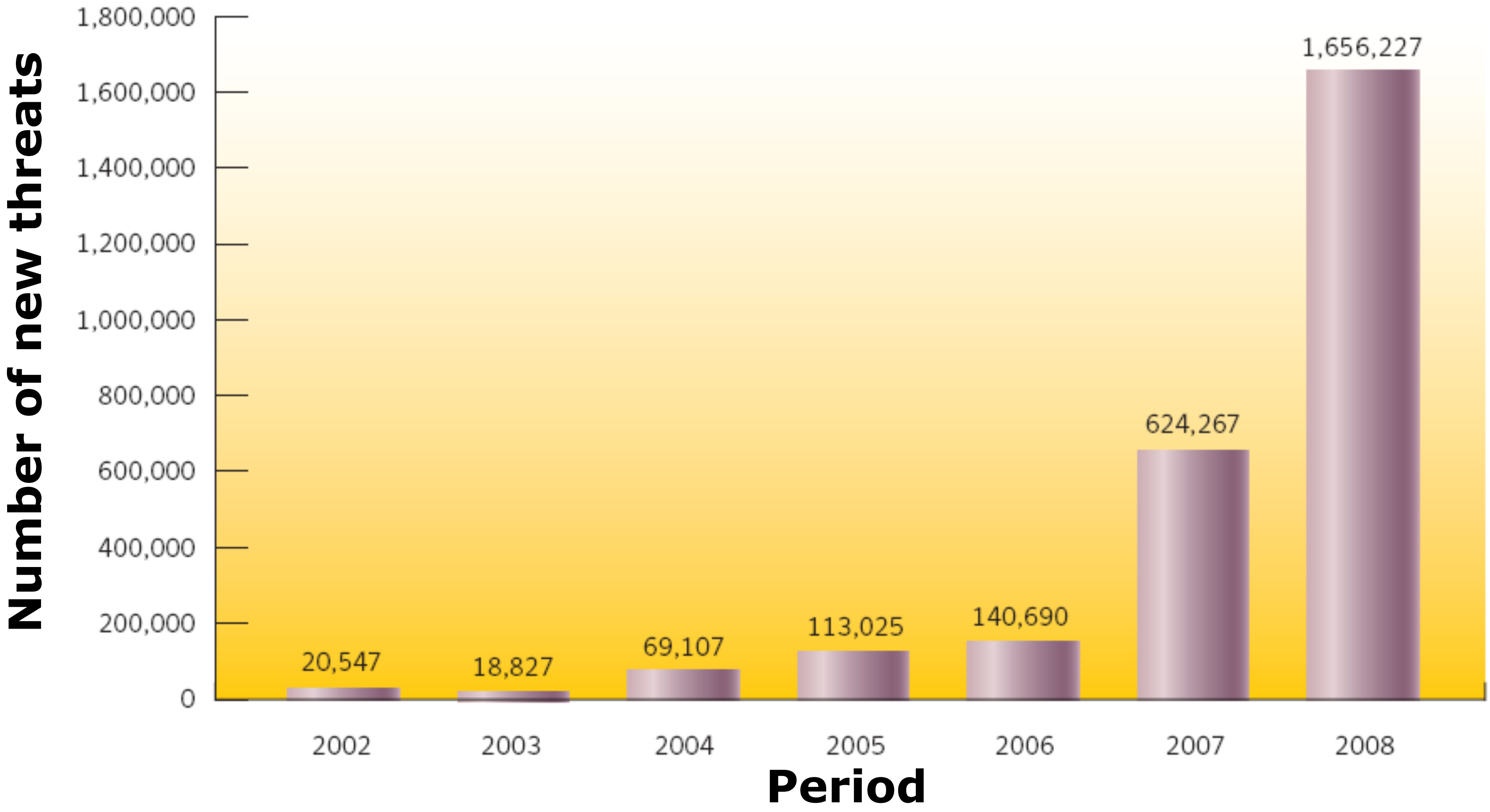
Trojan Horses

Rootkits

Spyware

Malicious Code: Critical Threat

Growth of New Malicious Code Threats



(source: Symantec)

Viruses

Worms

Botnets

Trojan Horses

Rootkits

Spyware

Malicious Code: Critical Threat

Outline

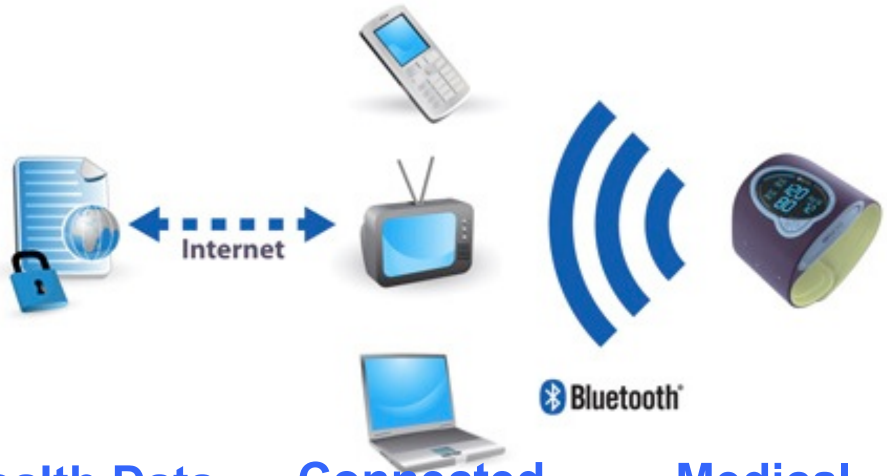
- **Malware: Emerging Threats**

- **Defense: New Approaches**

Malware enters new landscape as more parts of the world get connected

Changing Medical Device Landscape

- **More medical devices are becoming networked**
- **Increased software complexity**
  - **Software plays an increasing role in device failure**
    - **2005-2009 (18%) due to software failure, compared to (6%) in 1980s**
- **Medical device hardware and software is usually a *monoculture* within device model**



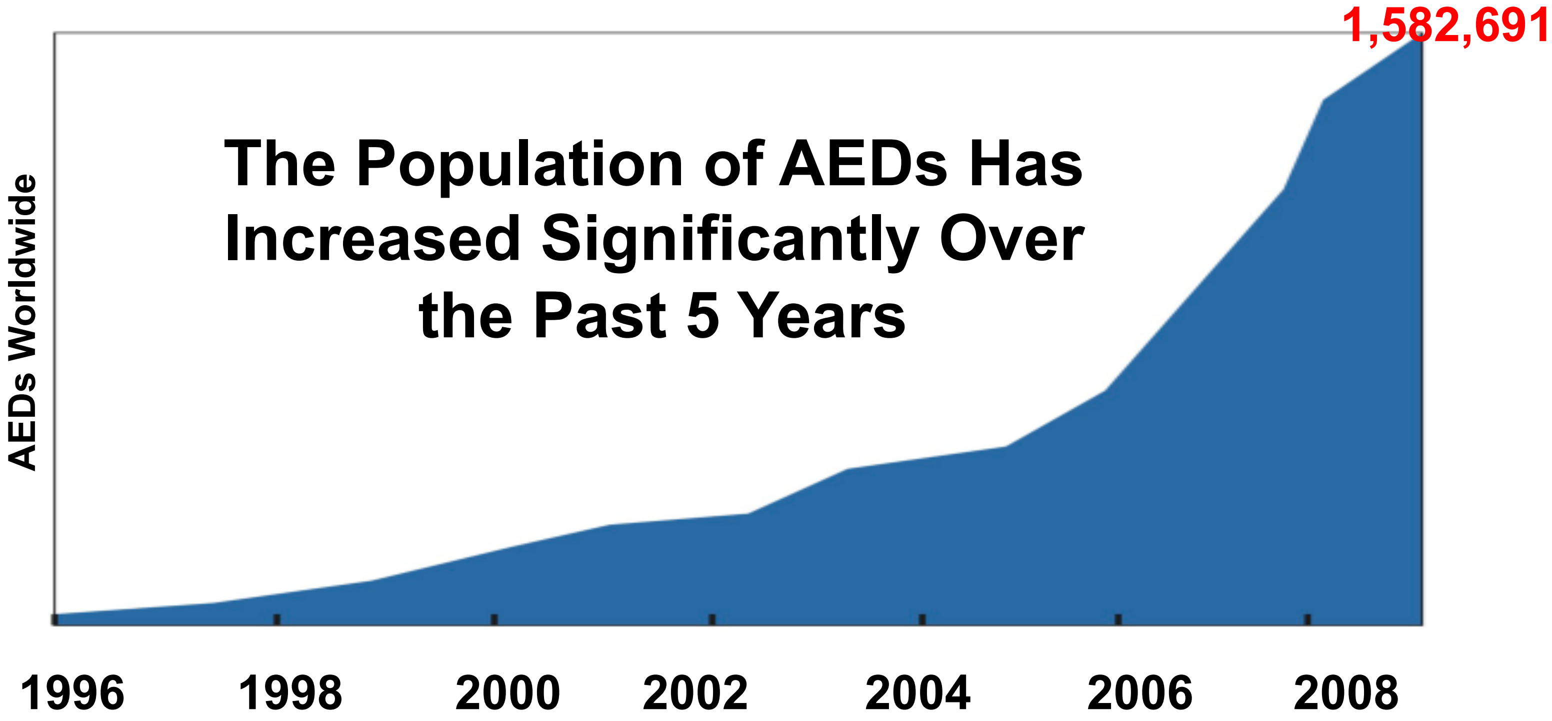
Health Data

Connected Devices

Medical Device

Smart Insulin Pump

Case Study: AED



The Population of AEDs Has Increased Significantly Over the Past 5 Years

Automated External Defibrillator Deployment

28,000 adverse event reports in 14 Models recalled 2005-2010.

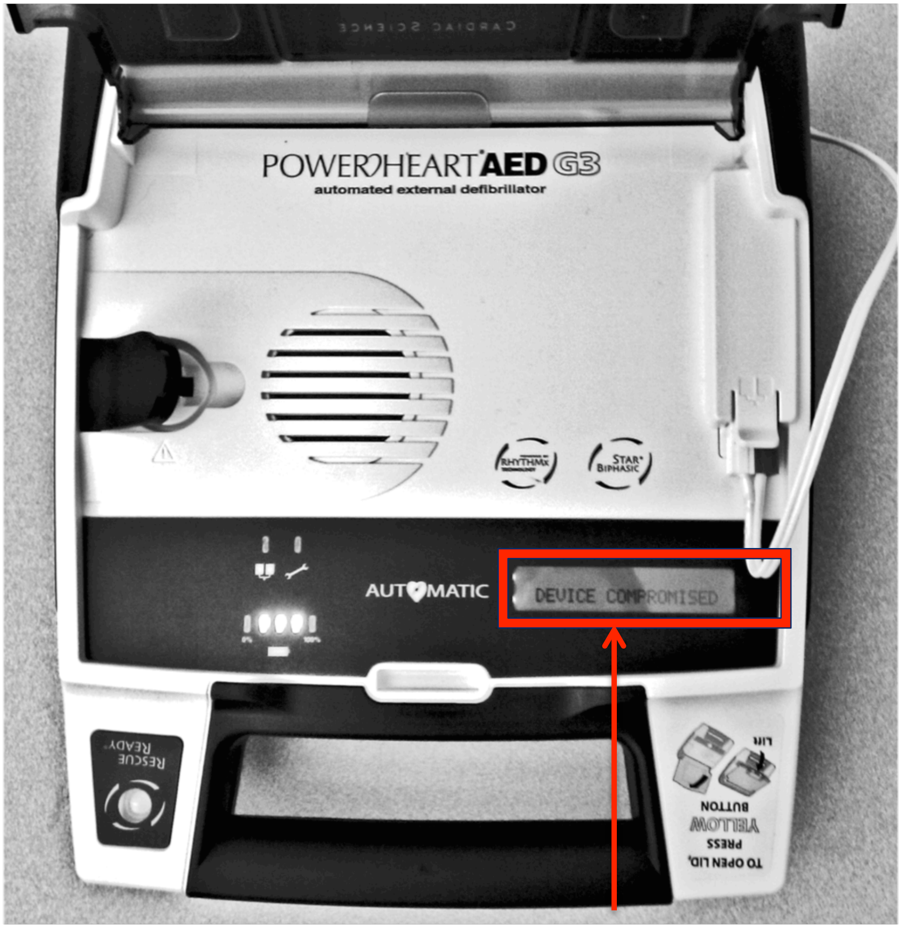
The case for Software Security Evaluations of Medical Devices [HealthSec'11]

Case Study

- Cardiac Science G3 Plus model 9390A
- Analysis
  - Manual reverse engineering using IDA Pro
    - *MDLink*, *AEDUpdate* and device *firmware*
  - Automatic binary analysis
    - BitBlaze binary analysis infrastructure
    - BitFuzz, the dynamic symbolic execution tool
- Vulnerabilities discovered
  1. AED Firmware - Replacement
  2. AEDUpdate - Buffer overflow
  3. AEDUpdate - Plain text user credentials
  4. MDLink - Weak password scheme

The case for Software Security Evaluations of Medical Devices [HealthSec’11]

Firmware Replacement



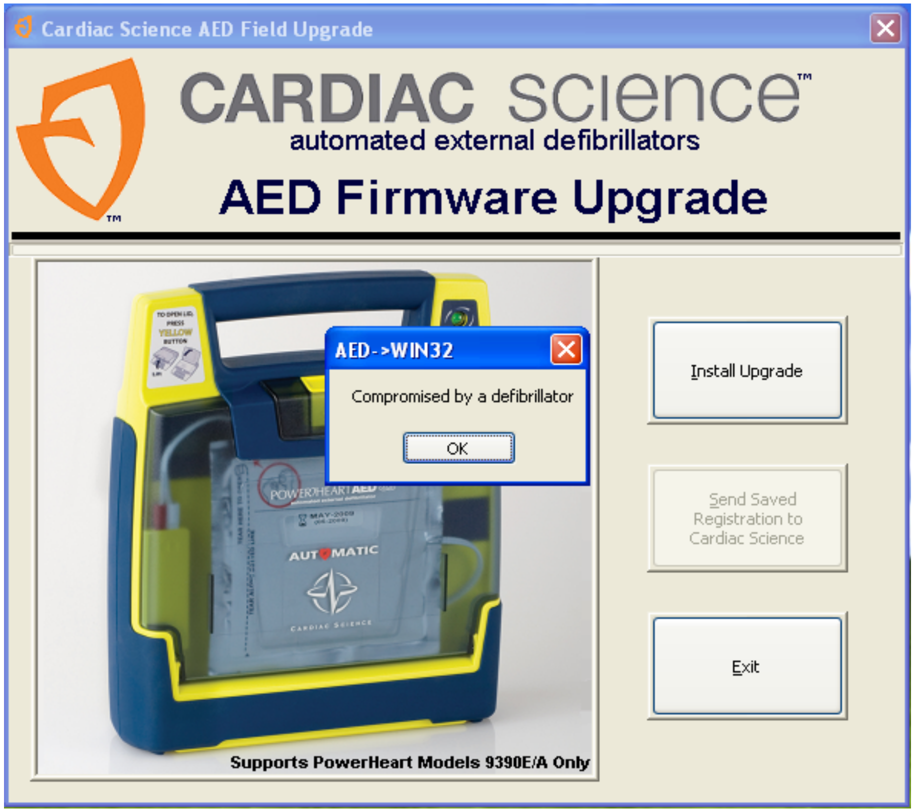
- **Firmware update uses custom CRC to verify firmware**
- **Modified firmware, with proper CRC, is accepted by AED and update software**
- **Impact: Arbitrary firmware**

DEVICE COMPROMISED

The case for Software Security Evaluations of Medical Devices [HealthSec'11]

AEDUpdate Buffer Overflow

- **During update device handshake, device version number exchanged**
- **AEDUpdate *improperly* assumes valid input**
- **Enables arbitrary code execution**
  - **Data sent from AED can be executed as code on the host PC**



Cardiac Science AED Field Upgrade

CARDIAC science™

automated external defibrillators

AED Firmware Upgrade

AED->WIN32

Compromised by a defibrillator

Supports PowerHeart Models 9390E/A Only

OK
Install Upgrade
Send Saved Registration to Cardiac Science
Exit
POWERHEART AED
AUTOMATIC
CARDIAC SCIENCE
TO OPEN LID, PRESS YELLOW BUTTON

The case for Software Security Evaluations of Medical Devices [HealthSec'11]

Initial Malicious Firmware Update

Malicious Update Computer

AED
Software Update

Malicious Firmware

AED
Automated External Defibrillator
Infected
Device 0

The case for Software Security Evaluations of Medical Devices [HealthSec'11]

Consumer-grade BCI Devices

- **Price: ≈ 300 USD**

HEADSET & ACCESSORIES

DEVELOPER & RESEARCH PACKAGES

APP STORE

Exercise Equipment for Your Mind

Experts agree that the human brain should be exercised like other body elements. Use the MindWave with specially designed neuroscience meditation, mental fitness and game applications on your home PC or Mac.

BLINKCHALLENGE

Uses a Emobot interface and it can catch your blink immediately. Try to beat your longest stare! Or how fast can you blink? You just wear the headset and try this game

Rate this product: ★★★☆

$4.95

BUY NOW

ARENA

This is a game that requires you to use the power of your mind against your opponent. To play the game, you must first train your mind to shoot fireballs using the Emotiv PUSH command.

This game supports single and dual player modes. For dual player mode (DUEL) each player will

Rate this product: ★★☆☆

$14.95

BUY NOW

SPIRIT MOUNTAIN DEMO GAME

Experience the fantasy of having supernatural powers and controlling the world with your mind. Your journey will take you through a mythical landscape of forests, temples and an environment that adjusts itself based on how you feel.

Rate this product: ★★★☆

FREE

FREE

DOWN LOAD

MASTER MIND

Master Mind allows users to play their favorite PC games with the power of their mind. Existing PC games such as World of Warcraft™ and Call of Duty™ can now be played with the power of your mind.

