



Revisiting SQL Injection

Will we ever get it right?

Michael Sutton, Security Evangelist



Overview

Background

- What it is?
- How are we doing?

Web 2.0

- SQL injection meets AJAX

Fuggle

- SQL Injection meets Google meets fuzzing

Conclusion





Attack Scenario #1

Good 'ol SQL Injection



SQL Injection



Unfiltered User Input

- Attacker can influence or rewrite a backend SQL query

Client Request

POST /SPIWare/Login.aspx HTTP/1.1

Host: localhost

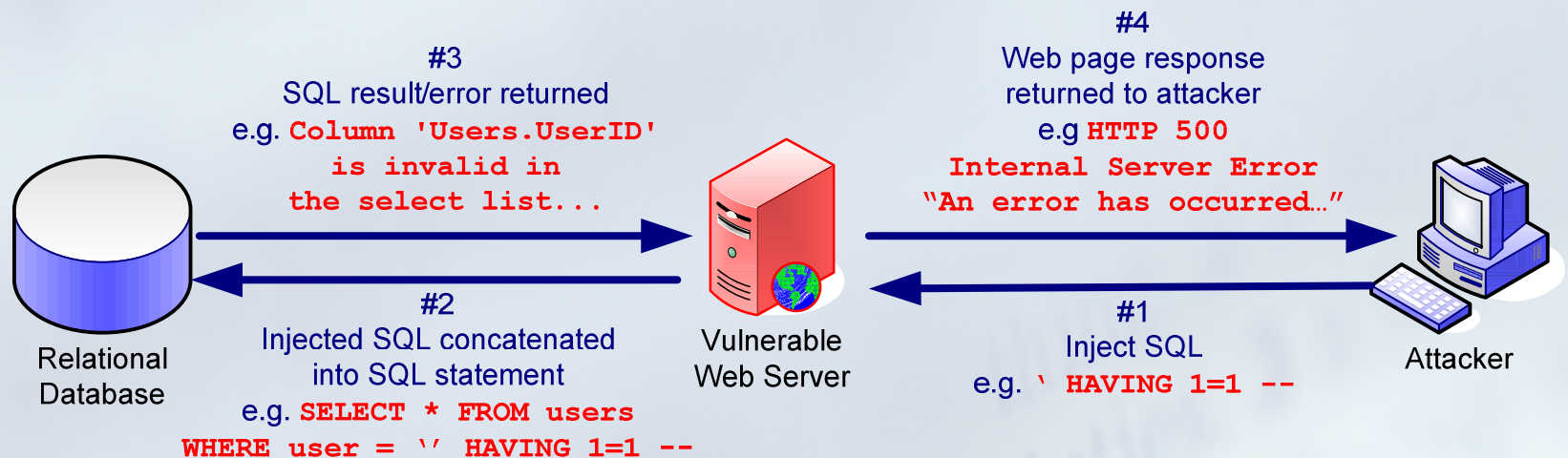
txtEmail=msutton@spidynamics.com&**txtPassword**=password

SQL Query

SELECT * FROM Customers WHERE Email = '{0}' and
Password = '{1}', **txtEmail.Text**, **txtPassword.Text**



SQL Injection



What is SQL Injection?



Verbose SQL Injection

- Server provides detailed error messages
- Error messages are leveraged by attacker to gain insight into database structure

Blind SQL Injection

- Database error messages are suppressed
- Attacker must alter approach and only make queries with Boolean outcomes (yes/no questions)



Verbose vs. Blind SQL Injection

Cause



Definition



Impact



Demo

Server Error in '/Hackware' Application.

*Column '**Customers.CustomerID**' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.*

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

VS

Whoops...an error has occurred!



What damage can be done?



Confidentiality

- SELECT
- Read from application and system tables

Integrity

- DELETE, INSERT, DROP
- Alter existing data or inject new data

Authentication Bypass

- E.g. ' OR 1=1 --

System Compromise

- Stored procedures
- Extended stored procedures
- Excellent paper - http://www.nextgenss.com/papers/advanced_sql_injection.pdf



SQL Injection Demo

Cause



Definition



Impact



Demo

SPI WARE

Sign In

Home
Contact Us

Sign In
Create New Account

Tracing

If you don't have an account, click [here](#) to create one.

Click [here](#) if you've forgotten your password.

Enter your Information

Email

Address:

Password:

☐ Sign me in automatically. (Do NOT use this feature if you are using a public computer or if you share your computer with other users. Please note that this feature is local to the current browser and computer that you are using now.)

Sign-In



Prevention

Input Validation

- Always ensure that you receive what you expect
- White list vs. black list
- Validate type, length, format, and range
 - Regular expressions - <http://regexlib.com>
- Escape special characters
- Always validate on the server side!

