



CYBER SECURITY: ESSENTIALS

Daniel Medina — medina@nyu.edu

Bill Dorney — wpd1@nyu.edu



RECAP

EXTERNAL-FACING ASSETS

Mitre Taxonomy and Trends

OWASP TOP 10 Web
Vulnerabilities

EXTERNAL-FACING ASSETS

PCI Requirement 6.6:

WAFs or Code Reviews

TARGETS CYCLE IN POPULARITY

Unix Services (sendmail, fingerd, telnet)

Windows Services (IIS, DCOM, RPC)

Websites (XSS, SQL injection)

Browsers (CSRF, plugins)

Apps (Java, Adobe)

Users (just about anything works)

USERS

Easy to blame

Do users have the necessary
inputs to make well-informed
security decisions?

USERS

Launch a phishing campaign
against your own employees to
train them to be more wary?

<https://duo.com/resources/duo-insight>



[Find Us](#) | [FAQs](#) | [Contact Us](#) | [Log On](#)

Welcome to Chase.com

User Logon

User ID

Password

Log On

© 2013 JPMorgan Chase & Co.

[Home](#) | [Privacy and Security](#) | [Log On](#)



[Find Us](#) | [FAQs](#) | [Contact Us](#) | [Log On](#)

Welcome to Chase.com

User Logon

User ID

Password

Log On

[Home](#) | [Privacy Notice](#)