★★★☆

$99.00

BUY NOW

MIND MOUSE

Mind Mouse is a revolutionary thought-controlled software application which allows the user to navigate the computer, click and double click to open programs, compose email and send with the power of their mind. *** "NON 'AA

★★★☆

$99.00

BUY NOW

EMOTIV EPOC UNITY3D™ DEVELOPER SUPPORT PACK

This package contains a full Unity3D™ Wrapper for the Emotiv EPOC EmoEngine API and a working demonstration game project and assets.

★★★☆

$79.95

BUY NOW

What if an EEG gaming app is malicious?

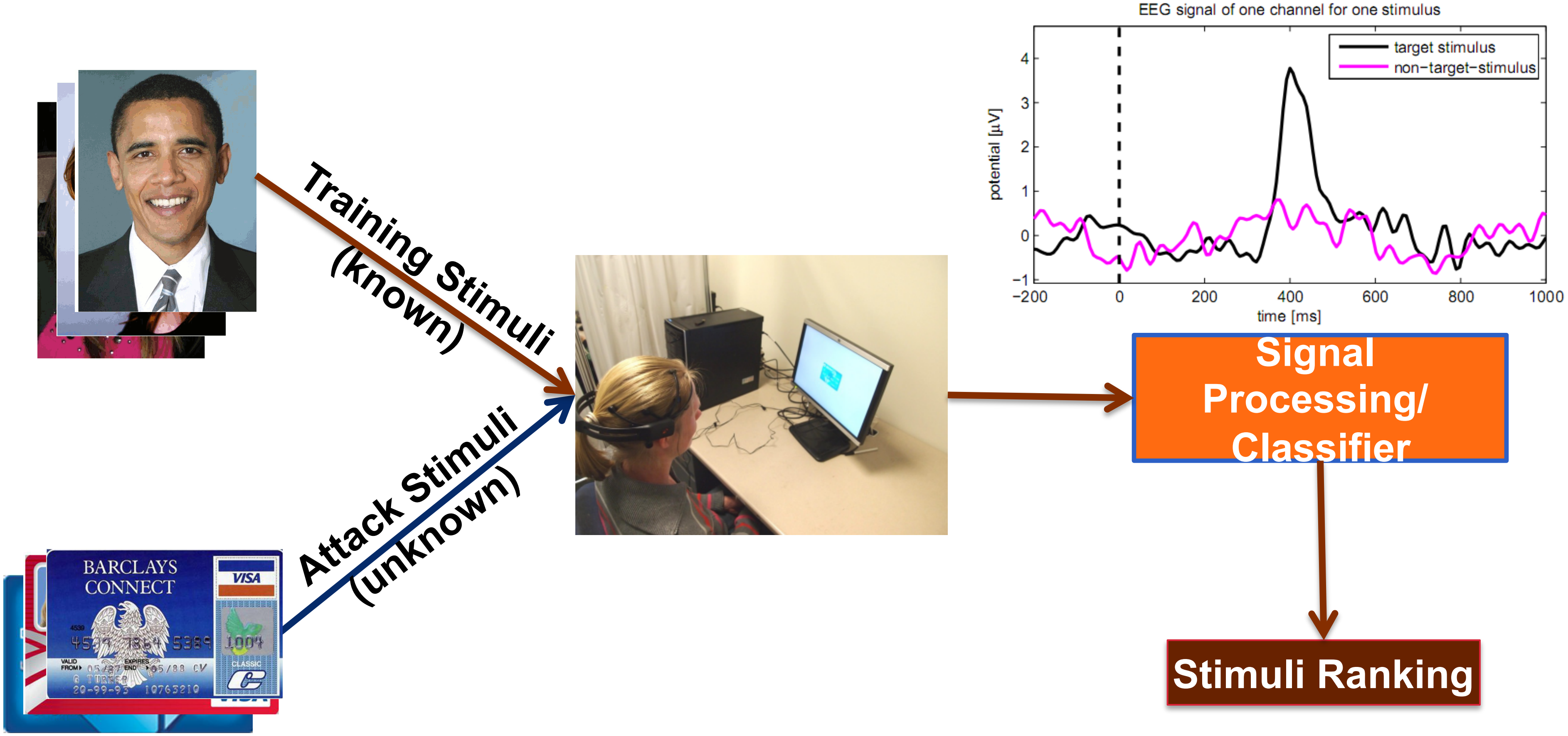
Secretly reading your mind?

BCI as Side-Channel to the Brain

- **Experiment objective:**
  - **Can the signal captured by a consumer-grade EEG device be used to extract potentially sensitive information from the user?**
- **Experiment setup:**
  - **30 EECS students (28)**
    - **18 male and 10 female**
  - **Minimal information: did not provide experiment objective**
  - **Experiments lasted about 45 minutes per participant**
    - **Each experiment lasted about 90 seconds**
      - **Flashing of multiple images on the screen**

On the Feasibility of Side−Channel Attacks with Brain−Computer Interfaces [USENIX Security’12]

Experiment Methodology



Training Stimuli (known)

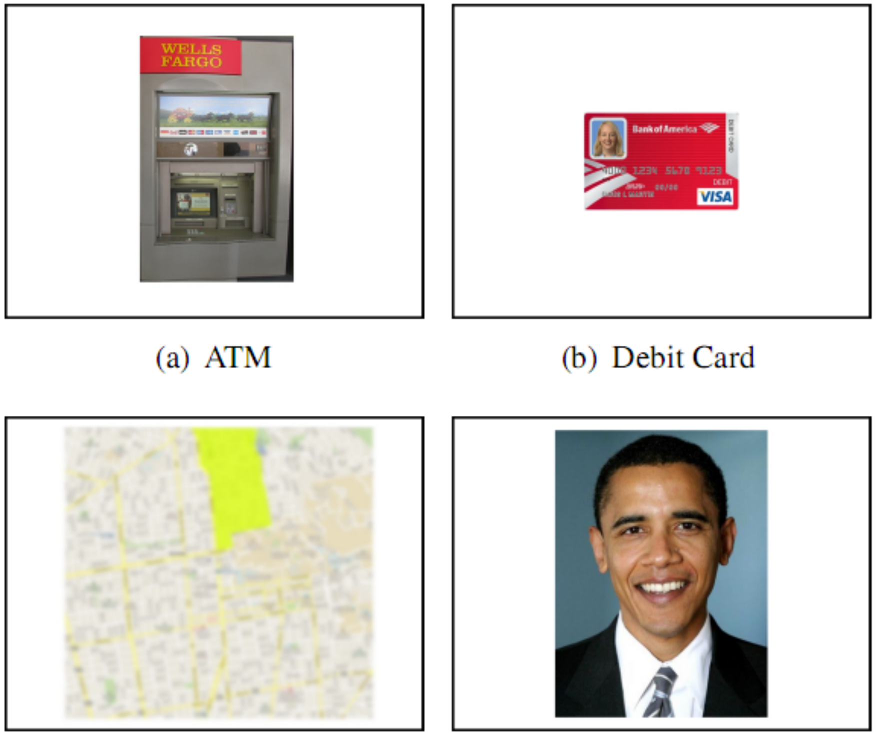
Attack Stimuli (unknown)

Signal Processing/ Classifier

Stimuli Ranking

On the Feasibility of Side−Channel Attacks with Brain−Computer Interfaces [USENIX Security'12]

Attack Stimuli



(a) ATM

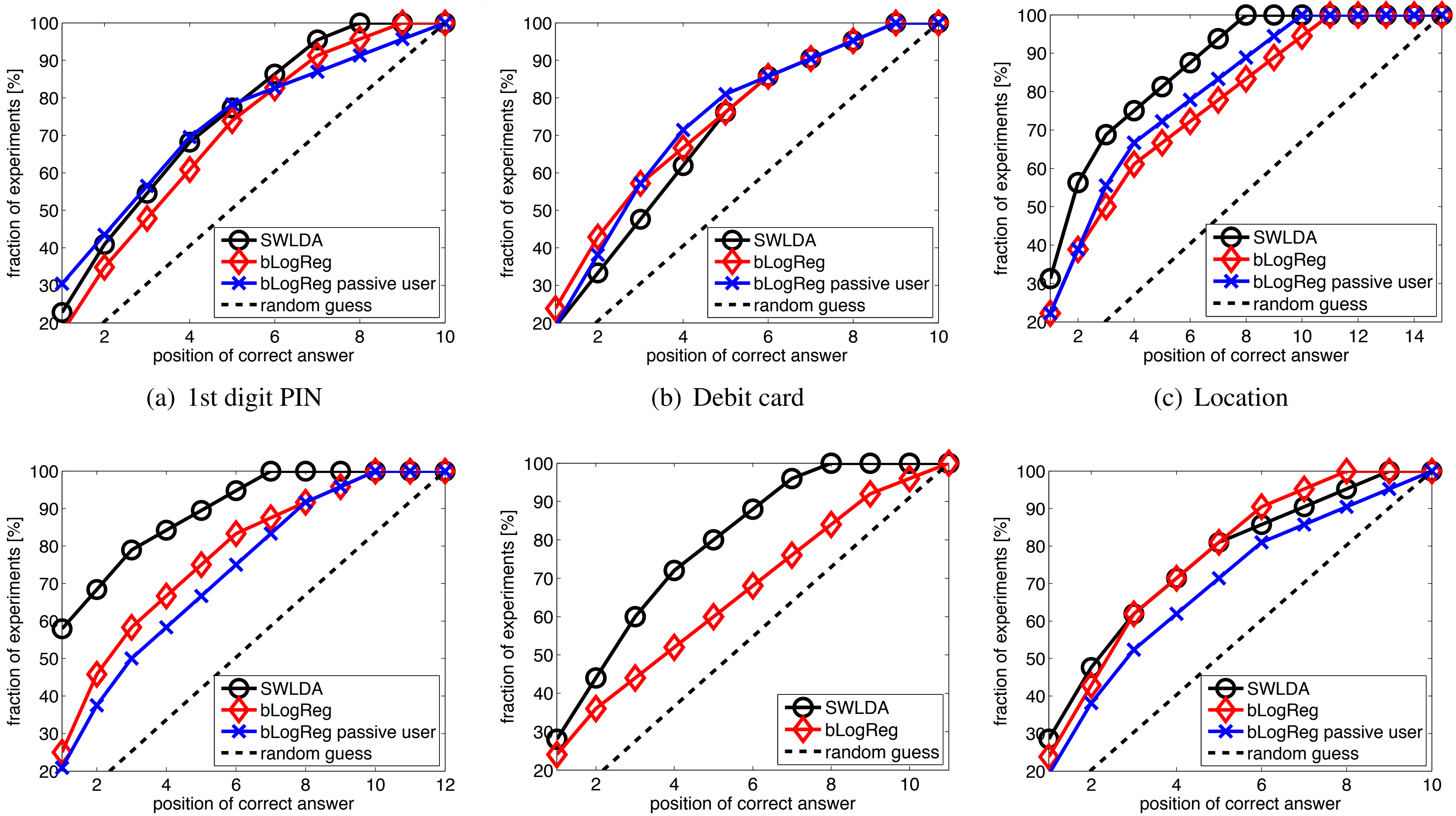
(b) Debit Card

Information tested:

- First digit of PIN
- Do you know this person?
- Do you have an account at this bank?
- What month were you born in?
- Where do you live?

On the Feasibility of Side−Channel Attacks with Brain−Computer Interfaces [USENIX Security'12]

Experimental Results



(a) 1st digit PIN

(b) Debit card

(c) Location

(d) Month of birth

(e) People

(f) ATM machine

On the Feasibility of Side−Channel Attacks with Brain−Computer Interfaces [USENIX Security'12]

Outline

- **Malware: Emerging Threats**

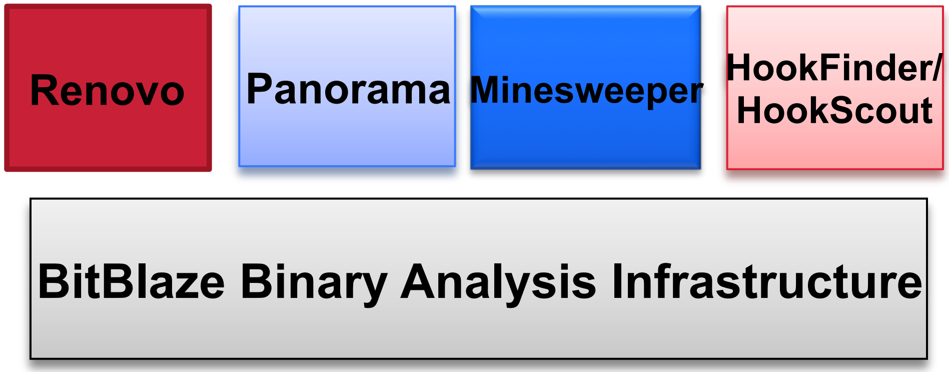
- **Defense: New Approaches**

Defenses

Reactive Approaches

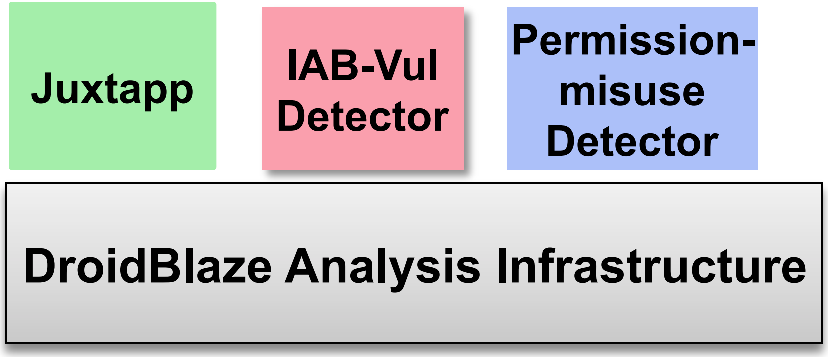
Detecting:

- Hidden code
- Privacy/sensitive data leakage
- Trigger-based behavior
- Hooking behavior

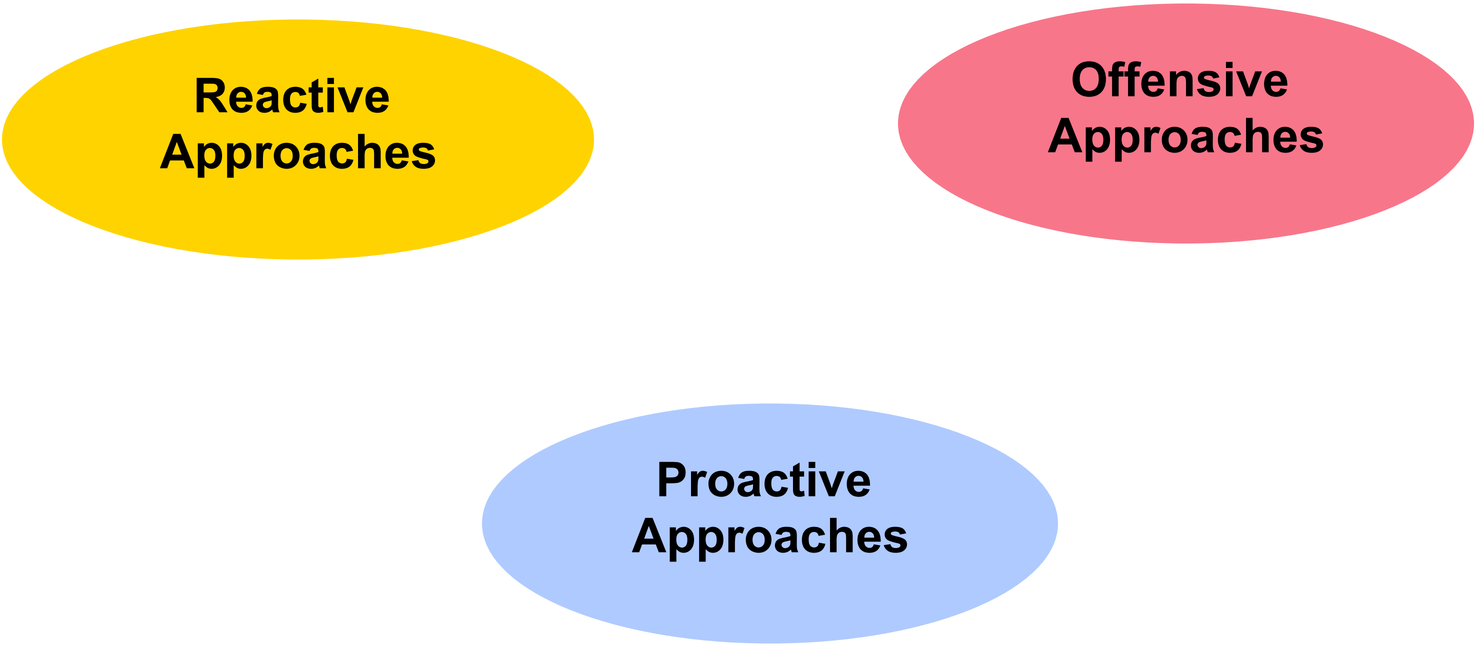


Detecting:

- Code reuse/repackage
- In-App Billing Vulnerability
- Permission misuse
- Security spec violation


Juxtapp
IAB-Vul Detector
Permission-misuse Detector
DroidBlaze Analysis Infrastructure

Defenses



Reactive Approaches

Offensive Approaches

Proactive Approaches

Finding Vulnerabilities in Malware

Offensive Approaches

- Attackers exploit vulnerabilities in benign software
- Does malware have vulnerabilities?
- Can we find vulnerabilities in malware?
- New arsenal to combat malware
  - Cleaning hosts
  - Malware genealogy
  - Botnet infiltration & take-down
  - Cyber warfare

Finding Implementation Vulnerabilities in Malware

Offensive Approaches

- **Decomposition-&-restitching dynamic symbolic execution [BitBlaze]**
- **Compare Stitched vs. Vanilla explorations**
  - **Run both on same malware for 10 hours and find bugs**

Name	Vulnerability Type	Encoding function	Search Time (Stitched)	Search Time (Vanilla)
Zbot	Null dereference	checksum	17.8 sec	>600 min
Zbot	Infinite loop	checksum	129.2 sec	>600 min
MegaD	Process Exit	decryption	8.5 sec	>600 min
Gheg	Null dereference	weak decryption	16.6 sec	144.5 sec
Cutwail	Heap Corruption	none	39.4 sec	39.4 sec