Leverage Frameworks

- Parameterized Queries
- 'Point and Click' query generation

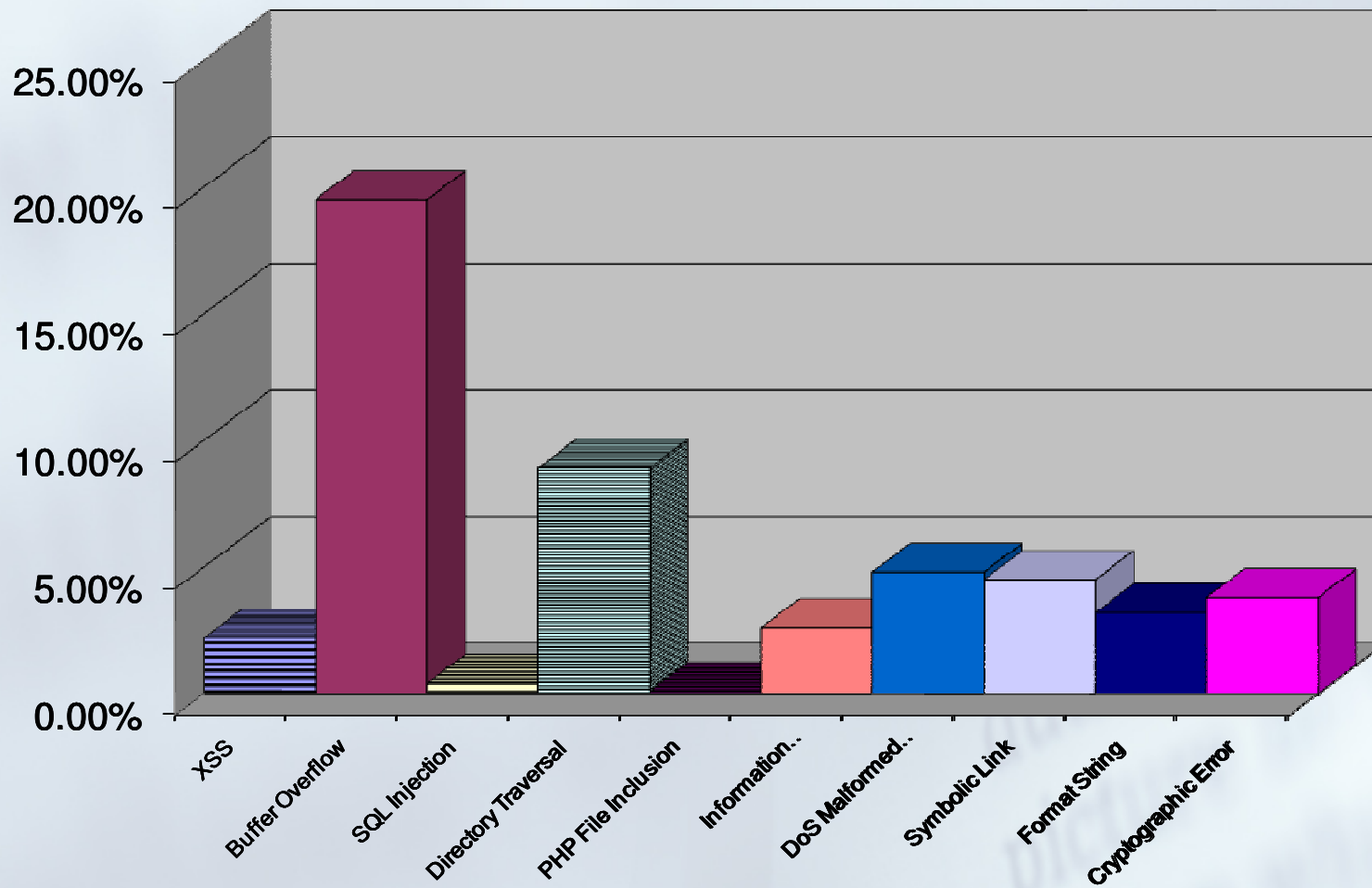
ACLs

- Execute queries with least privilege



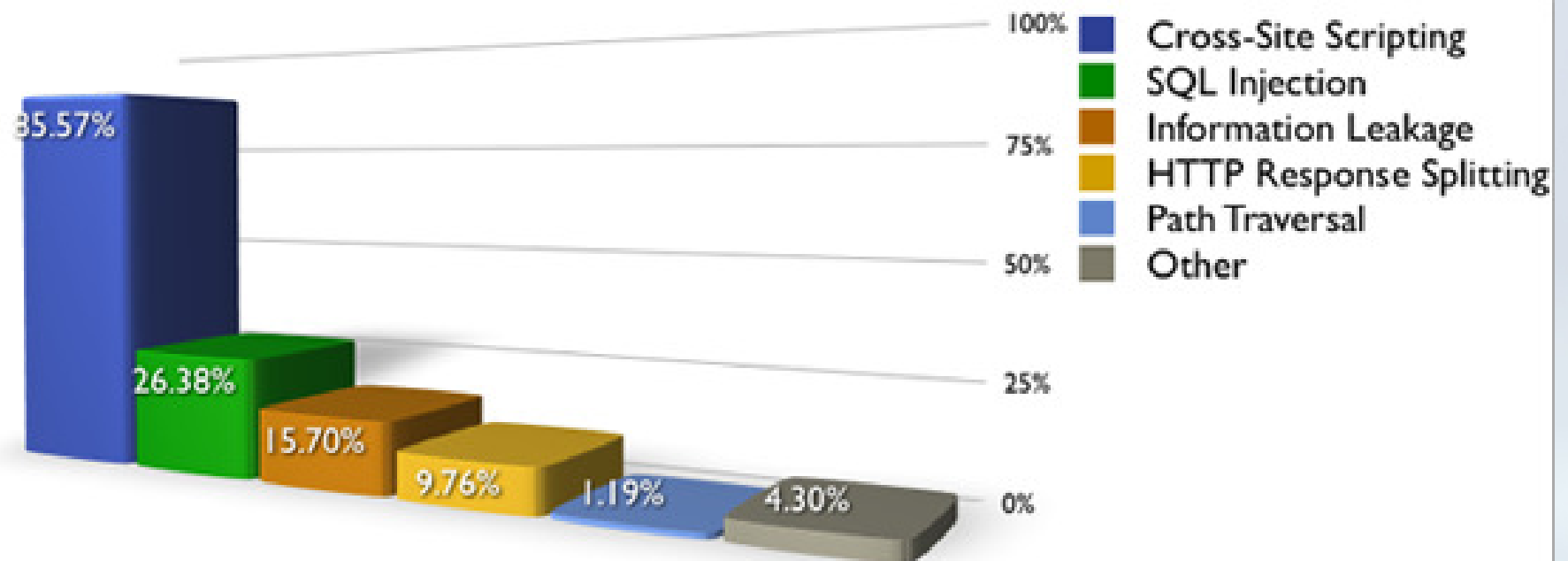
Mitre CVE Statistics

2001

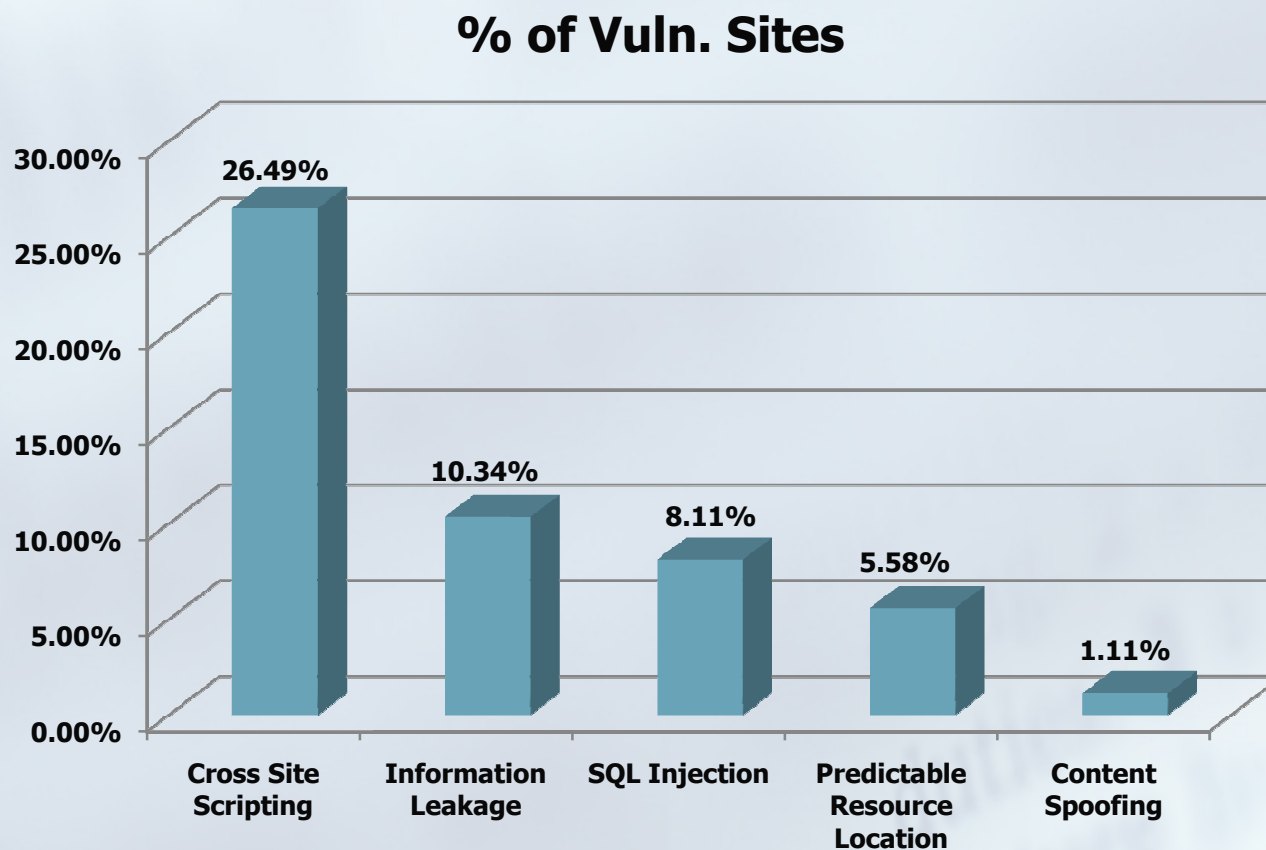


WASC Statistics - 2006

Percentage of websites vulnerable by class (Top 5)



WASC Statistics – 2007 (1H)





Attack Scenario #2

SQL Injection via AJAX



What is Web 2.0?

Tim O'Reilly

- *Web 2.0 is the business revolution in the computer industry caused by the move to the internet as platform, and an attempt to understand the rules for success on that new platform.*

Wikipedia

- *Web 2.0...refers to a perceived second-generation of Web based communities and hosted services — such as social networking sites, wikis and folksonomies — that facilitate collaboration and sharing between users.*



Web 2.0 My Definition



Web 1.0

- Incomplete pages were shameful
- "Please come back later when we're ready"



Web 2.0

- Incomplete pages are a feature!
- "Stick around and help us improve the site"

Same Vulnerabilities