PHISHING PAGE

MORE TECHNICAL ATTACKS

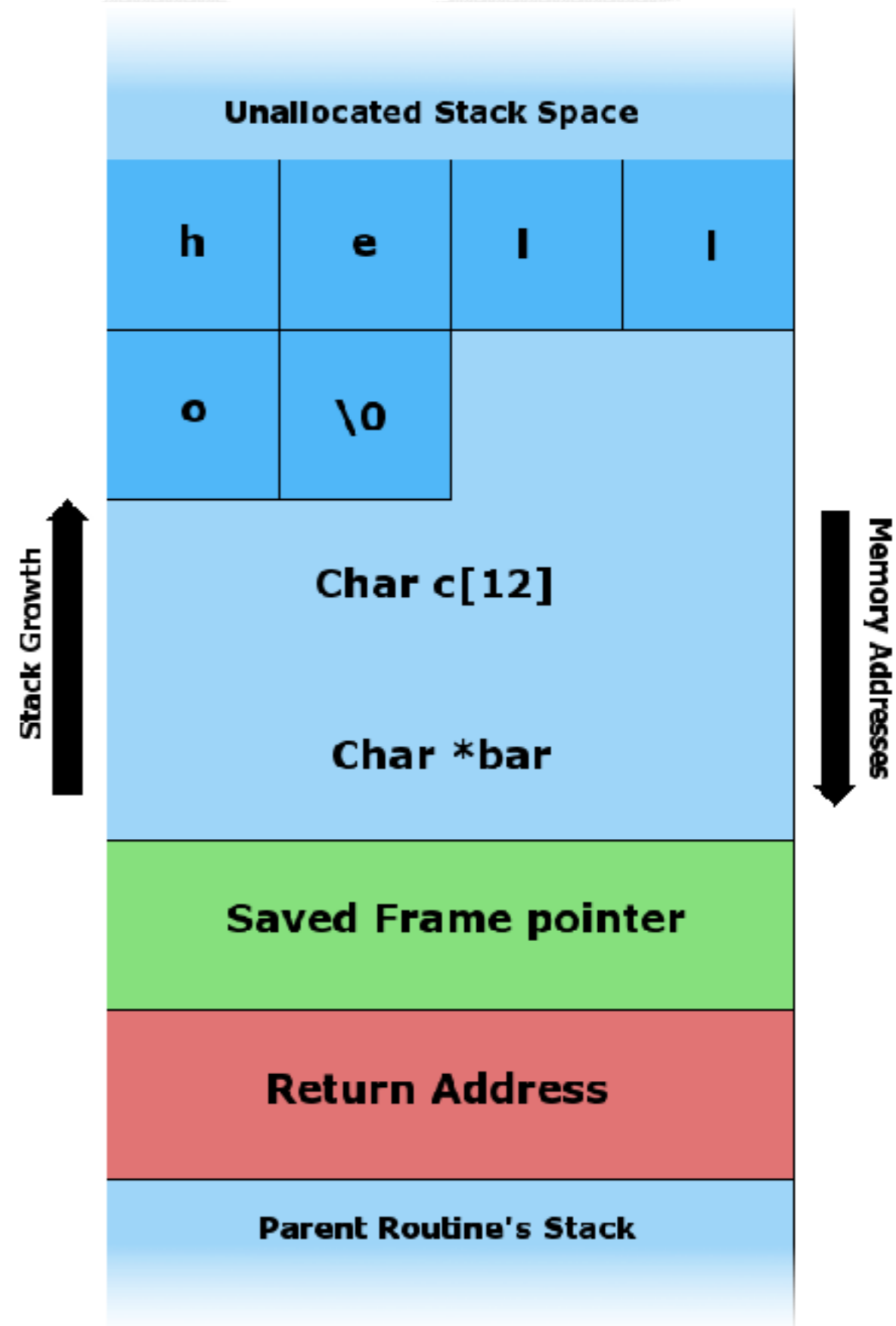