Input Generation via Decomposition and Re-Stitching: Finding Bugs in Malware [CCS'10]

Experimental Results: Bug Persistency

Offensive Approaches

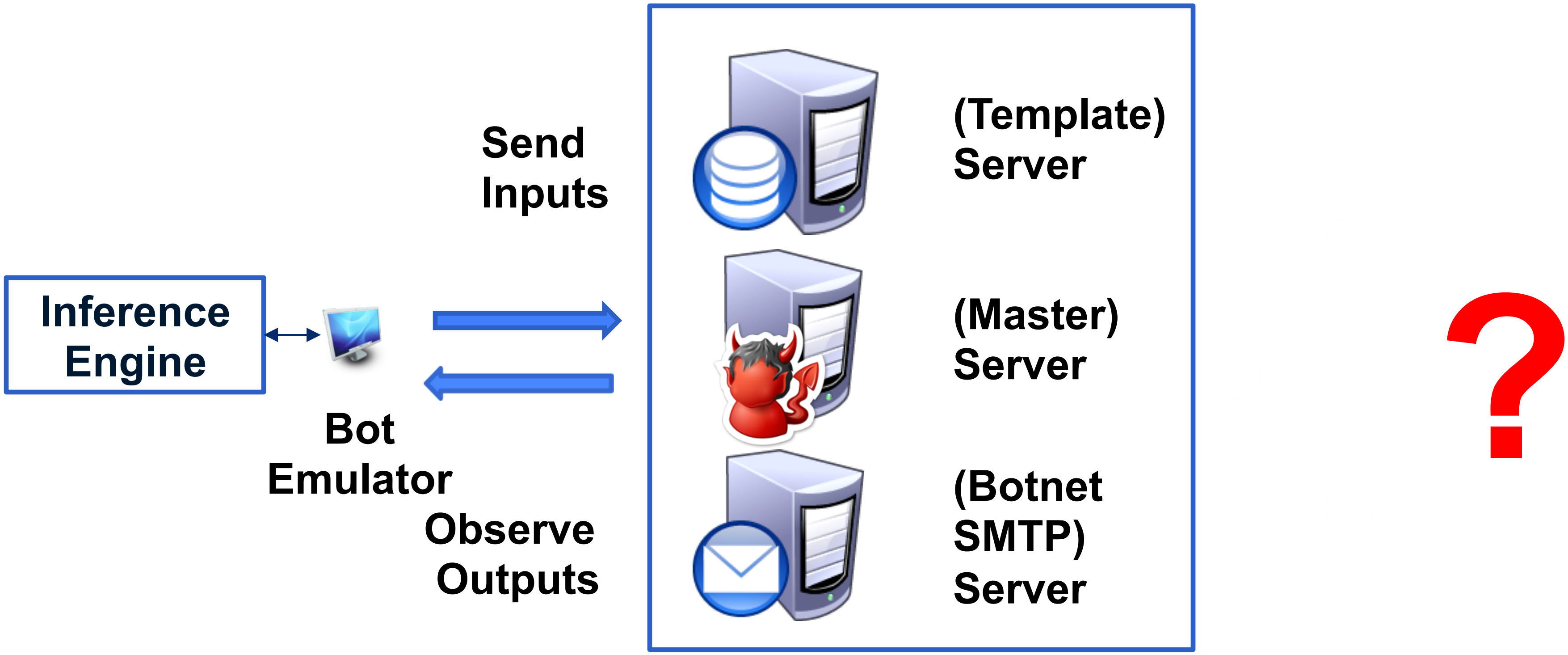
- Each malware family comprises many binaries over time
  - Packing, functionality changes …
- Bugs have been present in malware families for long time

Name	Number of Binaries	Bug reproducibility	Newest	Oldest
MegaD	4	~2 years	Feb. 24, 2010	Feb. 22, 2008
Gheg	5	~9.5 months	Nov. 28, 2008	Feb. 6, 2008
Zbot	3	~6 months	Dec. 14, 2009	Jun. 23, 2009
Cutwail	2	~3 months	Nov. 5, 2009	Aug. 3, 2008

Input Generation via Decomposition and Re-Stitching: Finding Bugs in Malware [CCS'10]

Protocol Model Inference & Finding Vulnerabilities in Botnet C&C Protocols

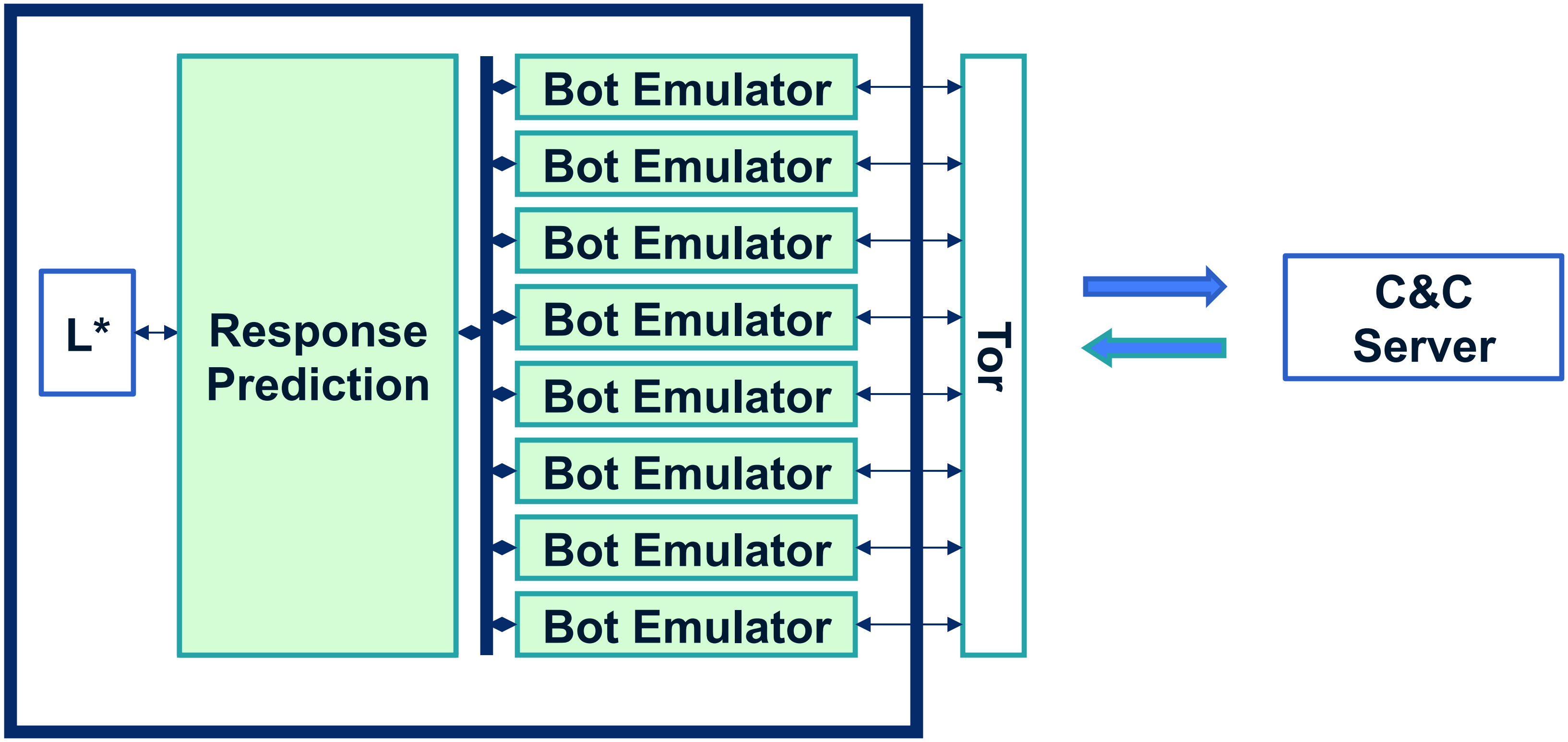
Offensive Approaches



Botnet Command and Control Distributed System

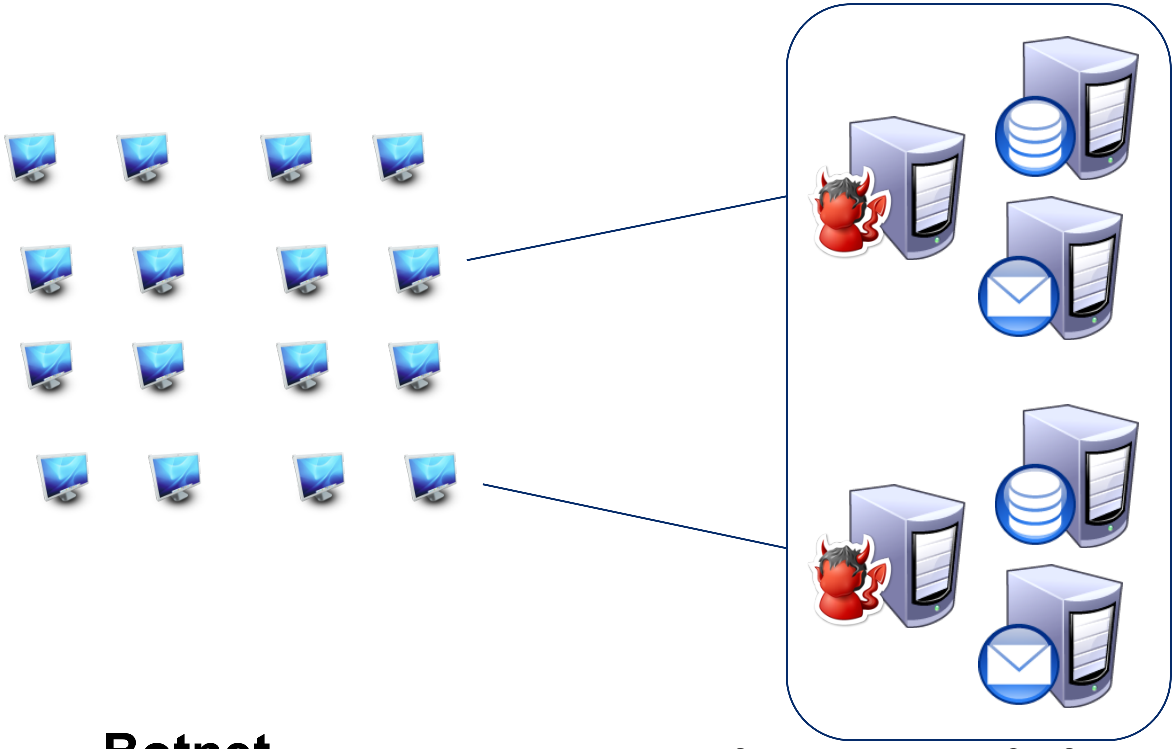
Inference and Analysis of Formal Models of Botnet Command and Control Protocols [CCS'10]

Automatic Protocol Model Inference for MegaD



Inference and Analysis of Formal Models of Botnet Command and Control Protocols [CCS'10]

App 1: Disabling Botnets



Botnet

Command & Control (C&C) Servers

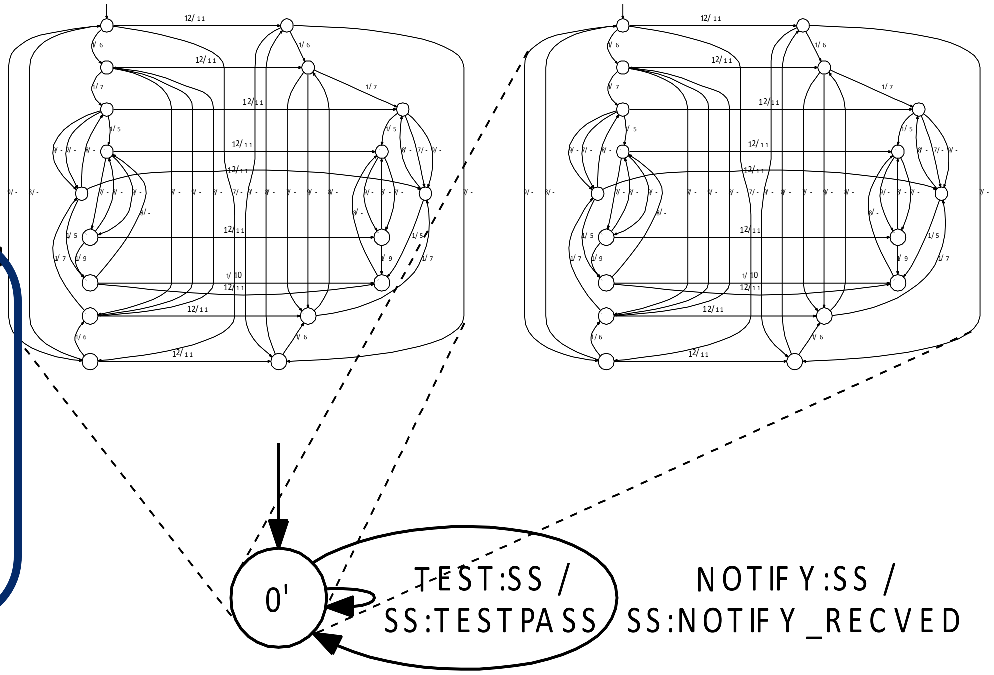
Disable Botnets through Critical Links?

Inference and Analysis of Formal Models of Botnet Command and Control Protocols [CCS'10]

App 1: Disabling Botnets

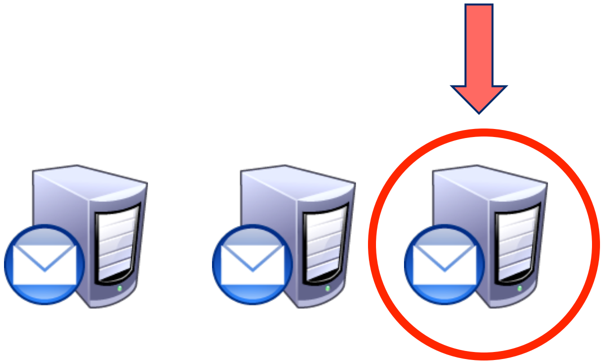
- **Identify Critical Links**

- **Significance**
  - **Taking down *1* MegaD SMTP Server**
  - **Stops bots spam across multiple MegaD C&C server groups**
  - **Validated through experiment**



Inference and Analysis of Formal Models of Botnet Command and Control Protocols [CCS'10]

App 2: Identify MegaD SMTP Servers



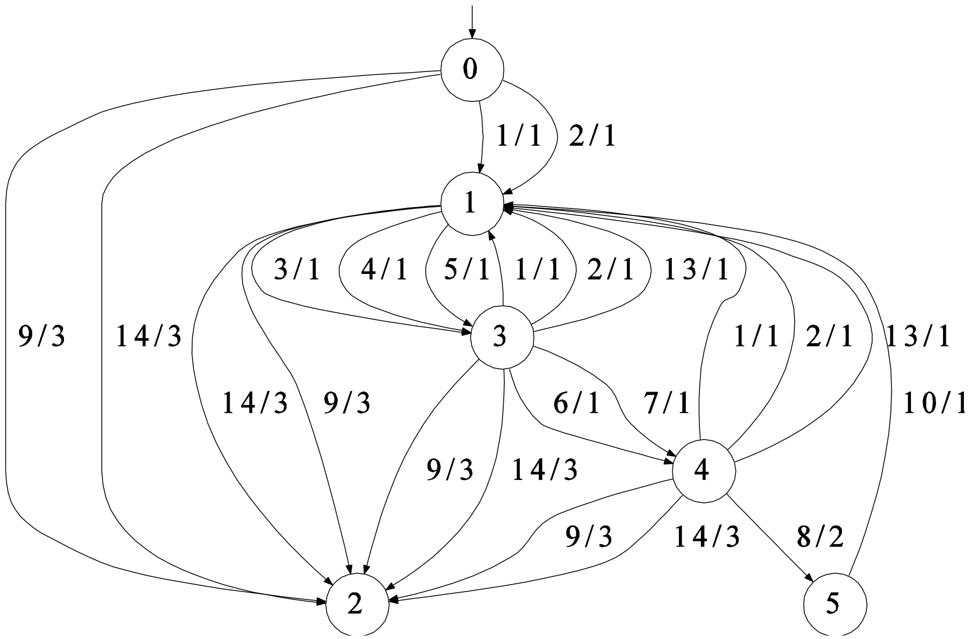
MegaD’s Fake SMTP Server

Inference and Analysis of Formal Models of Botnet Command and Control Protocols [CCS’10]