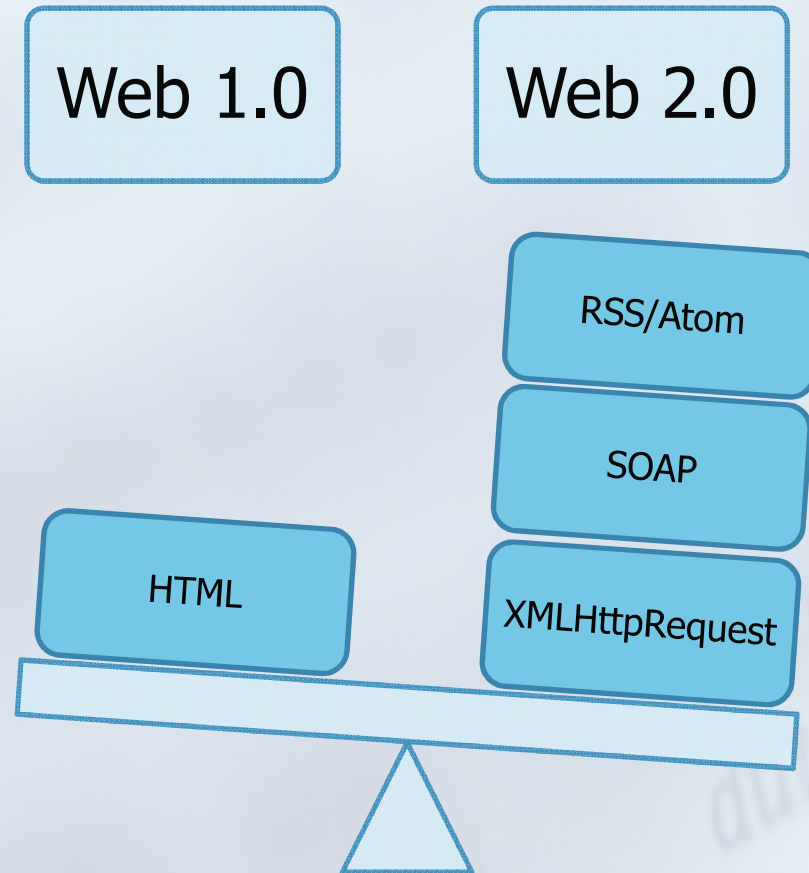
Additional Input Vectors



More Complexity



Input Vectors



AJAX



What does AJAX change?

- Business logic is dispersed among multiple client side files/functions
- Requests are made in the background without user intervention but are just as susceptible to attack

What doesn't AJAX change?

- AJAX does not create new vulnerabilities

How does AJAX affect security?

- Increased surface area
 - More business logic is exposed
 - New input vectors are exposed
- Security tools must understand the XHR objects and their syntax in order to identify input vectors



What is AJAX?



Asynchronous

- Requests are initiated in the background

JavaScript

- JavaScript instantiates the XMLHttpRequest object and generates the requests

and XML

- This is a misnomer as AJAX frameworks commonly employ alternate data interchange formats
 - JSON - Atlas
 - Serialized Java - Google Web Toolkit
 - HTML
 - XML



What is AJAX?



Multiple frameworks

- Prototype (<http://www.prototypejs.org/>)
- Script.aculo.us
- Dojo (<http://dojotoolkit.org/>)
- ASP.Net AJAX (<http://ajax.asp.net/>)
- Etc.