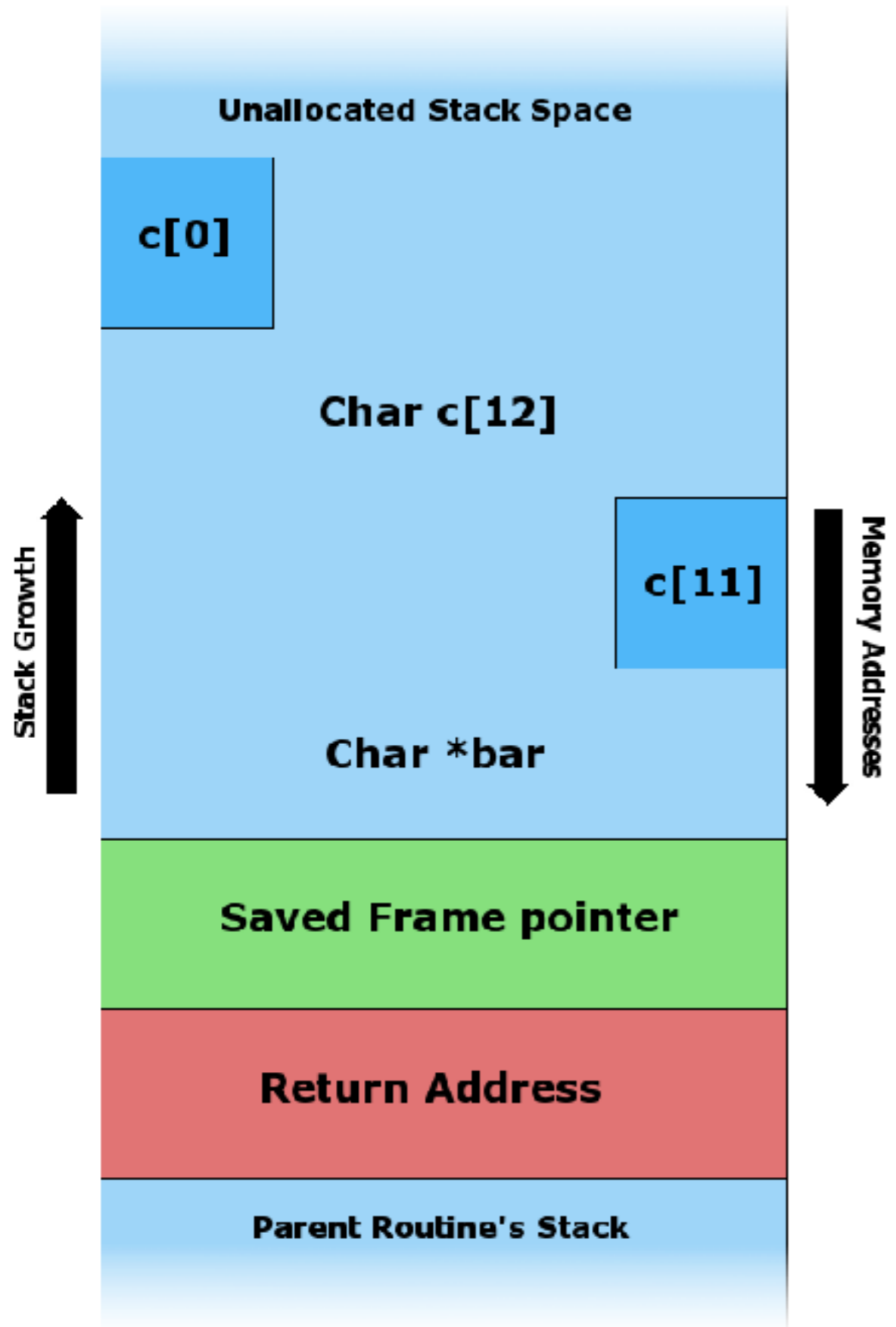
Remote execution