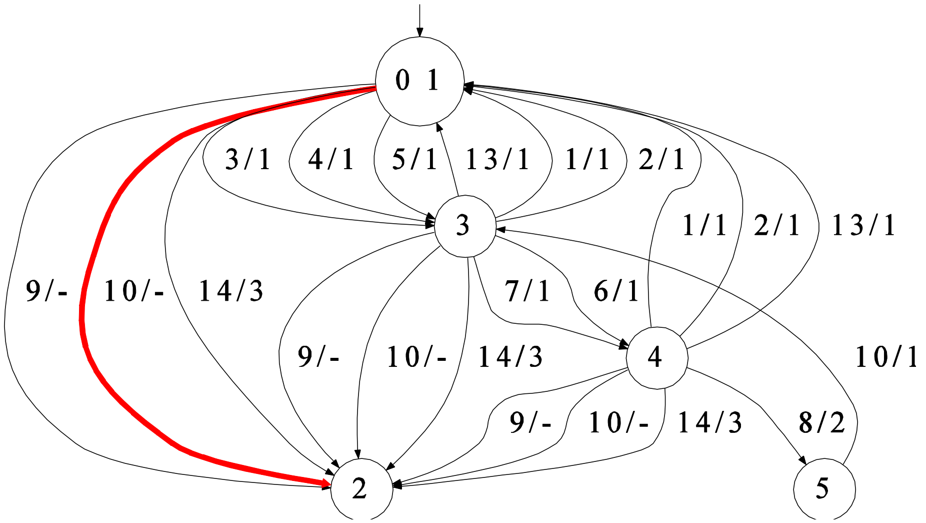
App 2: Implementation Differences

Fingerprint & Identify MegaD SMTPs in the wild

Postfix SMTP 2.5.5

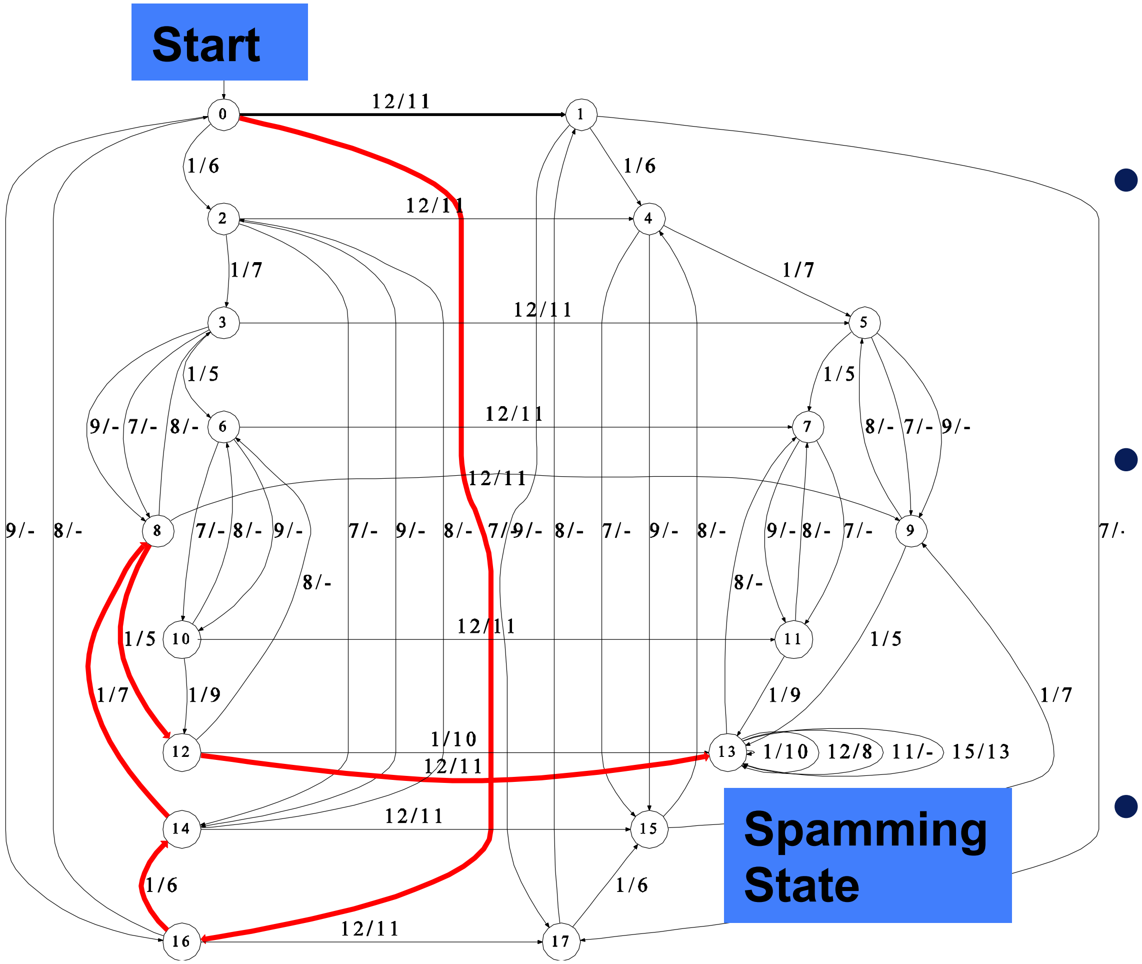


MegaD SMTP



Inference and Analysis of Formal Models of Botnet Command and Control Protocols [CCS'10]

App 3: Identify Design Flaws



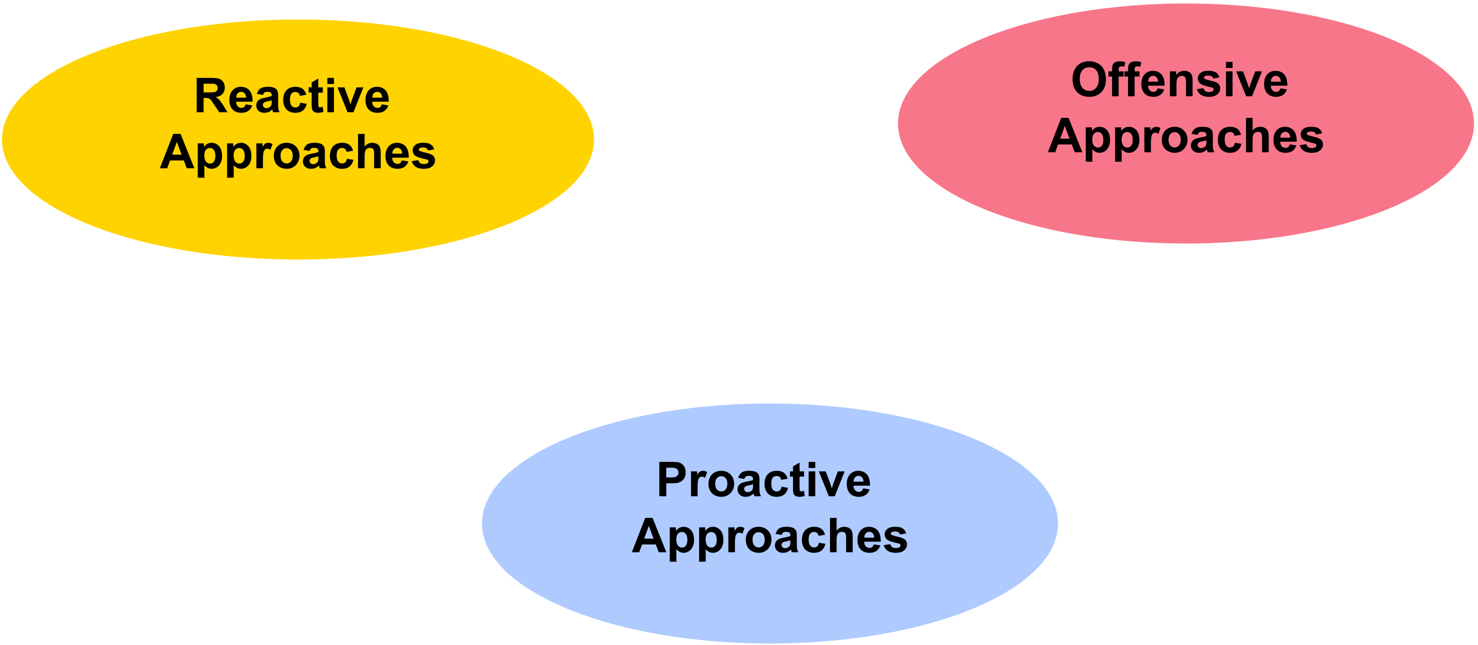
- **Real bot goes through long red path to obtain spam templates**

- **Fake bot may use shortcut: 0 -> 1, bypassing Master Server to loot spam templates**

- **Application [Botnet Judo]: Update spam filtering rules *before* spam is sent out**

Inference and Analysis of Formal Models of Botnet Command and Control Protocols [CCS'10]

Defenses



Reactive Approaches

Offensive Approaches

Proactive Approaches

New Security Primitives

Proactive Approaches

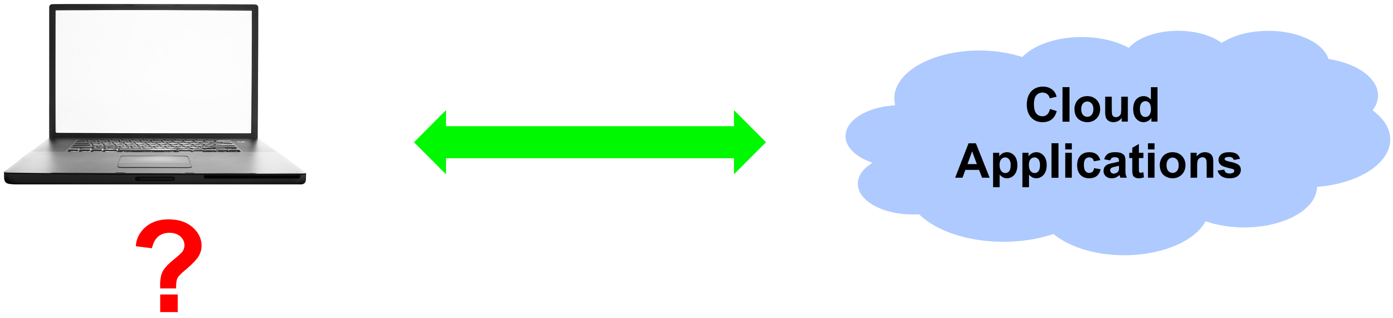
- **For building secure systems even when the machine may be compromised**
  - *Cloud Terminal [USENIX Annual Technical Conf'12]*

- **For building secure applications by design**
  - *Context-sensitive auto-sanitization in web templating languages using type qualifiers [CCS'11]*

- **For better security architecture & auditability**
  - *Privilege separation in HTML5 [USENIX Security'12]*

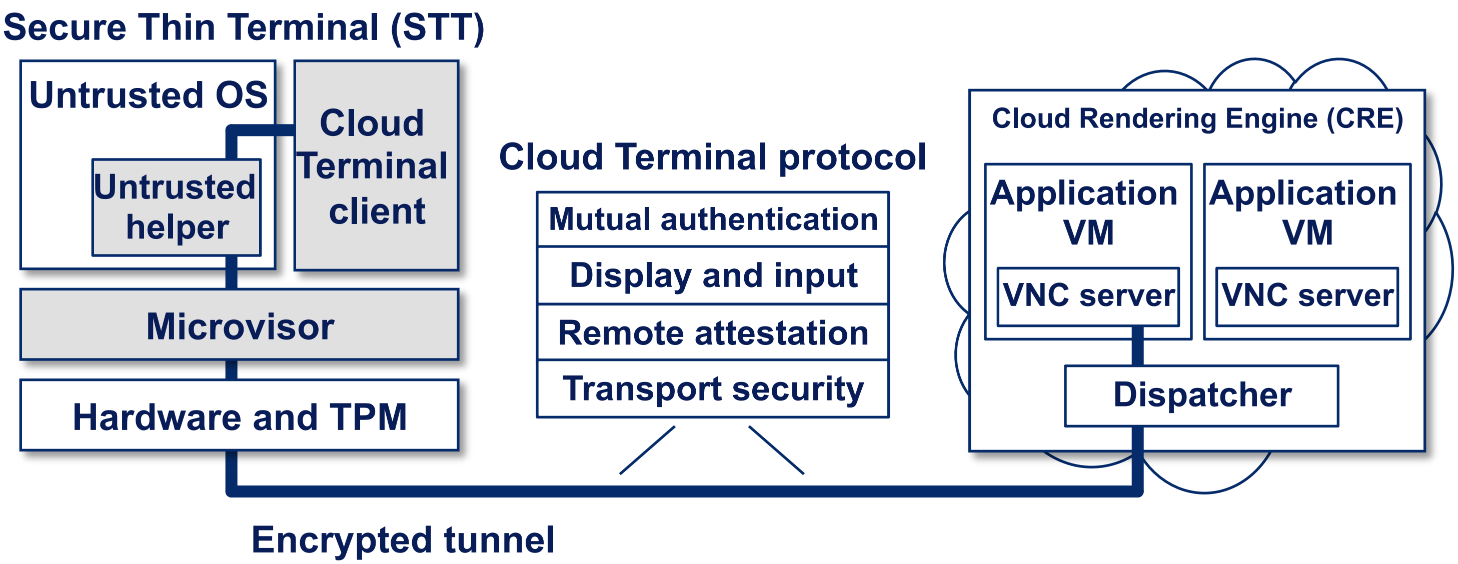
Goal: Trusted Path into the Cloud

- **How to securely access & interact with cloud applications?**
  - **E.g., online banking, enterprise apps**
- **Quickly switch your PC to a secure operation mode**
- **Application provides a normal GUI**
- **But, information security does not depend on primary OS or its software**
  - **Even if commodity OS is compromised by malware**



Cloud Terminal: Secure Access to Sensitive Applications from Untrusted Systems [USENIX ATC'12]

Cloud Terminal Architecture



Cloud Terminal: Secure Access to Sensitive Applications from Untrusted Systems [USENIX ATC'12]

Advantages over Existing Approaches

Property	Red/ Green VMs	Per-App VMs	Browser OS (Chrome)	VDI & Thin Client	Flicker	Cloud Terminal
Installable w/existing OS	✗	✗	✗	✓	✓	✓
Attestation	✗	✗	✗	✗	✓	✓
Generic Apps	✓	✓	✗	✓	✗	✓
Fine-grained isolation	✗	✓	✓	✗	✓	✓
No trust in host OS	✓	✓	✗	✗	✓	✓
User interface	any	any	browser	any	✗	any
Mgmt. effort	med.	high	low	low	low	low
TCB size (LOC)	>1M	>1M	>1M	>1M	250 + app logic	22K

Cloud Terminal: Secure Access to Sensitive Applications from Untrusted Systems [USENIX ATC'12]

Evaluation: client TCB

Component	Lines of code
Microvisor	7.7K
Terminal client	3.0K
Crypto (PolarSSL)	5.5K
Attestation (Flicker)	5.7K
Total	**21.9K**

Cloud Terminal: Secure Access to Sensitive Applications from Untrusted Systems [USENIX ATC'12]

Evaluation: performance

- **16 core, 64GB server, 670 mi from client**
- **Simultaneous clients replay recorded usage**

App.	Activity	Baseline (ms)	Latency (ms) with # of clients =			Network usage (bytes)	
			150	200	300	inbound	outbound
Edit	Launch	2,844	2,208	2,441	2,553	487,047	3,888
	Type a key	30	53	50	54	1,607	346
	Move mouse	32	49	59	51	480	138
PDF	Launch	1,699	2,093	2,147	2,493	483,219	2,040
	Scroll	114	1,270	1,380	1,704	352,358	5,497
Bank	Launch	6,911	2,319	2,563	---	490,149	4,680
	New page	1,183	2,610	2,661		415,732	10,939
Gmail	Launch	6,936	2,254	---	---	488,367	3,954
	Display msg.	992	2,254			318,300	8,416

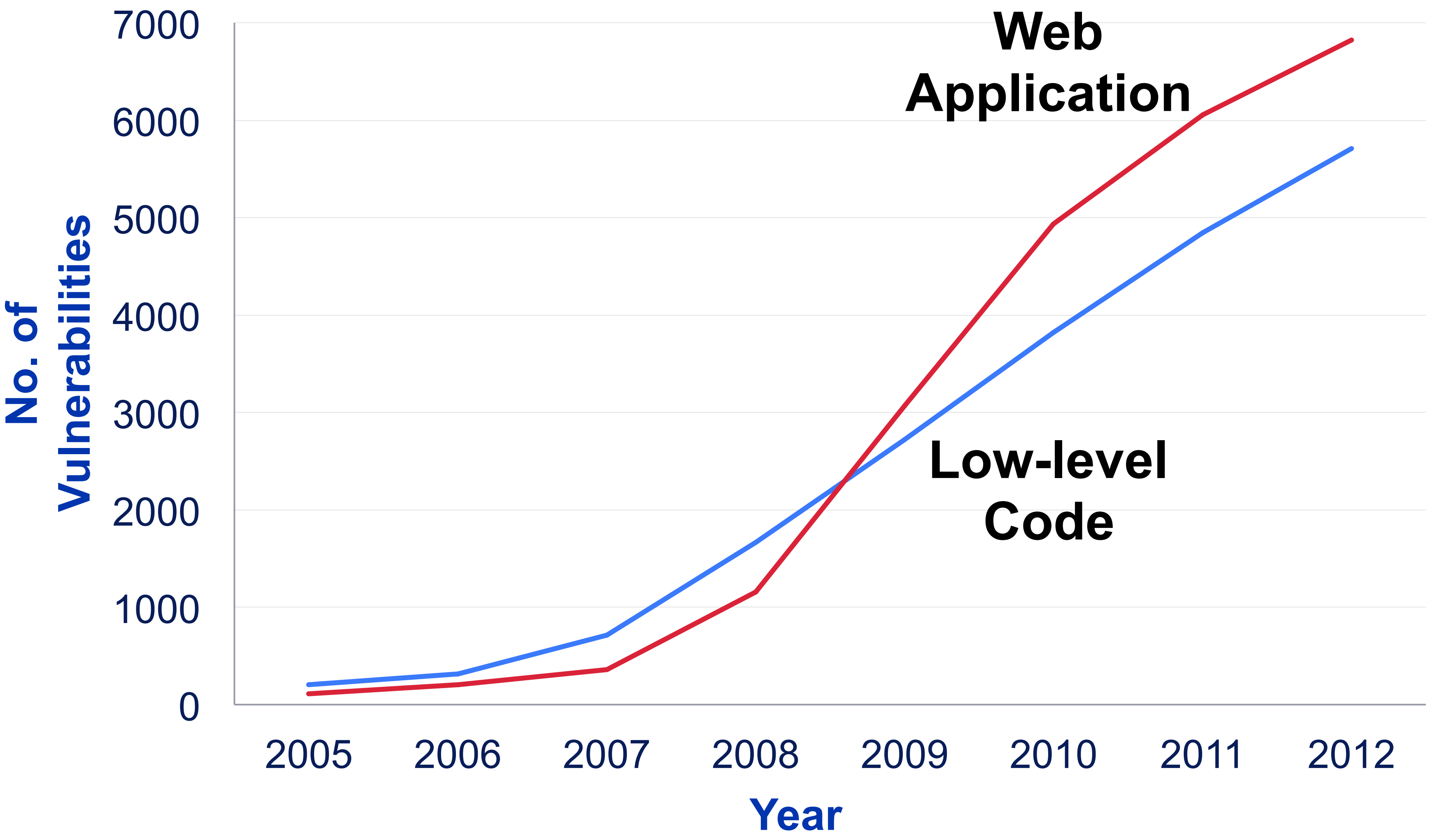
Cloud Terminal: Secure Access to Sensitive Applications from Untrusted Systems [USENIX ATC'12]