Multiple browser objects

- Internet Explorer
 - IE6 - XMLHttpRequest ActiveX control
 - IE7 – XMLHttpRequest native script object
- Firefox
 - XMLHttpRequest object



Classic Web Apps

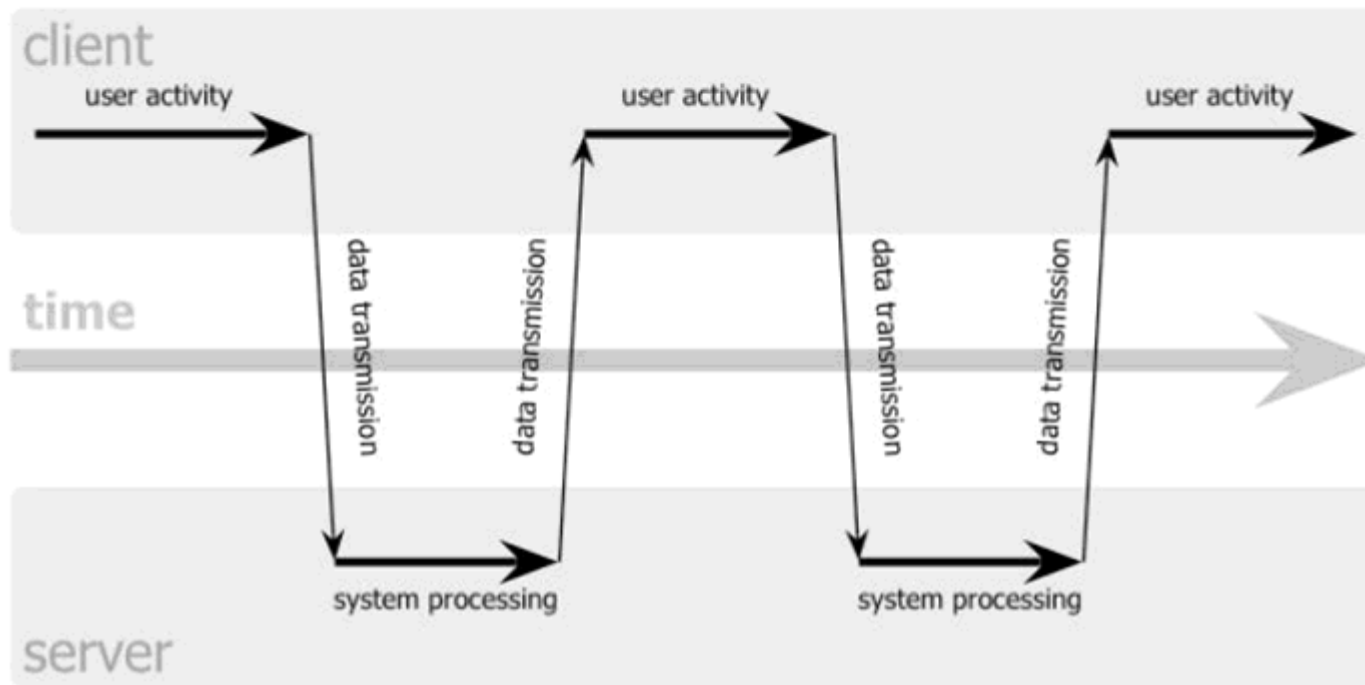
Cause

Definition

Impact

Demo

classic web application model (synchronous)



AJAX Web Apps

Cause



Definition

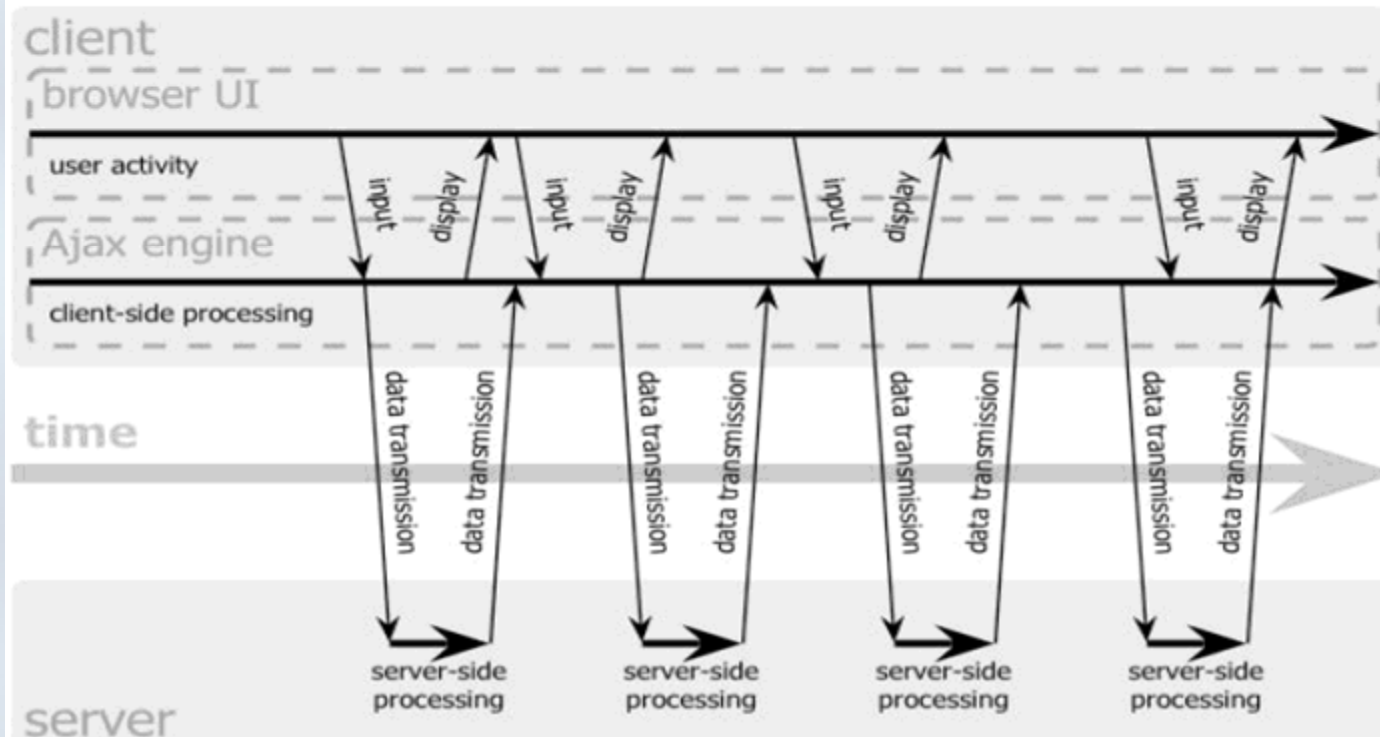


Impact



Demo

Ajax web application model (asynchronous)



Google Maps

3950 Las Vegas Blvd. South Las Vegas, NV 89119 - Google Maps - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://maps.google.com/

Web Images Video News Maps Gmail more

michaelawsutton@gmail.com | Saved Locations | Help | Web History | My Account | Sign out

Google Maps

3950 Las Vegas Blvd. South Las Vegas, NV 89119

Search Maps

Search the map Find businesses Get directions

Search Results My Maps

3950 Las Vegas Blvd S
Las Vegas, NV 89119

Print Email Link to this page

Street View New! Traffic Map Satellite Hybrid

Address:
3950 Las Vegas Blvd S
Las Vegas, NV 89119

Make this my default location
Get directions: To here - From here
Search nearby - Save to My Maps