Think “inputs” and “context”

OVERFLOW ATTACKS

**We received unexpected input
from an untrusted source**

**We executed input from an
untrusted source**

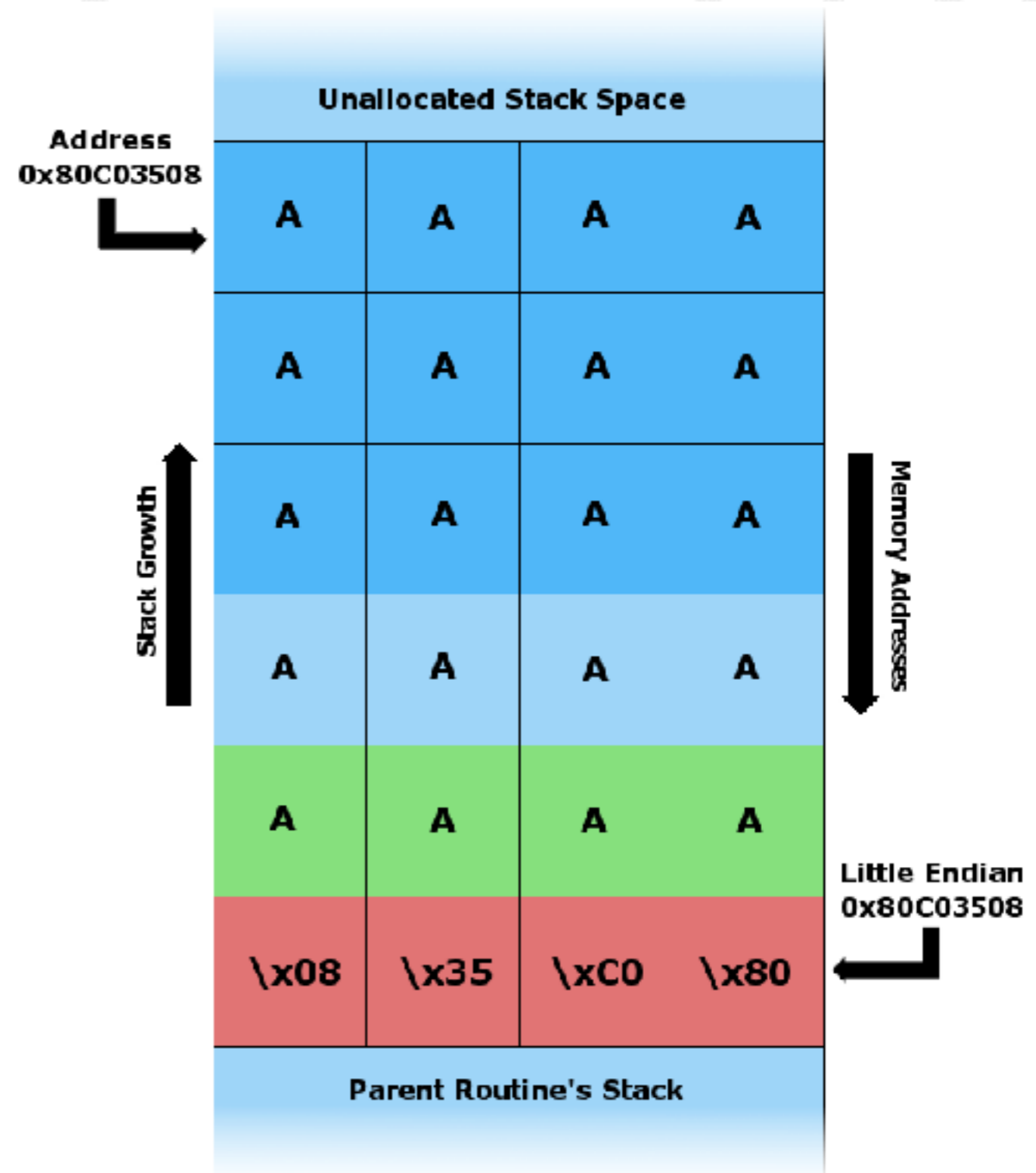


OVERFLOW ATTACKS

Stack buffer overflow

Smashing The Stack For Fun And Profit

As long as the person at the other end of the Internet did only exactly what we expect, we're fine!



OVERFLOW DEFENSES

Sandboxing

Canary Words

Non-Executable Memory

Randomized Memory (ASLR)

SQL INJECTION

We received unexpected input
from an untrusted source

We executed input from an
untrusted source

Little Bobby Tables

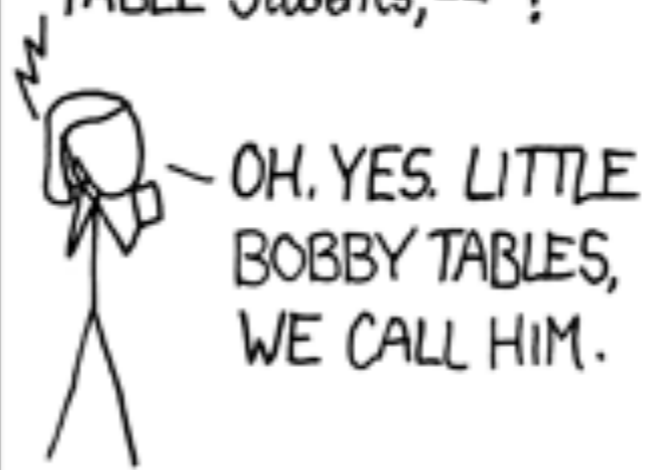
HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.



OH, DEAR - DID HE BREAK SOMETHING?
IN A WAY-



DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?



OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.

WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.



AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

SQL INJECTION

Examples

```
select * from userdata where  
user='$user' and  
password='$password'
```

```
select * from userdata where  
user='medina' and  
password='mypass'
```

Mass Attacks
have used this

```
select * from userdata where  
user='medina' and  
password=''; select * from  
userdata where 'x'='x'
```


SQL INJECTION DEFENSES

Bad Ideas:

“blacklist”, e.g., no apostrophes or keywords

Shaquille O’Neal, **And**rew Carnegie

Good Ideas:

Placeholders / Proper Encoding

CROSS-SITE SCRIPTING (XSS)