New Security Primitives

Proactive Approaches

- **For building secure systems even when the machine may be compromised**
  - *Cloud Terminal [USENIX Annual Technical Conf'12]*
- **For building secure applications by design**
  - *Context-sensitive auto-sanitization in web templating languages using type qualifiers [CCS'11]*
- **For better security architecture & auditability**
  - *Privilege separation in HTML5 [USENIX Security'12]*

Web Vulnerabilities: A Growing Threat

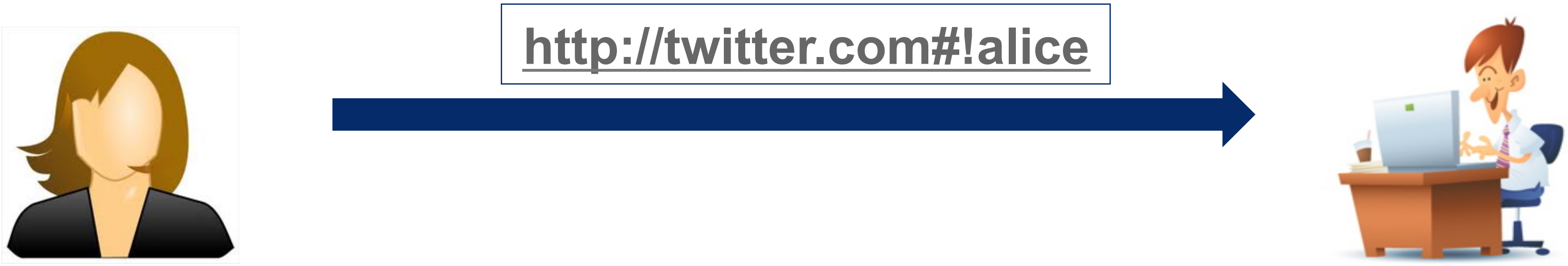


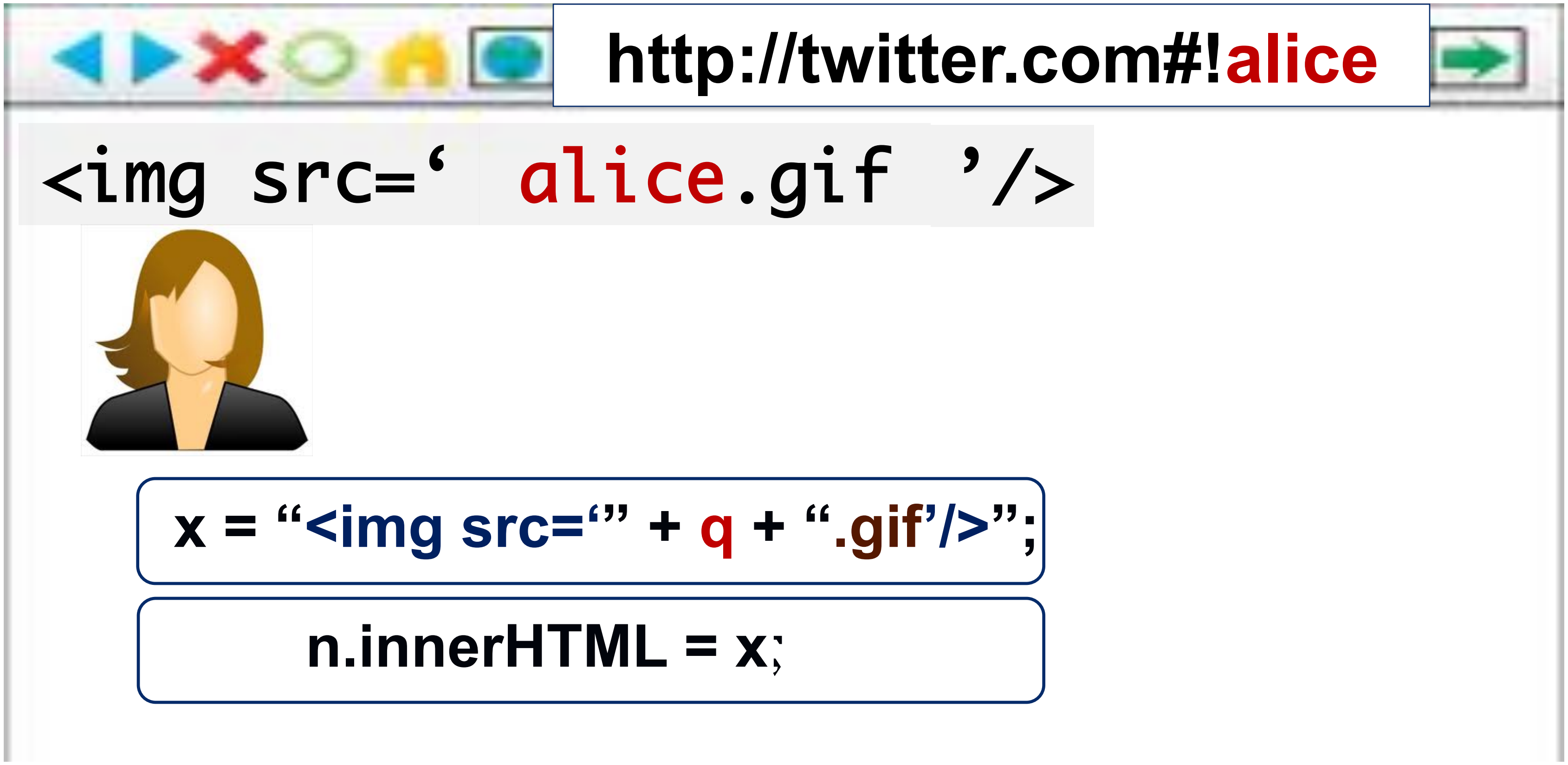
Source: CVE Database 2012

Can never find & fix all XSS vulnerabilities ☹

How to build web apps free of XSS vulnerabilities?

An Attack Example (XSS)



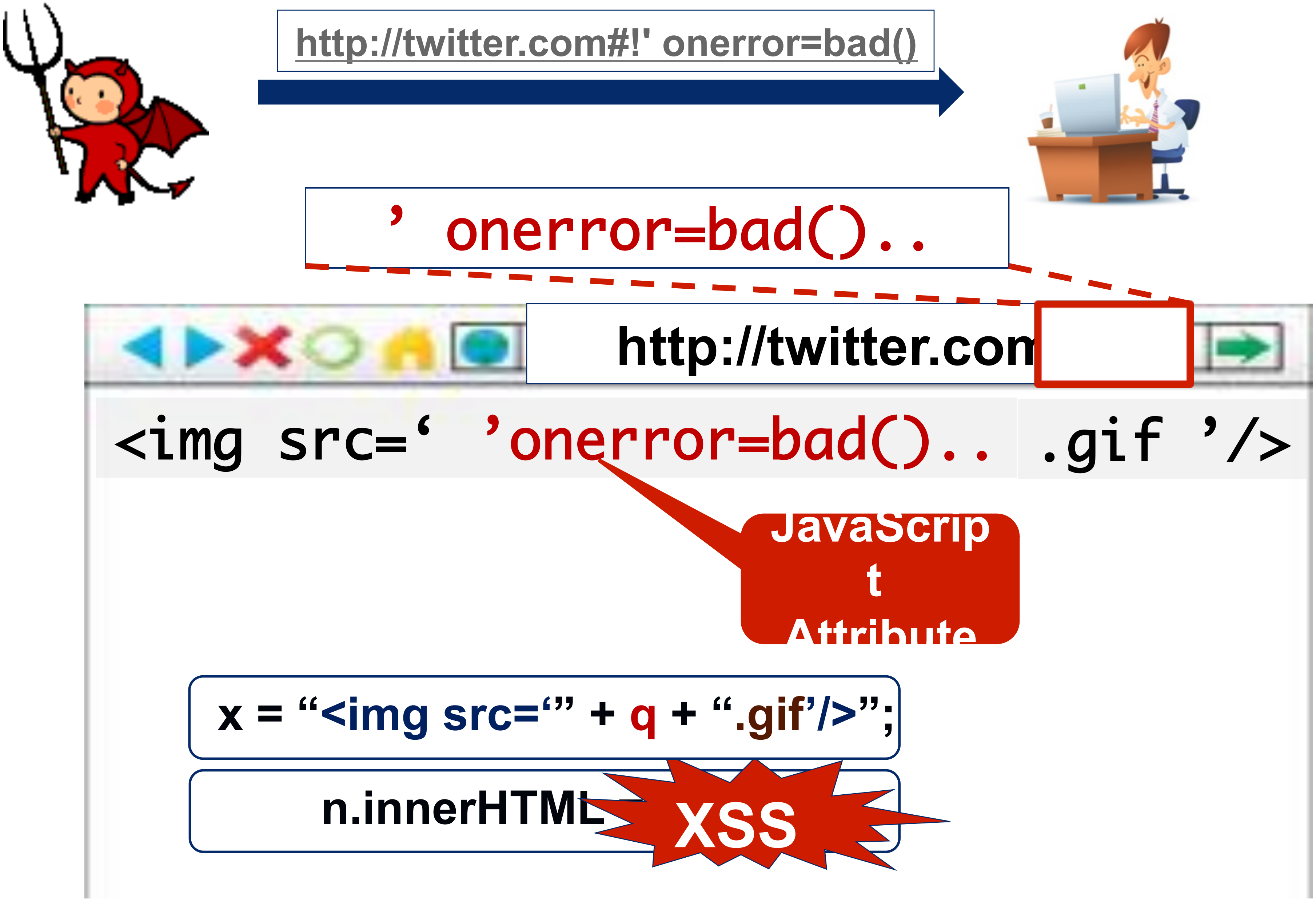


```
<img src=‘ alice.gif ’/>
```

```
x = “<img src=‘” + q + “.gif’/>”;
```

```
n.innerHTML = x;
```

An Attack Example (XSS)



http://twitter.com#!' onerror=bad()

’ onerror=bad()..

http://twitter.com

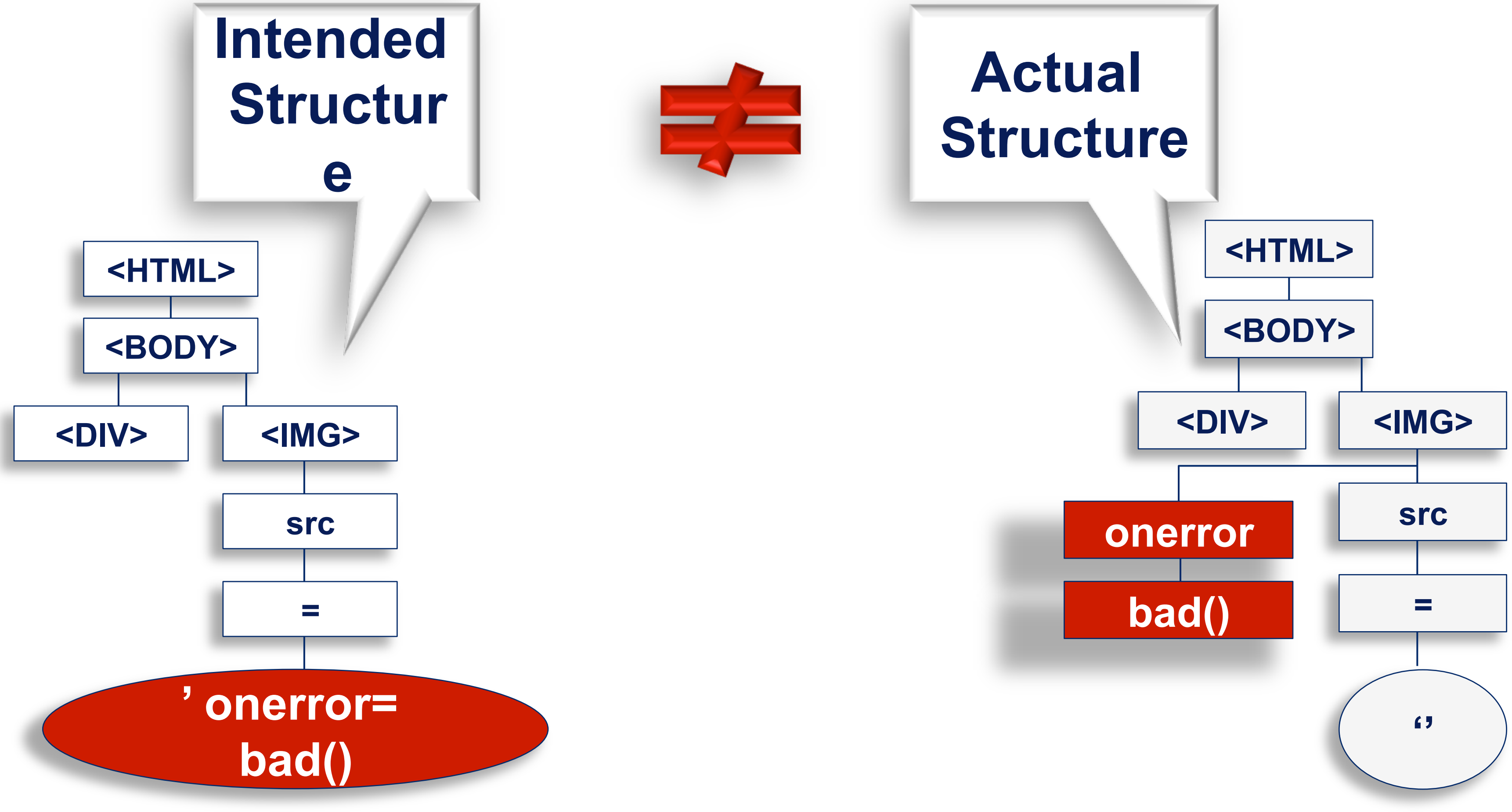
```
<img src=‘ ’onerror=bad().. .gif ’/>
```

```
x = “<img src=‘” + q + “.gif’/>”;
```

```
n.innerHTML
```

Key Property: Structure Integrity

```
<img src=‘   ’ onerror=bad()…   ’></img>
```



Intended
Structur
e

≠

Actual
Structure

<HTML>
<BODY>
<DIV>
<IMG>
src
=
’ onerror=
bad()

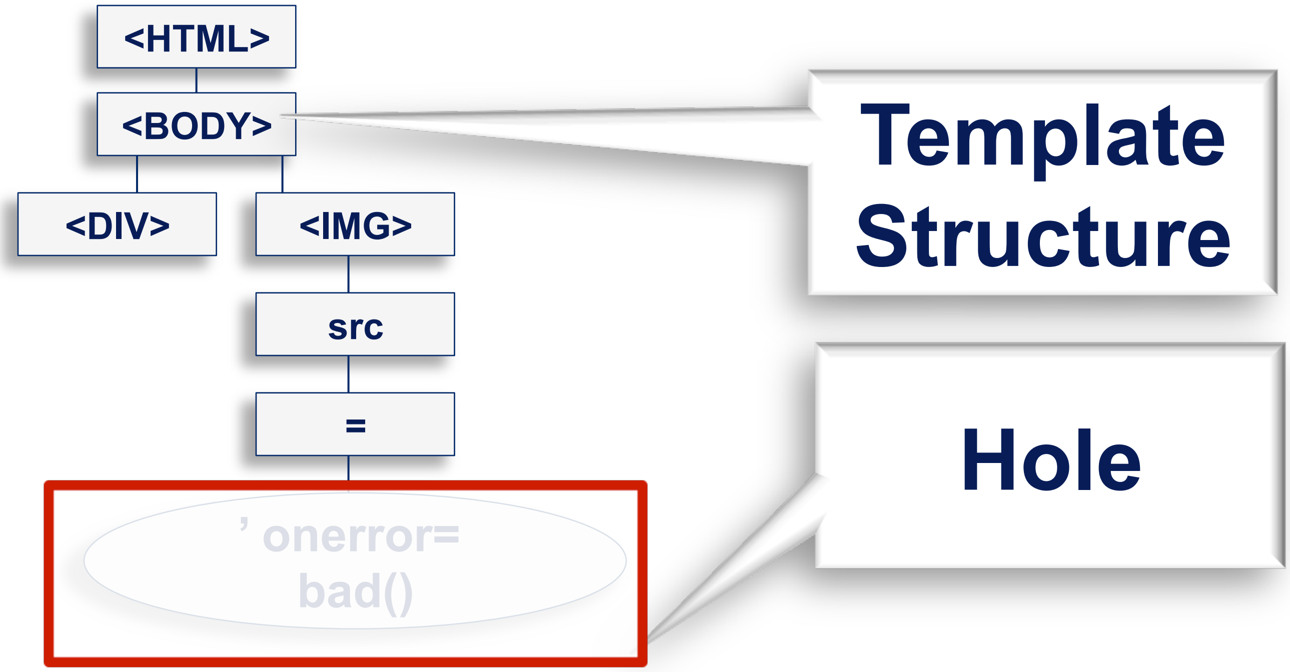
<HTML>
<BODY>
<DIV>
<IMG>
onerror
bad()
src
=
‘’

Structure Integrity Attacks

Web Languages		Structure Integrity Attacks
• SQL	→	SQL Injection
• JS & HTML	→	XSS
• HTTP URLs	→	HTTP Parameter Pollution
• CSS	→	CSS-based XSS
• SVG	→	SVG-based XSS
• MIME Types	→	Content-Sniffing

Solution: Templates & Holes

```
<img src=‘ ’ onerror=bad()… ’></img>
```



Today’s Predominant Enforcement: Sanitization

```
print(“<img src=‘”);
 print(Sanitize(userimg));
       print(“‘></img>”);
```