W Ali Baba Ln
W Hacienda Ave
Connector Rd
Dean Martin Dr
Frank Sinatra Dr
Mandalay Bay Rd
Mandalay Bay Resort and Casino
E Hacienda Ave
Dunsmuir Ln
Bainbridge Ln
Haven St
E Dewey Dr
4 Seasons Dr
S Las Vegas Blvd
Las Vegas Strip
604
693
7457

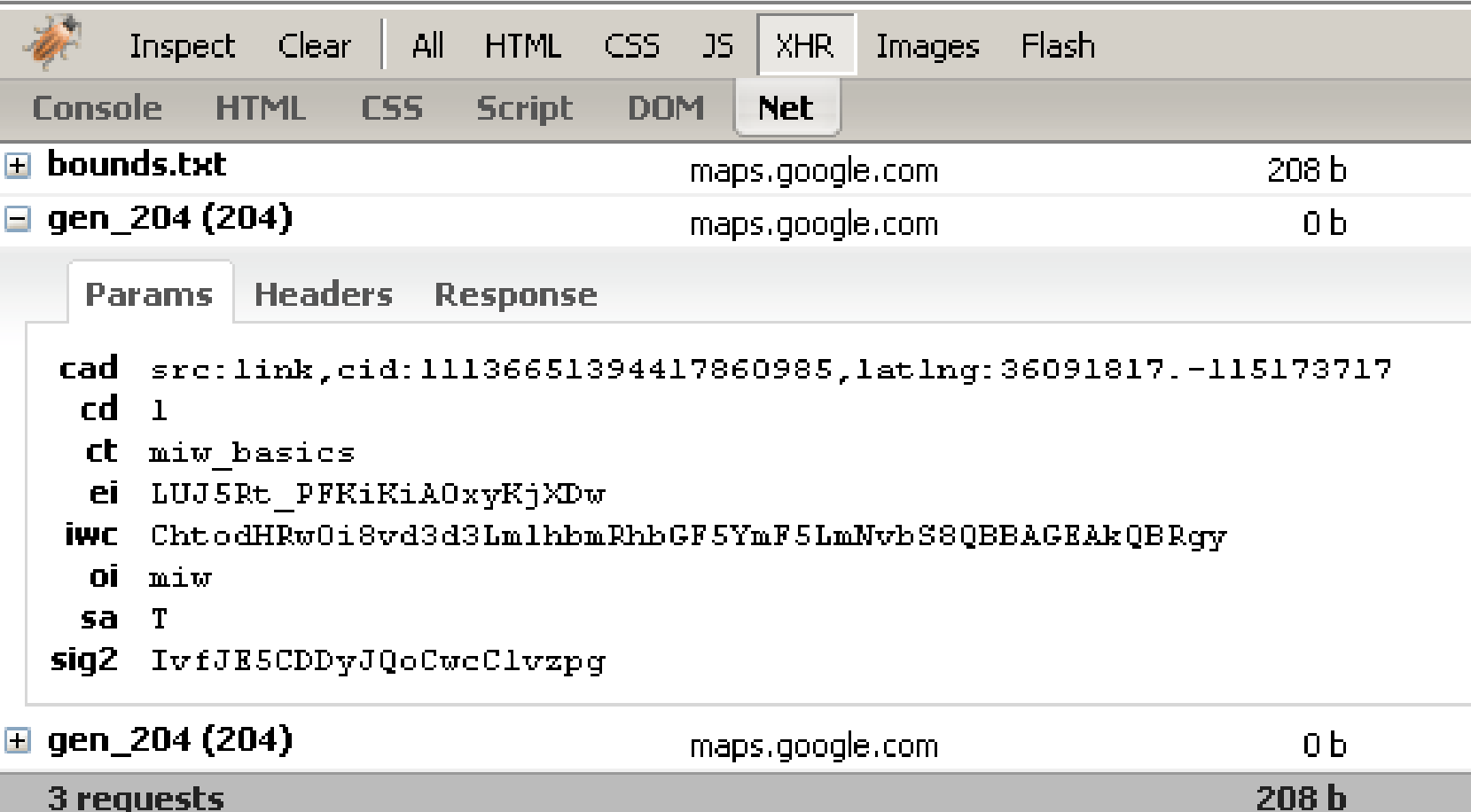
1000 ft
1000 m

Done

1 Error



FireBug



The image shows the FireBug interface with the XHR tab selected. The top bar includes a bug icon, 'Inspect', 'Clear', and tabs for 'All', 'HTML', 'CSS', 'JS', 'XHR', 'Images', and 'Flash'. Below this, the 'Console' tab is active, showing a list of requests. The first request is 'bounds.txt' from 'maps.google.com' with a size of '208 b'. The second request is 'gen_204 (204)' from 'maps.google.com' with a size of '0 b'. The 'gen_204 (204)' request is expanded, showing its 'Params' tab. The parameters are: 'cad' (src:link,cid:11136651394417860985,latlng:36091817.-115173717), 'cd' (1), 'ct' (miw_basics), 'ei' (LUJ5Rt_PFKiKiA0xyKjXDw), 'iwc' (ChtodHRw0i8vd3d3LmlhbmRhbGF5YmF5LmNvbS8QBBAGEAkQBRgy), 'oi' (miw), 'sa' (T), and 'sig2' (IvfJESCDdyJQoCwcClvzpg). Below the expanded request, another 'gen_204 (204)' request is listed. At the bottom, a summary bar shows '3 requests' and '208 b'.

| Request | Source | Size |
|----------------------|-----------------|-------|
| bounds.txt | maps.google.com | 208 b |
| gen_204 (204) | maps.google.com | 0 b |

Params Headers Response

cad src:link,cid:11136651394417860985,latlng:36091817.-115173717
cd 1
ct miw_basics
ei LUJ5Rt_PFKiKiA0xyKjXDw
iwc ChtodHRw0i8vd3d3LmlhbmRhbGF5YmF5LmNvbS8QBBAGEAkQBRgy
oi miw
sa T
sig2 IvfJESCDdyJQoCwcClvzpg

| Request | Source | Size |
|----------------------|-----------------|------|
| gen_204 (204) | maps.google.com | 0 b |

3 requests 208 b



What damage can be done?