**We received unexpected input
from an untrusted source**

**We executed input from an
untrusted source**

< XSS in Web-based admin UI #2018

 **Closed**

medina opened this issue a month ago · 7 comments



medina commented a month ago

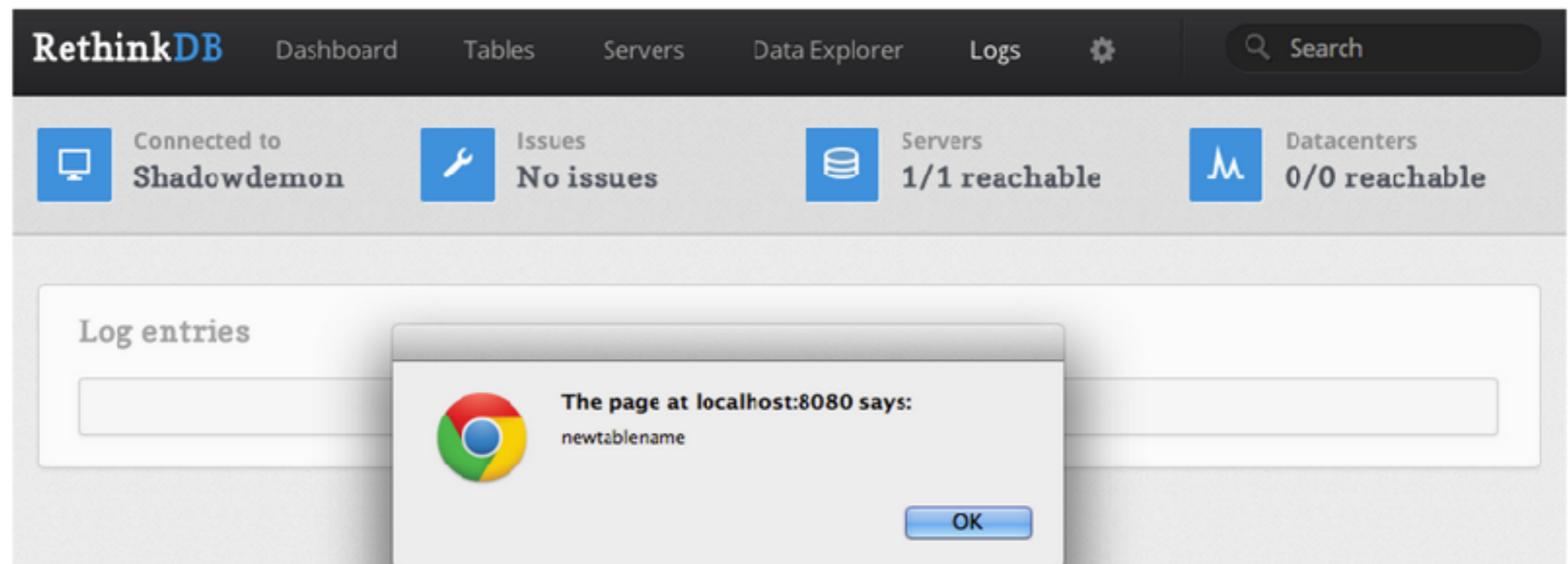
Attempting to rename a table `<script>alert('newtablename');</script>` gives

```
Sorry, something went wrong with the ajax request. Please try again.  
Error 400: invalid name: <script>alert('newtablename');</script>
```

and generates a log entry of:

```
info: HTTP request throw a schema_mismatch_exc_t with what = invalid name: <script>alert('newtablename
```

which looks like:



The screenshot shows the RethinkDB admin interface. At the top, there is a navigation bar with the RethinkDB logo and menu items: Dashboard, Tables, Servers, Data Explorer, Logs, and a search bar. Below the navigation bar, there are four status cards: 'Connected to Shadowdemon', 'Issues: No issues', 'Servers: 1/1 reachable', and 'Datacenters: 0/0 reachable'. The main content area is titled 'Log entries' and contains a text input field. A browser alert dialog is overlaid on top of the log entries, displaying the Chrome logo and the message: 'The page at localhost:8080 says: newtablename'. An 'OK' button is visible at the bottom of the alert dialog.

XSS DEFENSES

Input or Output Encoding

Server-side hint to browser: CSP

(Could this vuln be made extinct?!)

CROSS-SITE REQUEST FORGERY

aka CSRF (“sea-surf”)

A bit more subtle than the rest

CROSS-SITE REQUEST FORGERY

`https://bank.example.com/do?`

`action=transfer;`

`dstacct=123456789;`

`amount=1000`

(How is this authenticated?)

CROSS-SITE REQUEST FORGERY

site: evil.com serves victim:

```

```

CROSS-SITE REQUEST FORGERY

Another vulnerability that could
be made extinct?