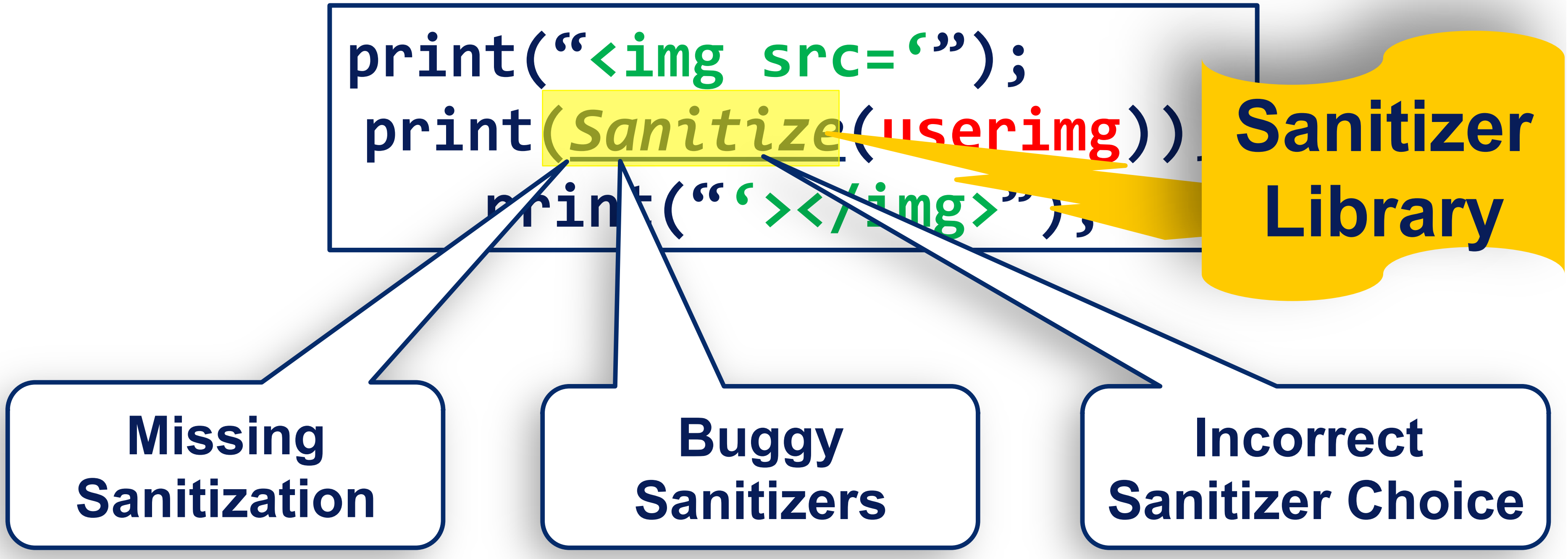
Example

```
<img src=‘  ’ onerror=alert(“XSS”);… ’></img>
```

URL Encode

```
%E2%80%99%20onerror
%3Dalert(%E2%80%9CXSS%E2%80%9D)
%3B%E2%80%A6%0A
```

Challenges:
Getting Sanitization Right



```
print(“<img src=‘”);
 print(Sanitize(userimg));
     print(“‘></img>”);
```

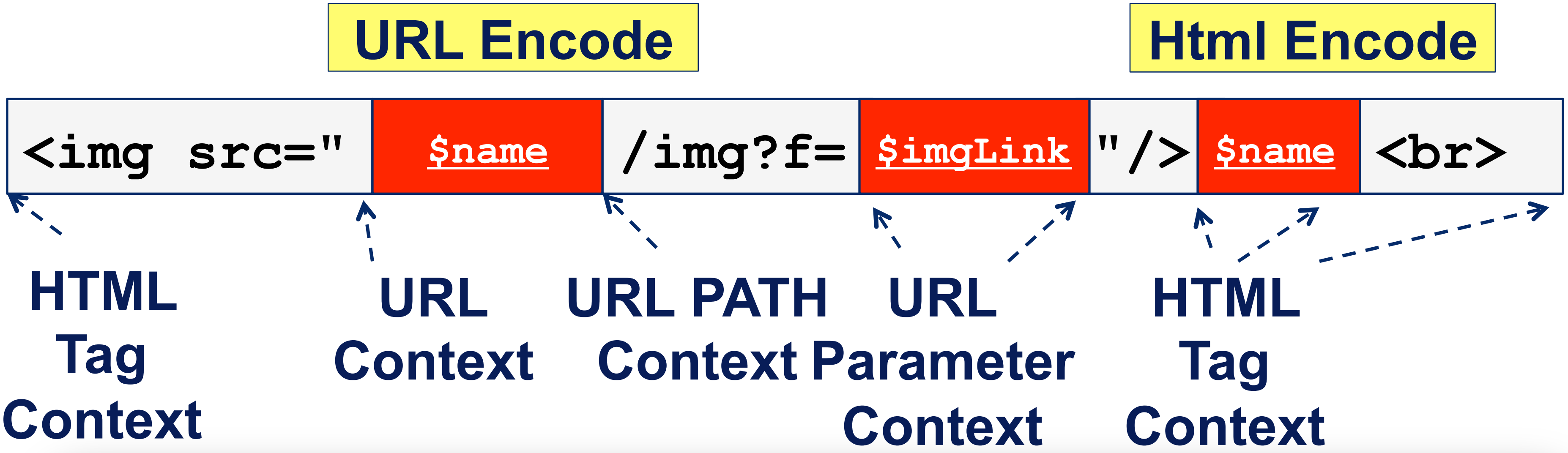
Sanitizer Library

Missing Sanitization

Buggy Sanitizers

Incorrect Sanitizer Choice

Incorrect Sanitizer Choice



URL Encode

Html Encode

```
<img src="$name/img?f=$imgLink"/>$name<br>
```

HTML Tag Context

URL Context

URL PATH Context

URL Parameter Context

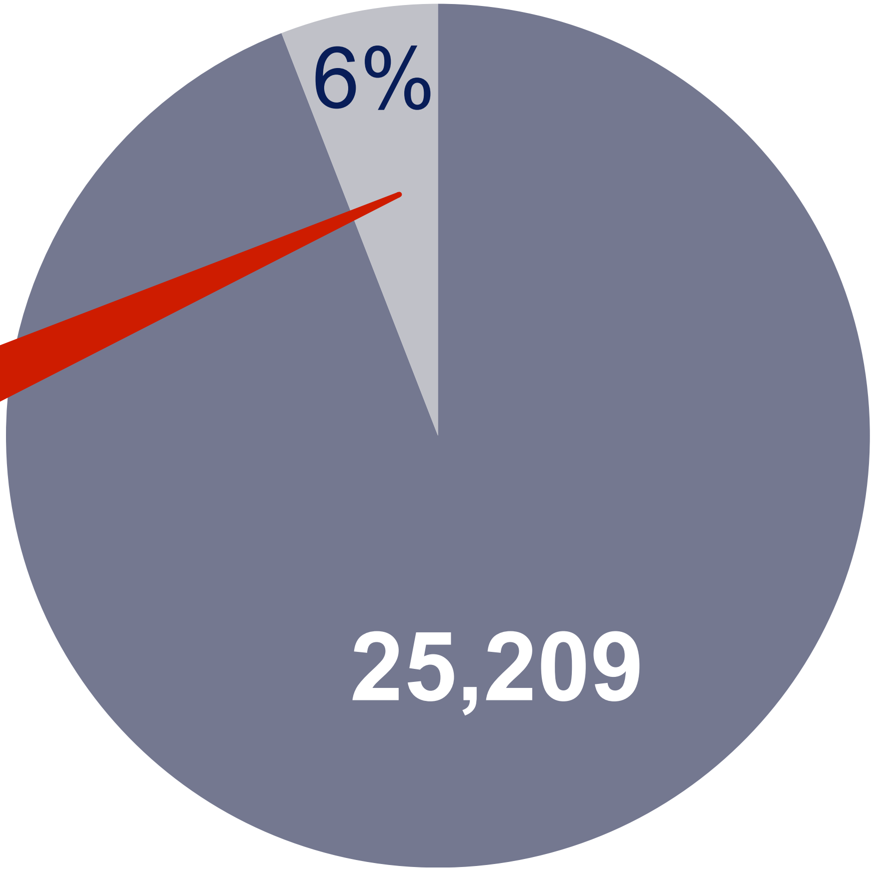
HTML Tag Context

Attacks Vary By Parsing Contexts!

Incorrect Sanitizer Choice

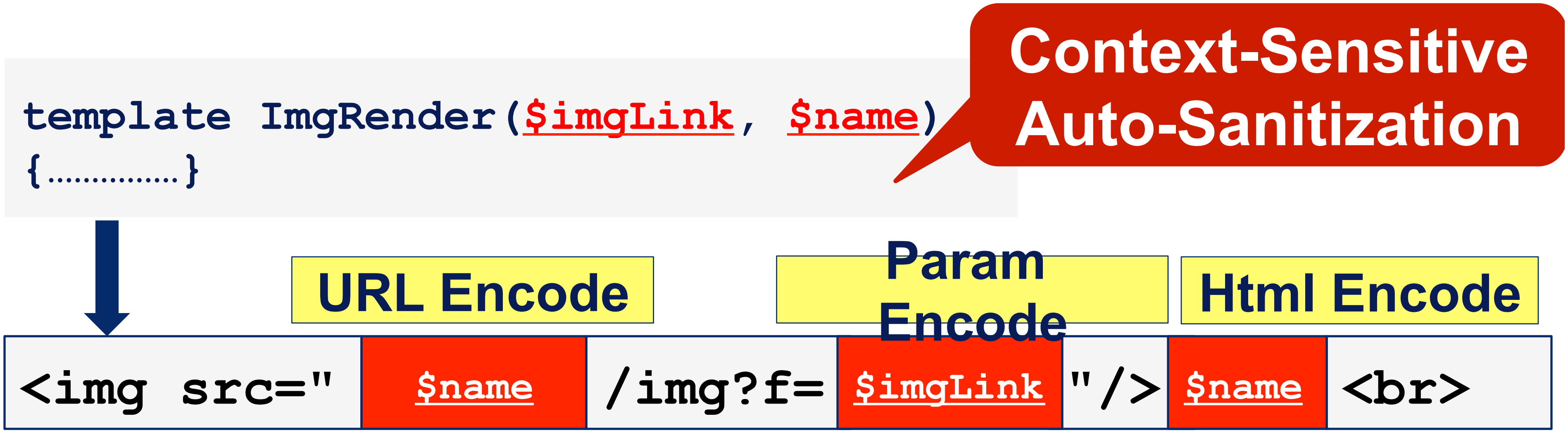
Does manual sanitization really fail?

- **Microsoft shipping .NET applications**
  - **400,000 LOC**
  - ***[Saxena et al. CCS'11]***



Context-Mismatch Sanitization

Our Solution



```
template ImgRender($imgLink, $name)
{..............}
```

Context-Sensitive Auto-Sanitization

	URL Encode		Param Encode		Html Encode	
`<img src="`	`$name`	`/img?f=`	`$imgLink`	`"/>`	`$name`	`<br>`

How To Auto-Sanitize Existing Code?

Compatible **Auditable** **Secure** **Fast**

Context-sensitive auto-sanitization in web templating languages using type qualifiers [CCS'11]

Key Ideas: Context Type Qualifier

- **Context Type Qualifier:**
  - **"Which contexts is a string safe to be rendered in"**

TERMS **TYPES**

```
<img src='
```

🟩 $HTML\_START \hookrightarrow URL\_START$

Type Inference To Decide Sanitizer Placement

```
x:="<img src='" . $imgLink;
```

☒

```
y:= UrlEncode ($imgLink)
```

🟥 ➡ 🟩 $URL\_START \hookrightarrow URL$

```
x:="<img src='" . y;
```

🟩 $HTML\_START \hookrightarrow URL$

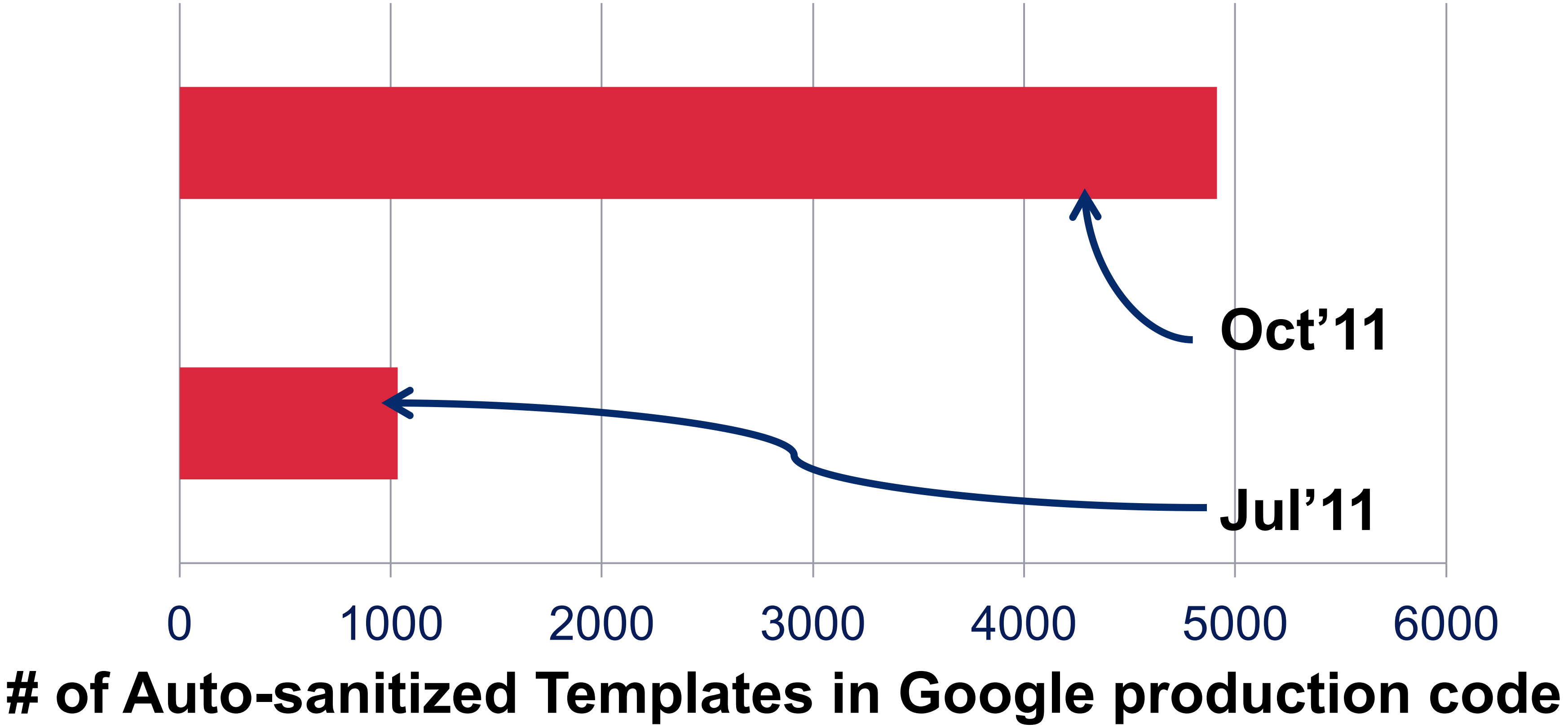
Context-sensitive auto-sanitization in web templating languages using type qualifiers [CCS'11]

Implementation

- **Implemented in Google Closure Templates**

- **Handles Flow-sensitivity**
- **Much faster than Runtime Parsing**

Adoption



In Other Frameworks…

jQuery
write less do more

GO
http://golang.org

GWT

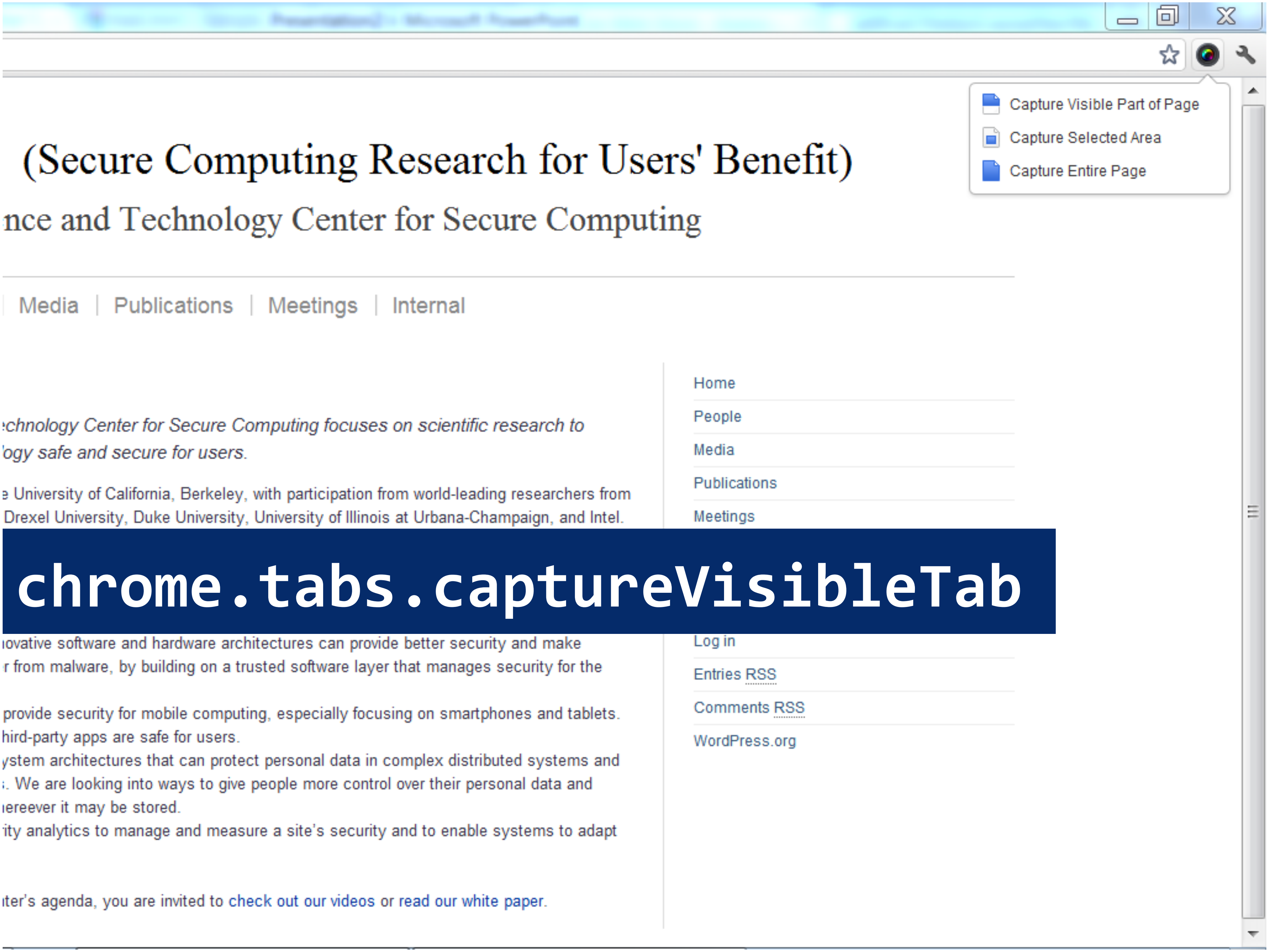
New Security Primitives

Proactive Approaches

- **For building secure systems even when the machine may be compromised**
  - *Cloud Terminal [USENIX Annual Technical Conf'12]*
- **For building secure applications by design**
  - *Context-sensitive auto-sanitization in web templating languages using type qualifiers [CCS'11]*
- **For better security architecture & auditability**
  - *Privilege separation in HTML5 [USENIX Security'12]*

Entire Web Application Code

One security principal with ambient authority(privileges)



(Secure Computing Research for Users' Benefit)

nce and Technology Center for Secure Computing

Media | Publications | Meetings | Internal

:chnology Center for Secure Computing focuses on scientific research to 'ogy safe and secure for users.

e University of California, Berkeley, with participation from world-leading researchers from Drexel University, Duke University, University of Illinois at Urbana-Champaign, and Intel.