Same old story

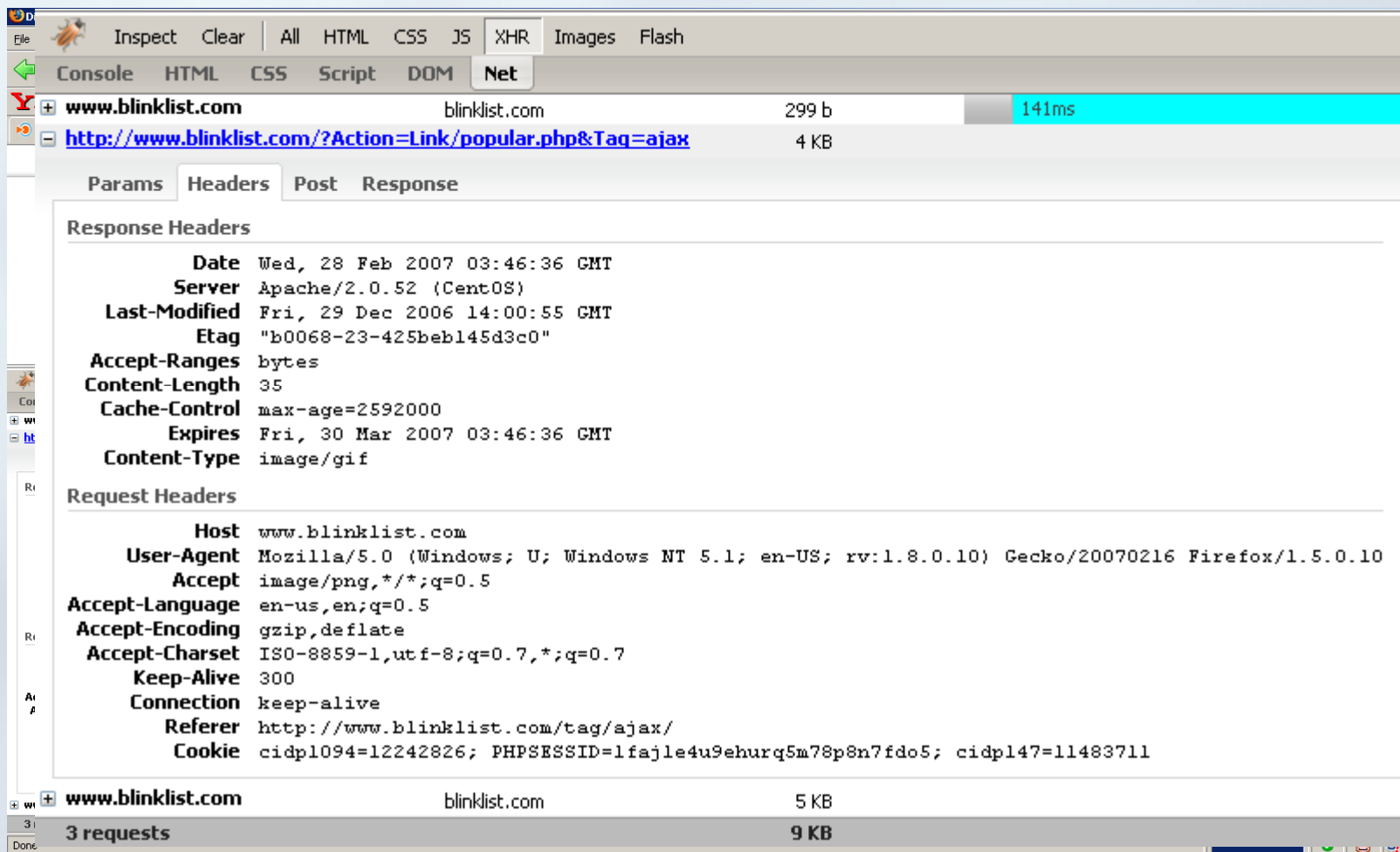
- Any classic web application vulnerabilities can be exploited through AJAX
 - SQL injection
 - XSS
 - Etc.

Risk

- In a rush to implement the latest Web 2.0 technologies, companies are ignoring security



How Not to Implement AJAX - BlinkList



How Not to Implement AJAX - BlinkList



How Not to Implement AJAX - BlinkList

Verbose SQL errors

- Multiple

XSS

- Ability to inject client side script

Exposed functionality

- Web based email

Directory browsing

- Access to restricted files



AJAX Demo



SPI WARE

Sign In

[Home](#)
[Contact Us](#)

[Sign In](#)
[Create New Account](#)

[Tracing](#)

If you don't have an account, click [here](#) to create one.

Click [here](#) if you've forgotten your password.

Enter your Information

Email

Address:

Password:

☐ Sign me in automatically. (Do NOT use this feature if you are using a public computer or if you share your computer with other users. Please note that this feature is local to the current browser and computer that you are using now.)