Same-Site Cookies

OTHER STUFF

Bad Randomness

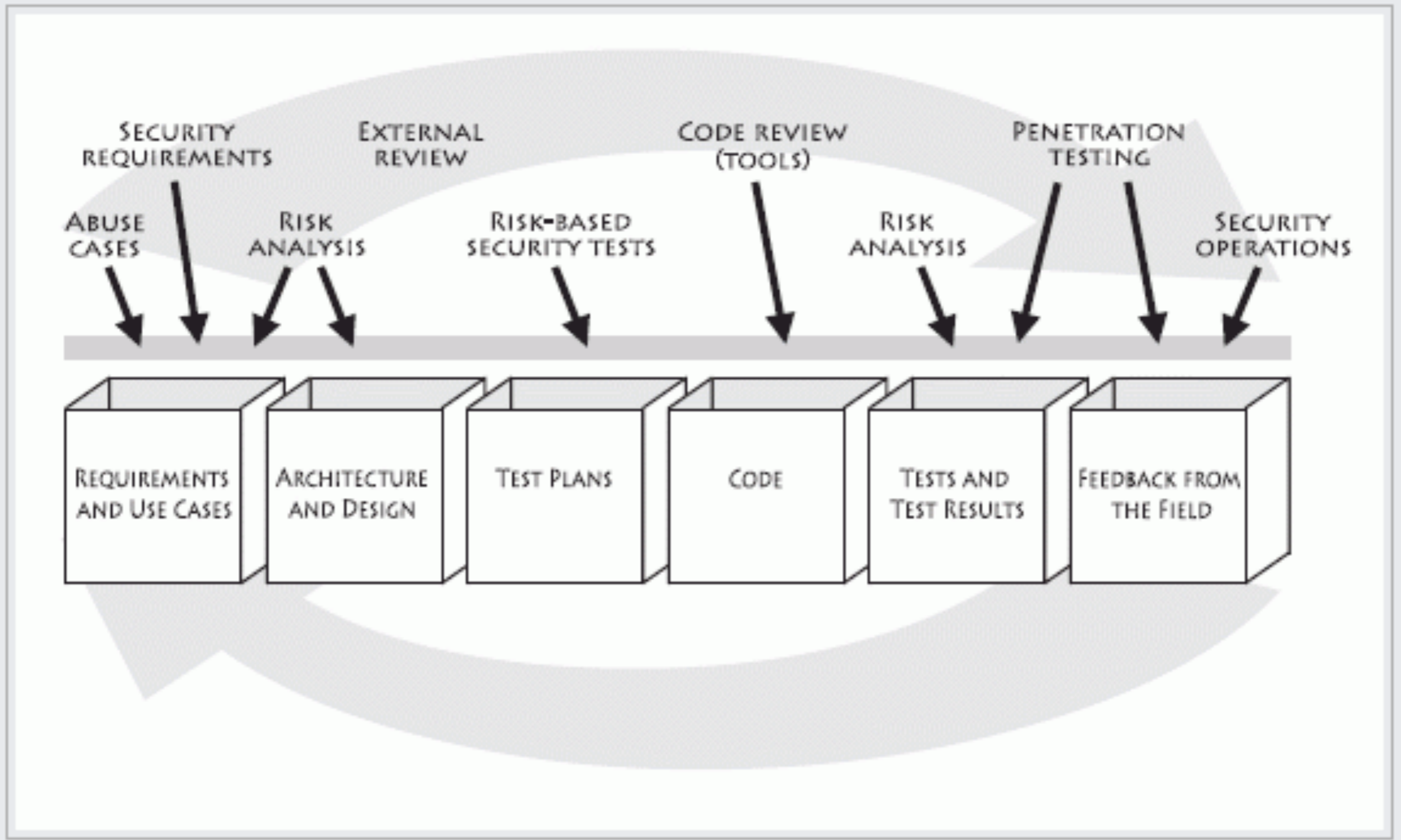
No or Weak Authentication

Bad Configuration

ASSESSING SECURITY



SECURE SDLC



PENETRATION TESTING

juliocesarfort/public-pentesting-reports

[SpiderOak Sample](#)

[TrueCrypt Audit by iSec Partners](#)

FUZZING

afl: “a novel type of compile-time instrumentation and genetic algorithms...”

Permute inputs, find unexpected outcomes (like crashes) that might be exploitable

BOUNTY PROGRAMS

Not always smooth to run a program

Yahoo Bug Bounty (shirts instead of \$)

Prezi Bug Bounty (hack of off-limits internal site)

Digital Ocean (public debate about vuln)

BOUNTY PROGRAMS

Managed programs:

HackerOne: hackerone.com

BugCrowd: bugcrowd.com

Crowdcurity / Cobalt: cobalt.io

Individual companies:

<https://bugcrowd.com/list-of-bug-bounty-programs>

FIXING SECURITY (PATCHES)

Patch Tuesday

Full Disclosure (forum)

Patching in public (Postgres)