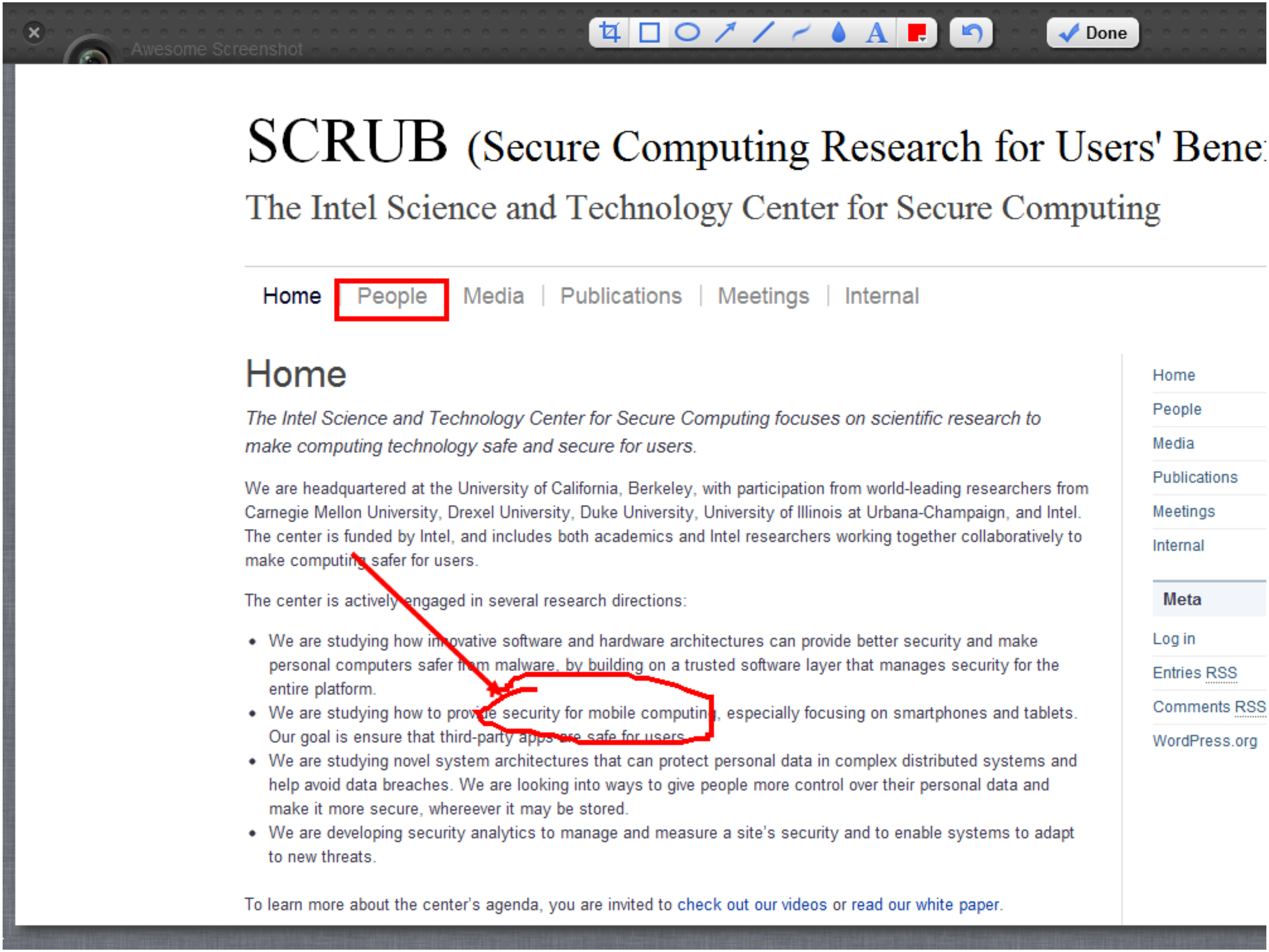
iovative software and hardware architectures can provide better security and make r from malware, by building on a trusted software layer that manages security for the

provide security for mobile computing, especially focusing on smartphones and tablets. hird-party apps are safe for users.
ystem architectures that can protect personal data in complex distributed systems and :. We are looking into ways to give people more control over their personal data and iereever it may be stored.
ity analytics to manage and measure a site's security and to enable systems to adapt

iter's agenda, you are invited to check out our videos or read our white paper.

Home
People
Media
Publications
Meetings
Log in
Entries RSS
Comments RSS
WordPress.org

`chrome.tabs.captureVisibleTab`



SCRUB (Secure Computing Research for Users' Bene

The Intel Science and Technology Center for Secure Computing

Home | People | Media | Publications | Meetings | Internal

Home

The Intel Science and Technology Center for Secure Computing focuses on scientific research to make computing technology safe and secure for users.

We are headquartered at the University of California, Berkeley, with participation from world-leading researchers from Carnegie Mellon University, Drexel University, Duke University, University of Illinois at Urbana-Champaign, and Intel. The center is funded by Intel, and includes both academics and Intel researchers working together collaboratively to make computing safer for users.

The center is actively engaged in several research directions:

- We are studying how innovative software and hardware architectures can provide better security and make personal computers safer from malware, by building on a trusted software layer that manages security for the entire platform.
- We are studying how to provide security for mobile computing, especially focusing on smartphones and tablets. Our goal is ensure that third-party apps are safe for users.
- We are studying novel system architectures that can protect personal data in complex distributed systems and help avoid data breaches. We are looking into ways to give people more control over their personal data and make it more secure, whereever it may be stored.
- We are developing security analytics to manage and measure a site's security and to enable systems to adapt to new threats.

To learn more about the center's agenda, you are invited to check out our videos or read our white paper.

Home
People
Media
Publications
Meetings
Internal

Meta

Log in
Entries RSS
Comments RSS
WordPress.org

580KB of code

permissions ?

all data on all websites

580KB of ~~code~~ TCB

(javascript)

The Problem

- **#1: bundling**
  - **one origin, two applications**

Screenshot Component can save files (doesn’t need to)

Image Editor
can take screenshots
(doesn’t need to)

Not the exception

19 out of top 20 extensions exhibited this behavior

The Problem

- **#1: Bundling**
  - **One origin, two applications**

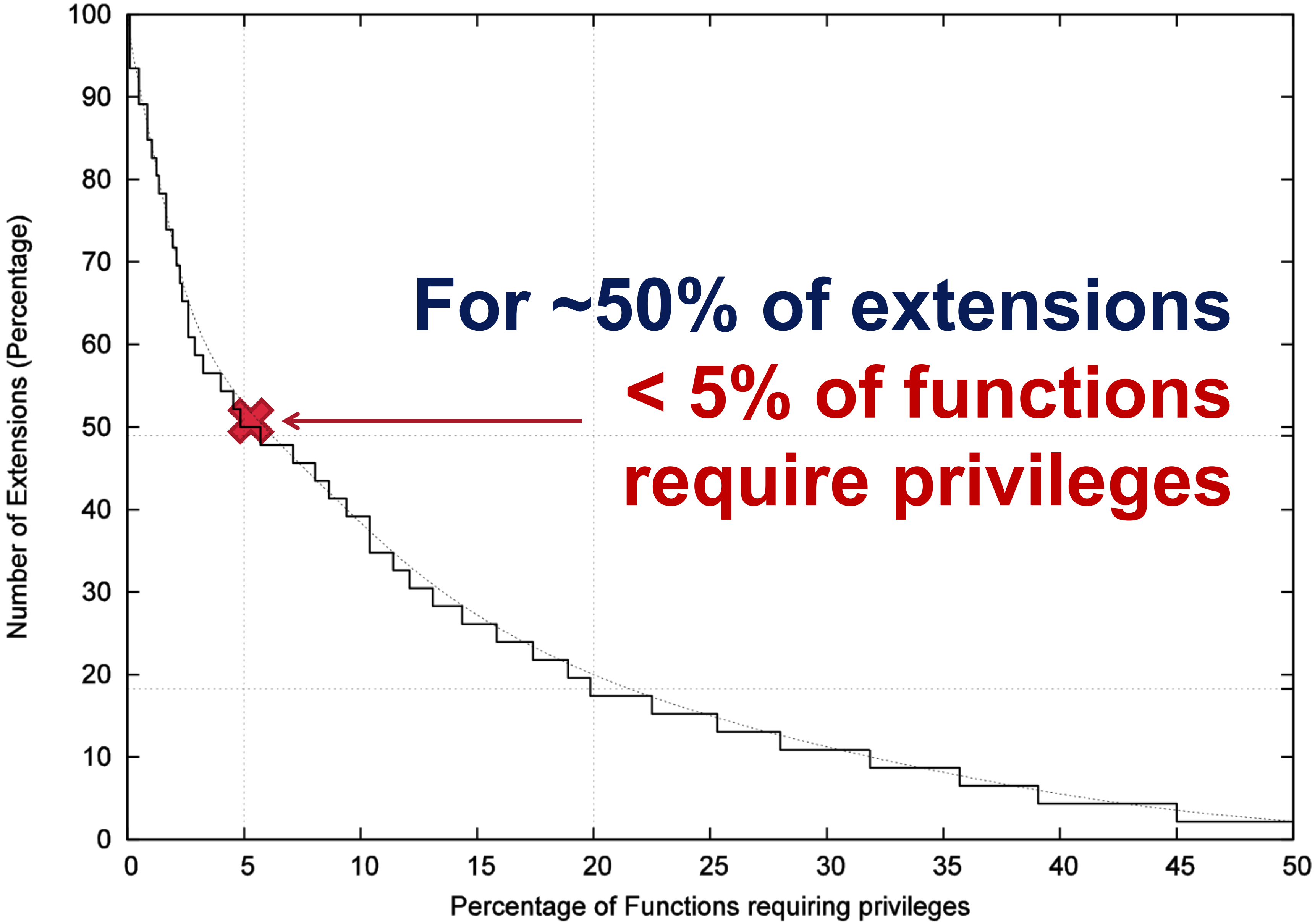
- **#2: TCB inflation**
  - **All code runs with full privileges**
  - **Only core application needs to**

580KB of TCB

500KB generic libraries
(jquery, jquery-ui, …)

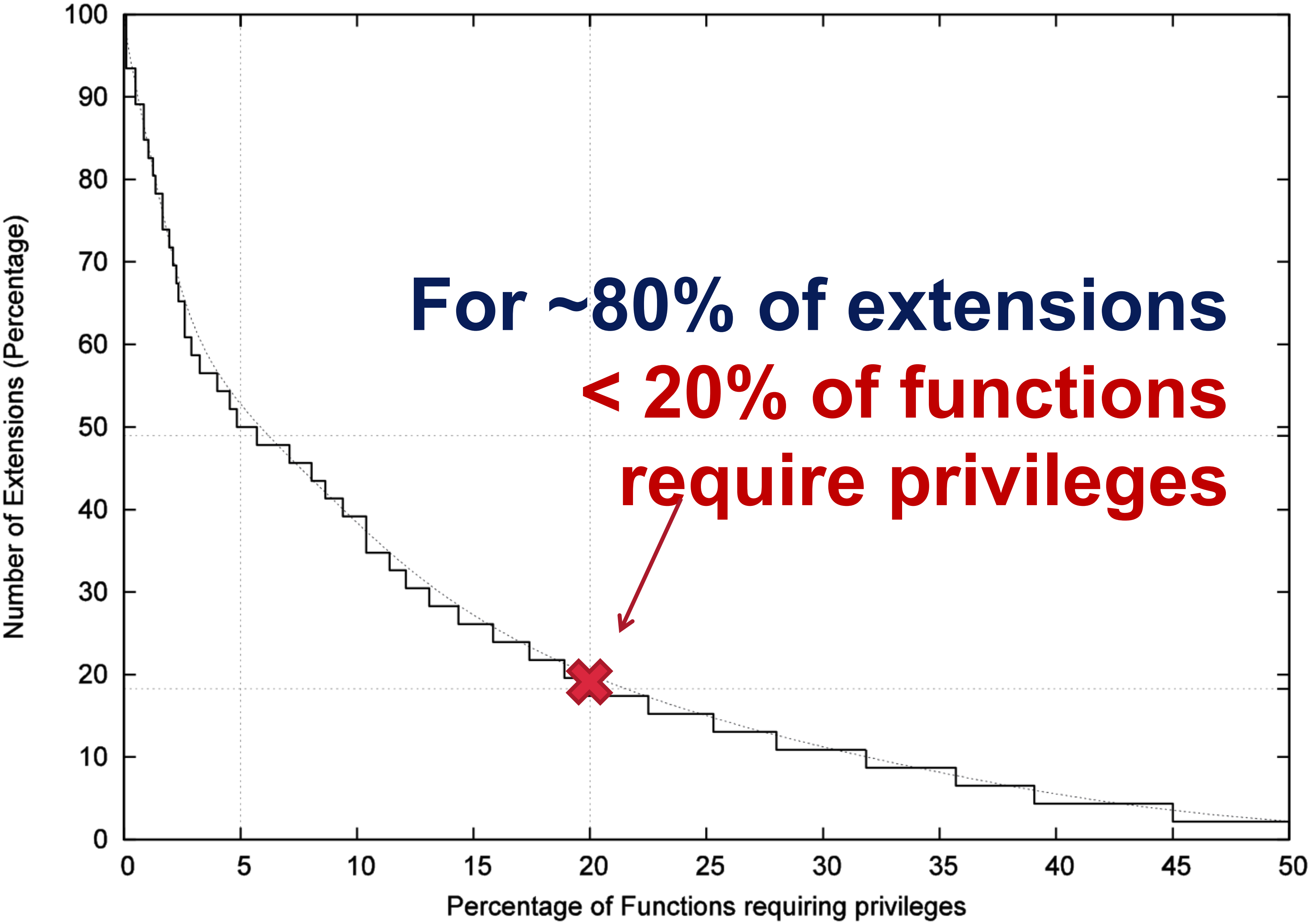
Not the exception

We measured the fraction of functions requiring privileges



For ~50% of extensions
< 5% of functions
require privileges

Data collected from the Top 50 Chrome Extensions



For ~80% of extensions < 20% of functions require privileges

Data collected from the Top 50 Chrome Extensions

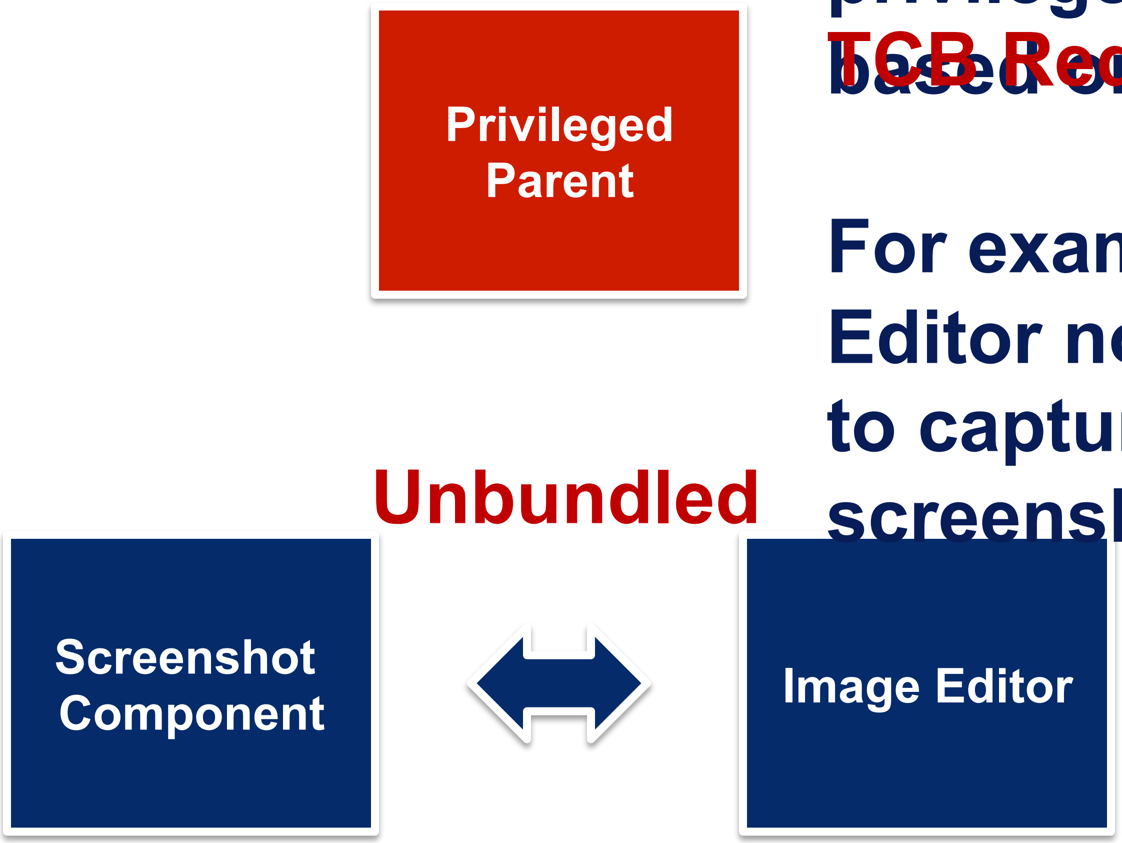
Our Solution:
privilege separation

Only the parent
runs privileged

Parent proxies
privileged calls
based on a policy.