Sign-In



FUGGLE

Fuzzing

Using

Google

Gets

Low hanging fruit

Easily

Fugggle™



FUGGLE

Fuzzing

Using

Google

Gets

Low hanging fruit

Easily

Fuggle™





Hackers steal credit card info from R.I. Web site

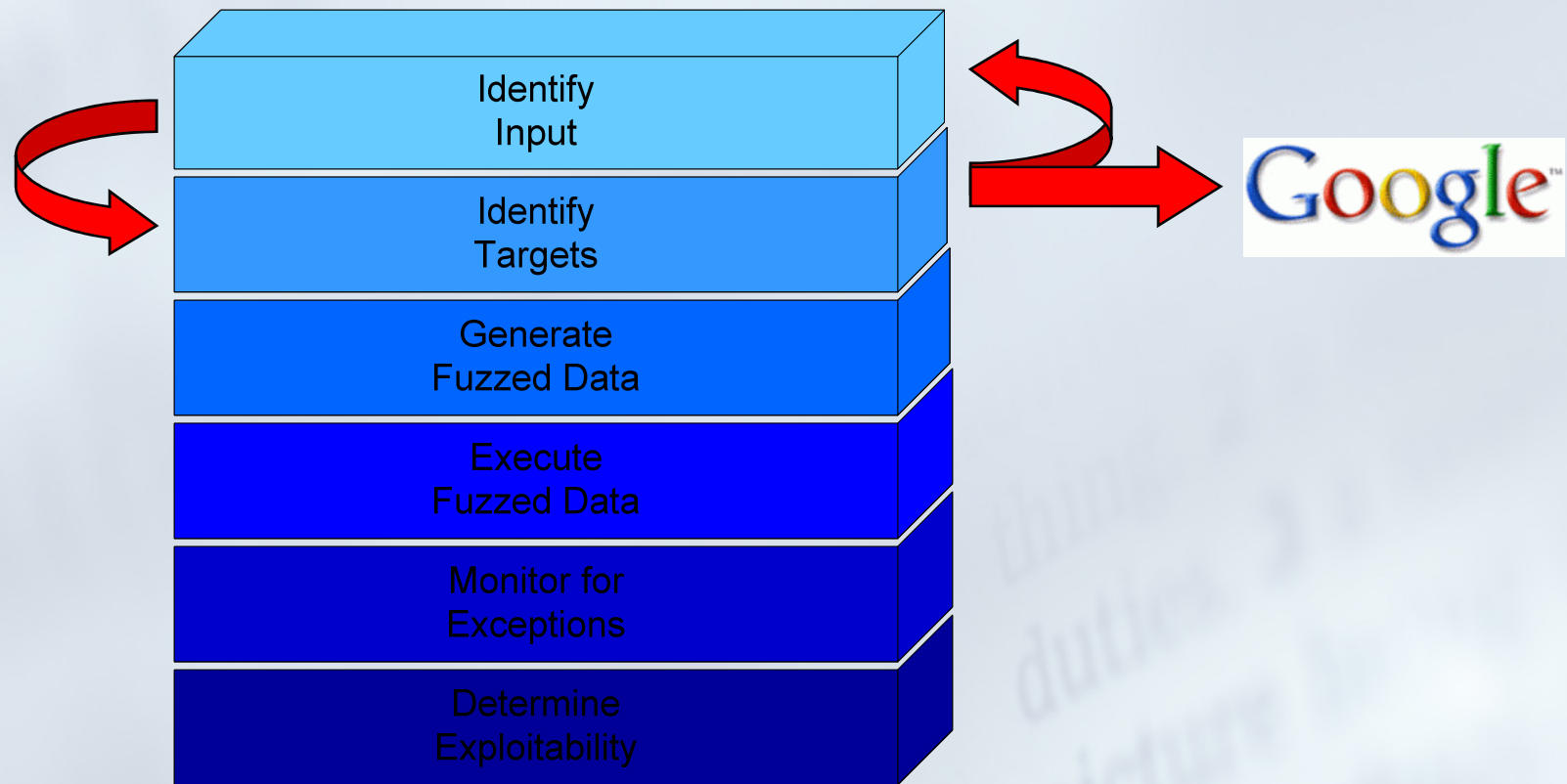
Dibya Sarkar

Published on Jan. 27, 2006

A Russian hackers broke into a Rhode Island government Web site and allegedly [stole credit card data](#) from individuals who have done business online with state agencies.

The story was first reported by The Providence Journal this morning and comes two days after state and local government officials released national surveys indicating they need more cybersecurity guidance and help in strengthening their systems.

Fuggle Fuzzing Phases



Fuggle vs. Google Hacking

| Fuggle | Google Hacking |
|--|---|
| Focus on input <i>e.g. URI parameters</i> | Focus on output <i>e.g. page content</i> |
| Identifying targets for further testing | Identifying pages using vulnerable 3 rd party apps or leaking confidential information |
| Flexible search terms <i>e.g. inurl:"id=10"</i> | Fixed signature based searches <i>e.g. intitle:index.of "parent directory"</i> |
| Custom vulnerabilities | Known vulnerabilities |

Fuggle Prerequisites

Fuzz Variables

- Input vectors must be indexed by Google and accessible via search operators
 - ✓ - Title
 - ✓ - Displayed page content
 - ✓ - URI
 - ✗ - Request/response headers
 - ✗ - Page source code

Limitations

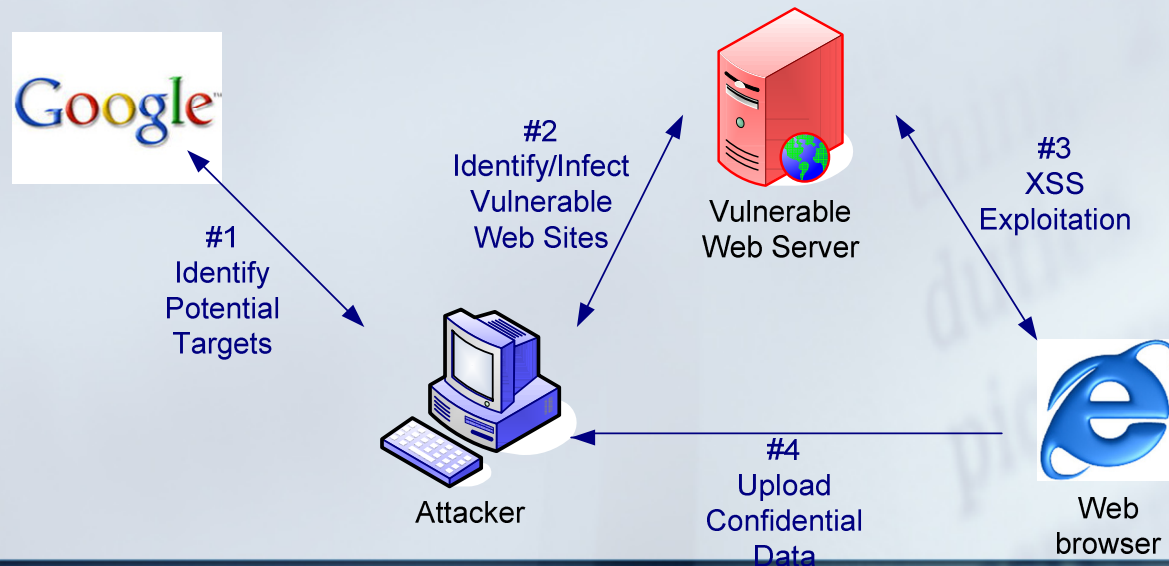
- Effectively limits using Fuggle to pages using GET method
 - Input vectors indexed in URL



Fuggle Threat

How can Fuggle be abused?

- Indiscriminate web application hacking
- Vulnerability scanning for self propagating worms / web application worms



Fuggle SQL Injection – Identify Input

Input

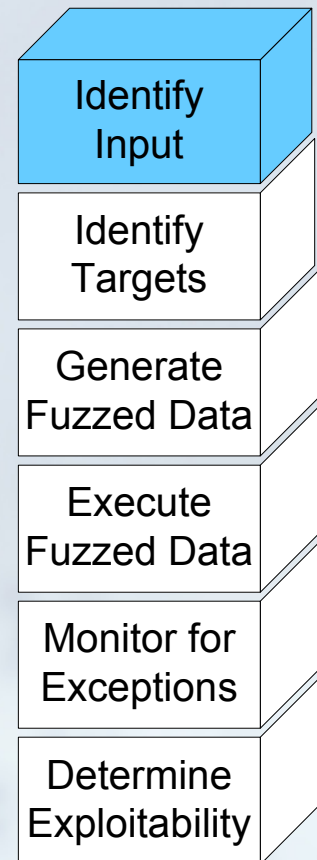
- User supplied values concatenated into SQL queries

`www.example.com?id=10`

`SELECT product from products WHERE id=10;`

Goal

- Identify pages with verbose SQL errors



Fuggle SQL Injection – Identify Targets

Search Term

- inurl:"id=10"

