Web Application Security

Adam Doupé

CS 177

10/28/13

THE COMPUTER SECURITY GROUP AT UCSB

Me

- 9 years as UCSB student (yikes)
  - 4th year PhD student
- ~ 1 year at Microsoft as SDE
- Research securing web applications
- Professional pentester

USA TODAY

Search

ON AIR NOW ›

[IP host]

hackers gone wild

the fast times & hard fall of the green hat gang

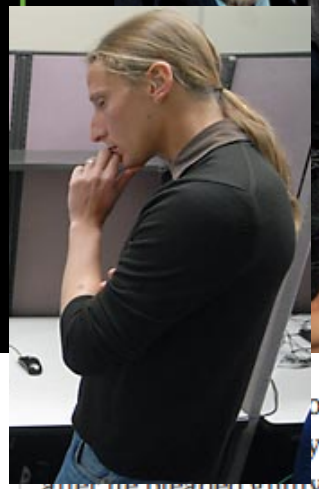
how three teenage friends, fueled by sex, drugs and illegal code, pulled off the biggest cybercrime of all time

// by sabrina rubin erdely

THEY'D BEEN HIGH ALL WEEKEND LONG – ON ECSTASY, COKE, MUSHROOMS AND acid – so there seemed little harm in doing one last bump of Special K while they packed up to leave their $5,000-a-night duplex in South Beach. For the past three days, the three friends had barely bothered leaving their hotel, as a dozen club kids in town for Winter Music Conference, the annual festival that draws DJs and ravers from all over the world, flocked to their luxury suite to partake of the drug smorgasbord laid out on the coffee table. But even stoned on industrial-

the nonstop party

Albert Gonzalez's crew lived a lifestyle as outrageous as their crimes. **(1)** Albert stole 170 million credit-card numbers while relaxing at places like New York's Hotel on Rivington. **(2)** Patrick Toey was his best operative, and **(3)** Stephen Watt was the group's coding genius. Their cybercrimes netted millions **(4)**, enabling Albert to throw a $75,000 birthday party for himself and Stephen at an exclusive New York club **(5)**.



News in pictures

BitFloor 433

suffered Sha

ost notorious computer hackers
years in prison on Thursday
after he pleaded guilty to helping run a global ring

Related articles

Identity of 'computer hacker' is revealed

Ethics

- Only hack into sites you own
  - Or you have permission
- You will get caught

Tech

- HTML
- CSS
- JavaScript
- SQL
- Server-Side (Python/PHP/Ruby)

Many Vulnerabilities

- Cross-Site Scripting (XSS)
- SQL Injection
- Cross-Site Request Forgery (XSRF)
- HTTP Parameter Pollution (HPP)
- Command Injection
- Parameter Manipulation
- File Exposure
- Directory Traversal
- Forced Browsing
- Logic Flaws
- Execution After Redirect (EAR)

Many Vulnerabilities

- Cross-Site Scripting (XSS)
- SQL Injection
- Cross-Site Request Forgery (XSRF)
- HTTP Parameter Pollution (HPP)
- Command Injection
- Parameter Manipulation
- File Exposure
- Directory Traversal
- Forced Browsing
- Logic Flaws
- Execution After Redirect (EAR)

Tech

- HTML
- CSS
- JavaScript
- SQL
- Server-Side (Python/PHP/Ruby)

Hacker Mindset

- Understand the application
  - Build mental model
- Only need to find one flaw

Injection Vectors

- User input to the application
- Web application
  - Query parameters
  - POST parameters
  - Cookies
  - Referer header
  - Files

SQL – Structured Query Language

- SQL Commands: SELECT, UPDATE, INSERT

- General form of a query
  `SELECT * FROM Users WHERE userName = 'dick'`

- Majority of web applications interact with a database (and SQL)

SQL Injection

- Allows attacker to alter semantics of SQL query

- Consequences
  - Steal database
  - Alter database
  - Bypass login

SQL Injection – Example

```
“select * from `users` where `id`
=‘” + $id + “’;”

$id = “10”

select * from `users` where `id` =
‘10’;
```

SQL Injection – Example

```
“select * from `users` where `id`
=‘” + $id + “’;”
```

```
$id = “-1 or 1=1”
```

```
select * from `users` where `id` =
‘-1 or 1=1’;
```

SQL Injection – Example

```
“select * from `users` where `id` =‘” + $id + “’;”

$id = “-1’ or 1=1”

select * from `users` where `id` = ‘-1’ or 1=1’;
```

SQL Injection – Example

```
“select * from `users` where `id`
=‘” + $id + “’;”

$id = “-1’ or 1=1; #”

select * from `users` where `id` =
‘-1’ or 1=1; #’;
```

SQL Injection – Example

```
“select * from `users` where `id` =‘” + $id + “’;”

$id = “-1’; drop table `users`;#”

select * from `users` where `id` = ‘-1’; drop table `users`;#’;
```

SQL Injection – Examples

```
“select * from `users` where `id` =‘”
+ $id + “’;”
```

```
$id = “-1’; insert into `admin`
(‘username’, ‘password’) values
(‘adamd’, ‘pwned’);#”
```

```
select * from `users` where `id` =
‘-1’; insert into `admin` (‘username’,
‘password’) values (‘adamd’,
‘pwned’);#’;
```

SQL Injection – Detection

- Passive – Look for success
  - `1+2`
  - `(select 2)`

- Active – Look for errors
  - `O'Malley`
  - `< 10`

SQL Injection – Prevention

- Prepared statements
  - Specify structure of query then provide arguments
- Prepared statements – example

```
$stmt = $db->prepare(“select * from `users`
where `username` = :name and `password` =
SHA1( CONCAT(:pass, `salt`)) limit 1;”);
$stmt->bindParam(':name', $name);
$stmt->bindParam(':pass', $pass);
```

- Sanitize inputs

Cross-Site Scripting (XSS)

- Malicious JavaScript running in the context of your web application

- Consequences
  - Steal cookies
  - Perform actions as the user
  - Present fake login form

XSS – Examples

```
<html>
  <body>
    <p>Hello <?= $name ?></p>
  </body>
</html>
```

XSS – Examples

```
$name = “adam”;
<html>
   <body>
      <p>Hello adam</p>
   </body>
</html>
```

XSS – Examples

```
$name = “<script>alert(‘xss’);
         </script>”;

<html>
   <body>
      <p>Hello <script>alert(‘xss’);
</script></p>
   </body>
</html>
```

XSS – Detection

- Understand how input is used in HTML source
- Input “forbidden” characters
  - < >
  - ‘ “ ; /
- Understand what sanitization is performed

XSS – Prevention

- Sanitize all user inputs using known sanitization routine

- Depends on context
  - < and > necessary in HTML
  - Only need ‘ in JavaScript

Review

- Hacker mindset
  - Understand the application
  - Build a mental model
  - Break the mental model

Homework

- Get the password
  - Part 1: http://192.35.222.169/pt1/
  - Part 2 (admin password): http://192.35.222.169/pt2/

- Questions to answer
  - What type of vulnerability?
  - Where is the vulnerability?
  - How does the application work?
  - How would you fix the vulnerability?

Tools

- Wireshark
- Burp Proxy
- SQLMap
- OWASP Broken Web Apps Project
- Google Gruyere

Questions?

Go forth and hack!

(ethically, of course)

adoupe@cs.ucsb.edu