TCB Reduction

For example, Image
Editor not allowed
to capture
screenshots.

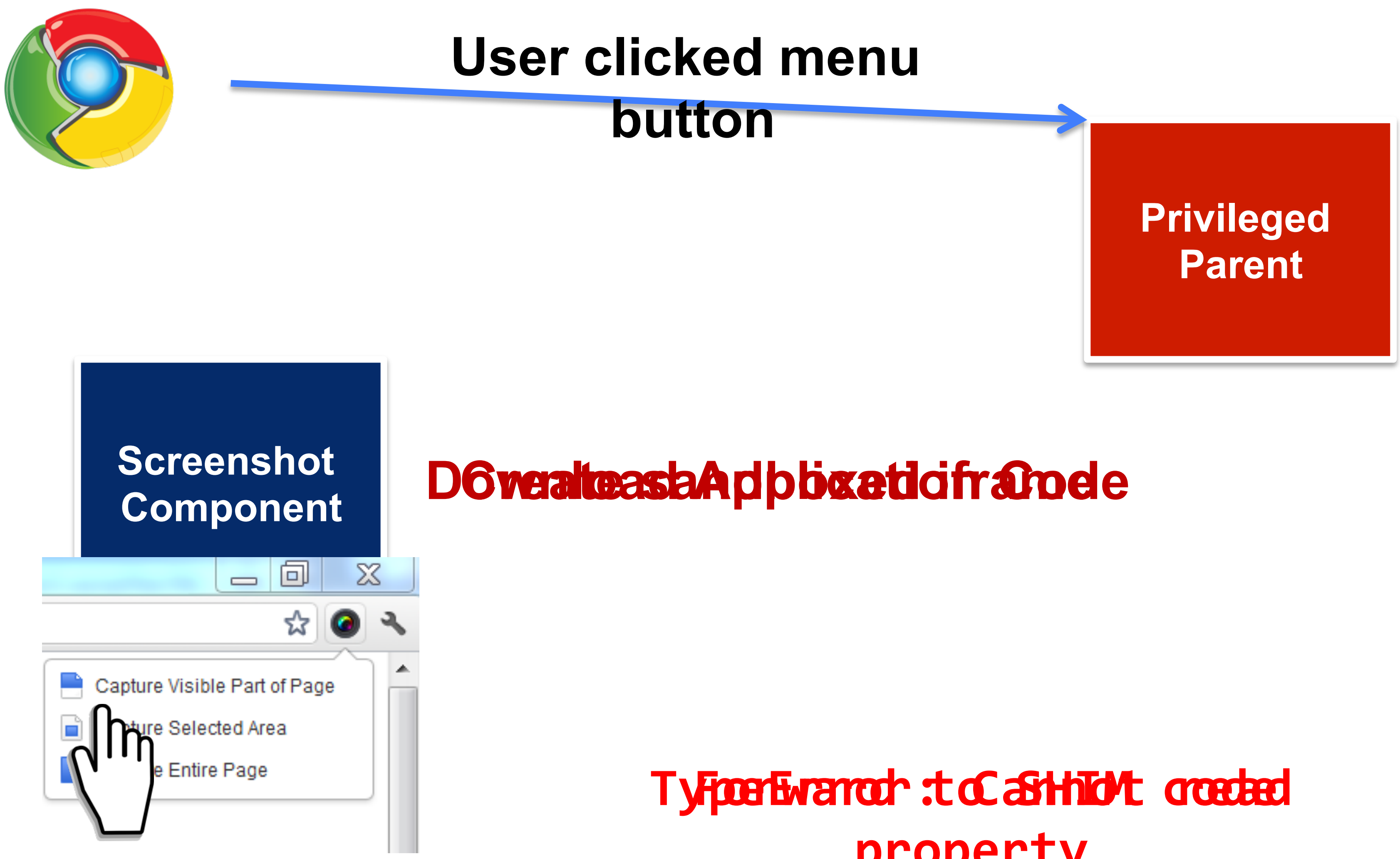


Privileged Parent

Unbundled

Screenshot Component

Image Editor



User clicked menu button

Privileged Parent

Screenshot Component

Download Application Code

Create a sandboxed iframe

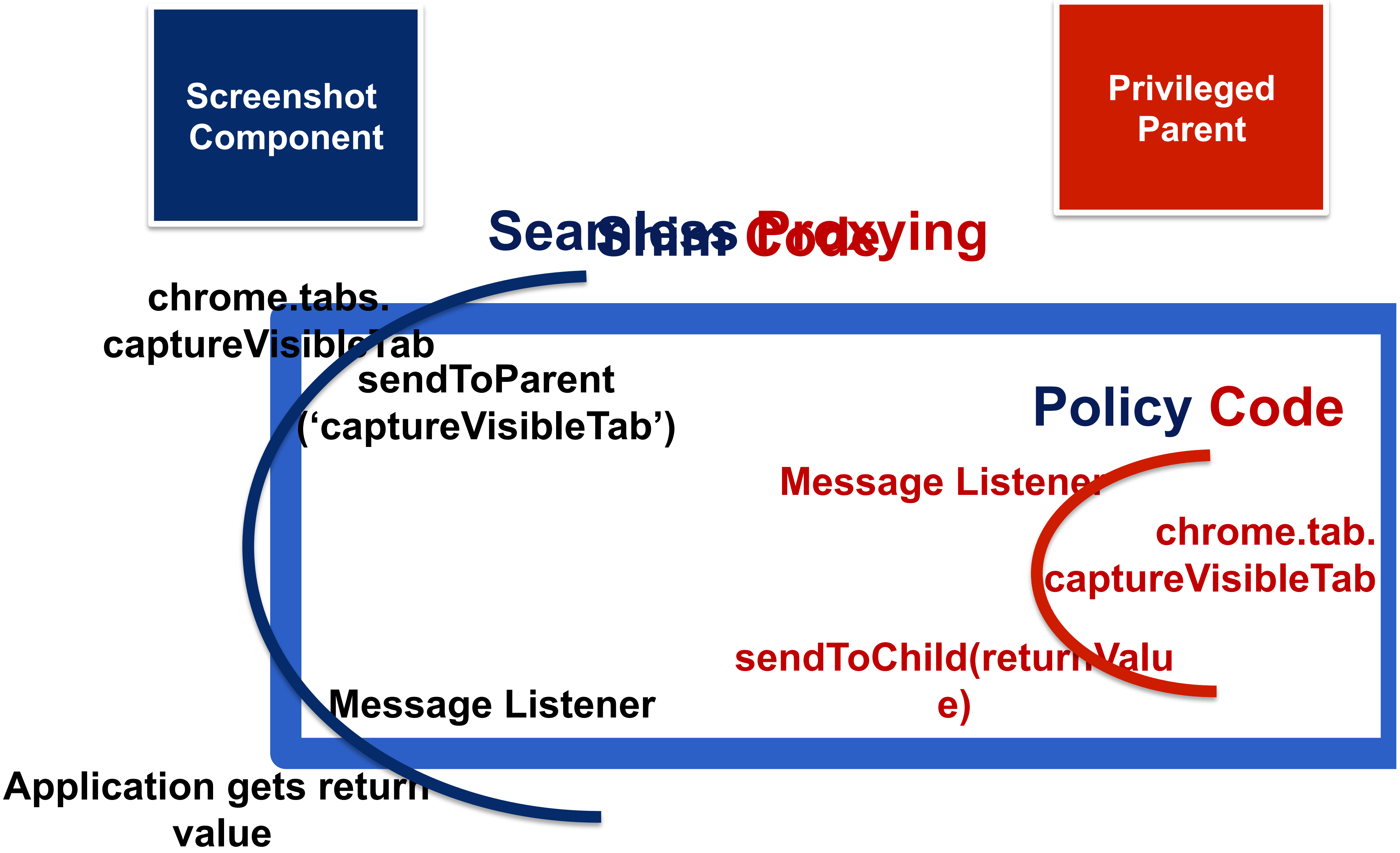
TypeError: Cannot read property 'captureVisibleTab' of undefined

Forward to SHIM code

call captureVisibleTab

Seamless Proxying

Shim Code



Screenshot Component

Privileged Parent

chrome.tabs.
captureVisibleTab

sendToParent
(‘captureVisibleTab’)

Policy Code

Message Listener

chrome.tab.
captureVisibleTab

sendToChild(returnValu
e)

Message Listener

Application gets return
value

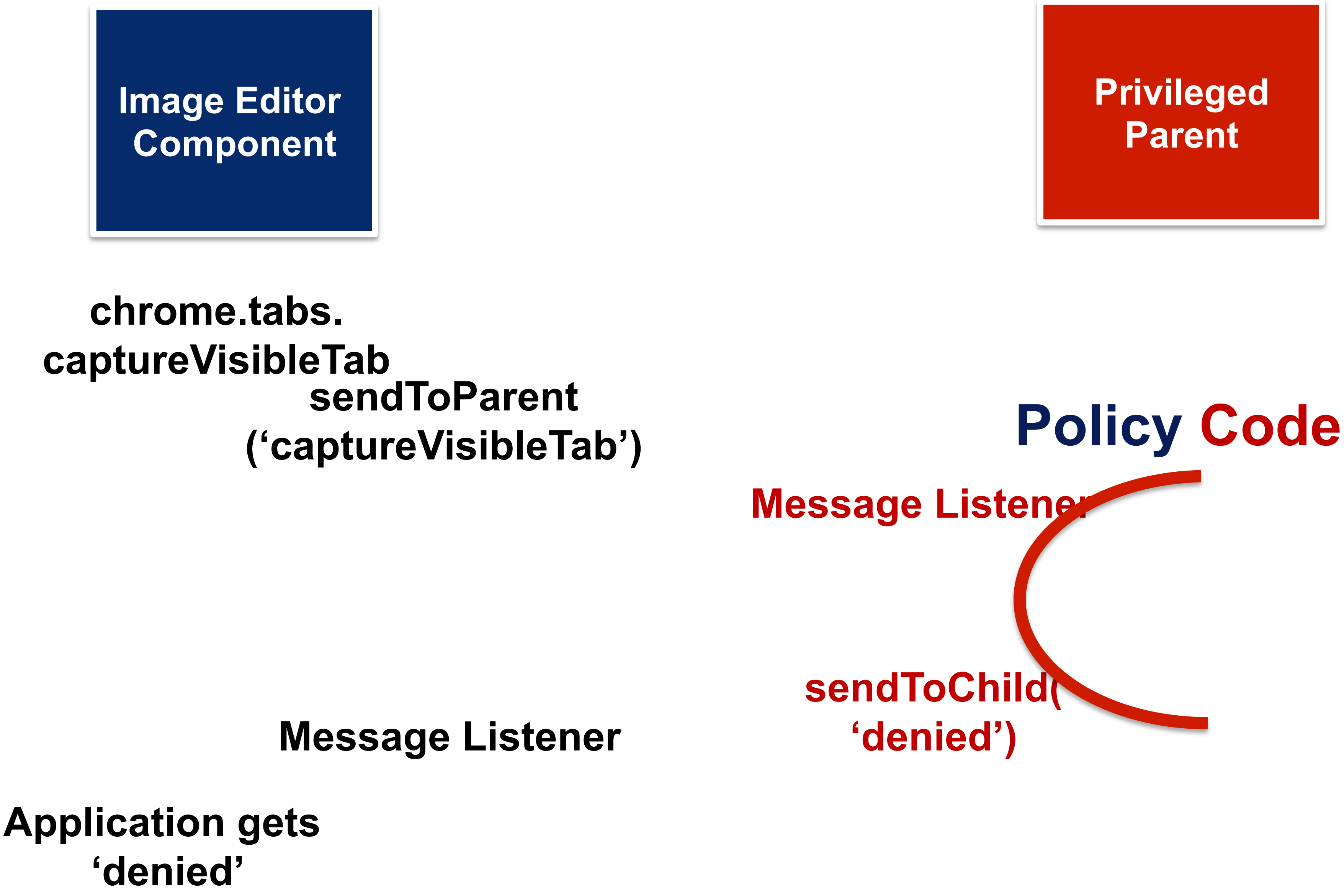


Image Editor Component

Privileged Parent

chrome.tabs.
captureVisibleTab

sendToParent
(‘captureVisibleTab’)

Policy Code

Message Listener

sendToChild(
‘denied’)

Message Listener

Application gets
‘denied’

parent invariants

the parent can’t convert
string to code

the parent can’t execute
arbitrary code from the web

the parent is the
only entry point into the
privileged origin

only primitive
data types cross the
privilege boundary

Application	Number of Users	Initial TCB (KB)	New TCB (KB)	Lines Changed
Awesome Screenshot	802,526	580	16.4	0
SourceKit	14,344	15,000	5.38	13
SQL Buddy	45,419	100	2.67	11

Privilege separation in HTML5 applications shows how applications can cheaply create arbitrary number of components.

Our approach utilizes standardized abstractions already implemented in modern browsers.

We retrofit applications to demonstrate TCB reductions.

New Security Primitives

Proactive Approaches

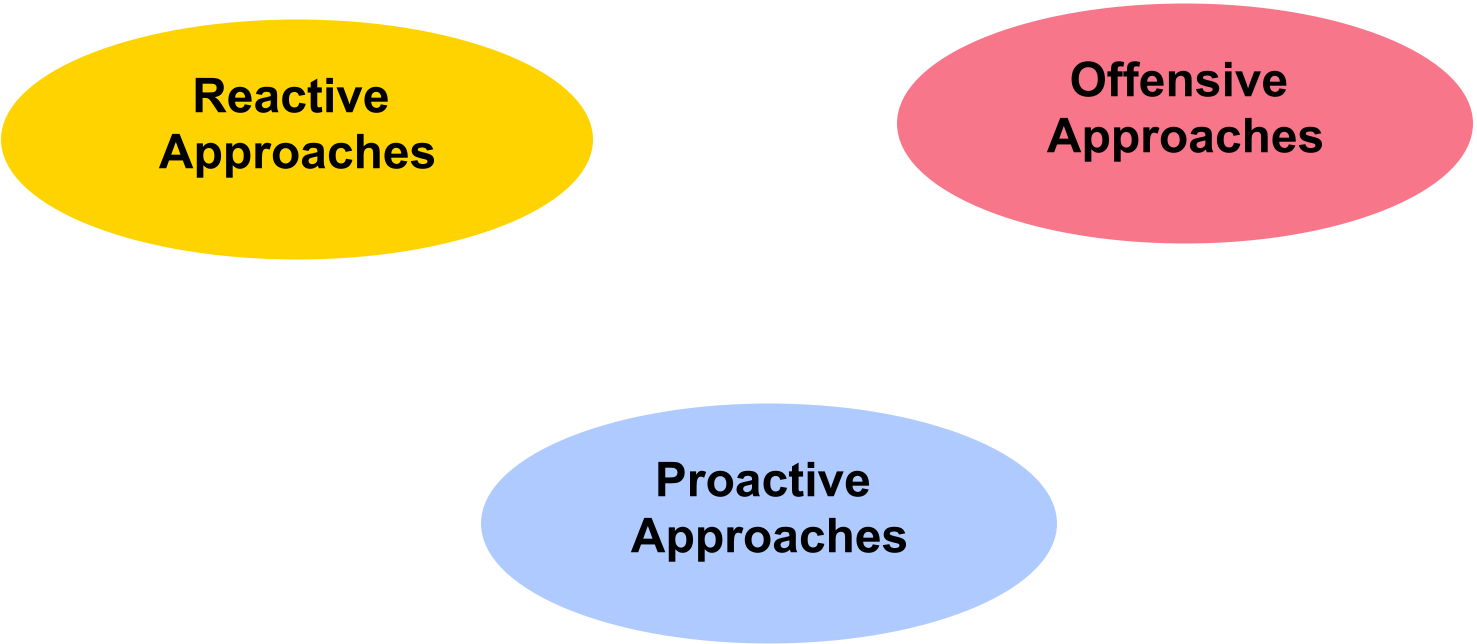
- **For building secure systems even when the machine may be compromised**
  - *Cloud Terminal [USENIX Annual Technical Conf’12]*

- **For building secure applications by design**
  - *Context-sensitive auto-sanitization in web templating languages using type qualifiers [CCS’11]*

- **For better security architecture & auditability**
  - *Privilege separation in HTML5 [USENIX Security’12]*

Conclusion

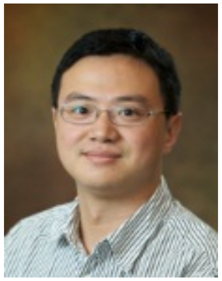
Malware enters new landscape as more parts of the world get connected



Reactive Approaches

Offensive Approaches

Proactive Approaches



http://bitblaze.cs.berkeley.edu

http://webblaze.cs.berkeley.edu

dawnsong@cs.berkeley.edu