Targets

- Retail stores
 - E.g. Product catalog
- Informational sites
 - E.g. News archive

Search results

- Results 1 - 10 of about **2,010,000** for inurl:"id=10". (0.05 seconds)

Cleanse results

- Remove URLs w/out "id=10"
- Remove duplicate results form single domain

Identify
Input

Identify
Targets

Generate
Fuzzed Data

Execute
Fuzzed Data

Monitor for
Exceptions

Determine
Exploitability



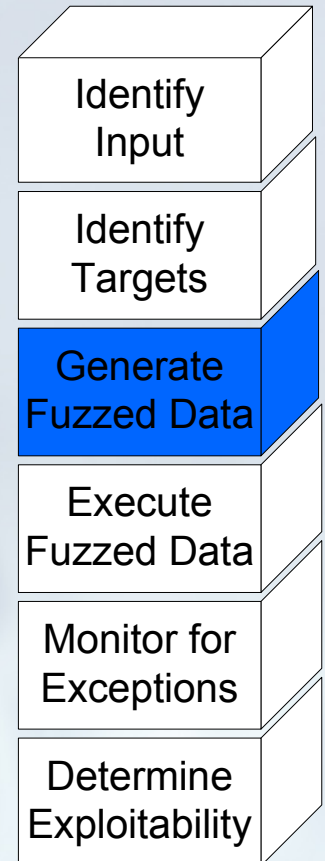
Fuggle SQL Injection – Generate Data

Goal

- Identify pages with verbose SQL errors

Fuzz data

- Verbose SQL injection
 - `id='10`
- Blind SQL injection
 - `id=10 OR 1=1`
- Comment remainder of query
 - `id='10--`
- Encode query
 - `id=%2710`



Fuggle SQL Injection – Execute Data

Submit queries

Capture responses

- Raw response
 - Headers
 - HTML source code
 - HTML Status codes

Associate requests with responses

Archive for automated and manual review

Identify
Input

Identify
Targets

Generate
Fuzzed Data

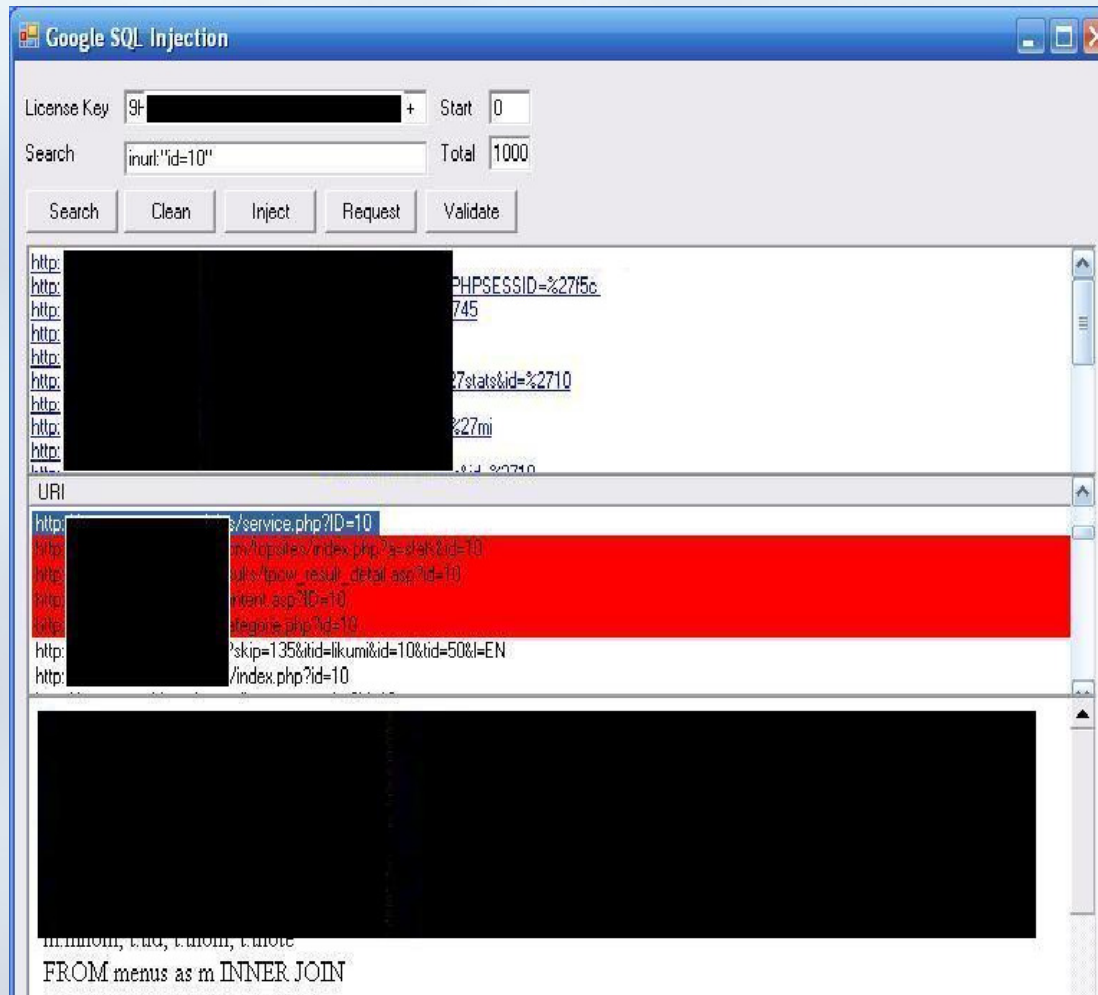
Execute
Fuzzed Data

Monitor for
Exceptions

Determine
Exploitability



Fuggle SQL Injection – Monitor Exceptions



Identify
Input

Identify
Targets

Generate
Fuzzed Data

Execute
Fuzzed Data

Monitor for
Exceptions

Determine
Exploitability



Fuggle SQL Injection – Exploitability

Confidentiality

- SELECT

Integrity

- DROP
- INSERT
- DELETE

System compromise

- Stored procedures
- Extended stored procedures

Identify
Input

Identify
Targets

Generate
Fuzzed Data

Execute
Fuzzed Data

Monitor for
Exceptions

Determine
Exploitability



Fugggle SQL Injection – Results

| | |
|--|--------------|
| Initial population of URLs | 1,000 |
| Population after removal of duplicate servers | 732 |
| Population after removal of failed requests | 708 |
| Total number of verbose SQL errors | 80 |
| Percentage of sample web sites potentially vulnerable to SQL injection attacks | 11.3% |

Questions



Michael Sutton, Security Evangelist

<http://portal.spidynamics.com/blogs/msutton>

Michael.Sutton@hp.com

