BISHOP FOX®

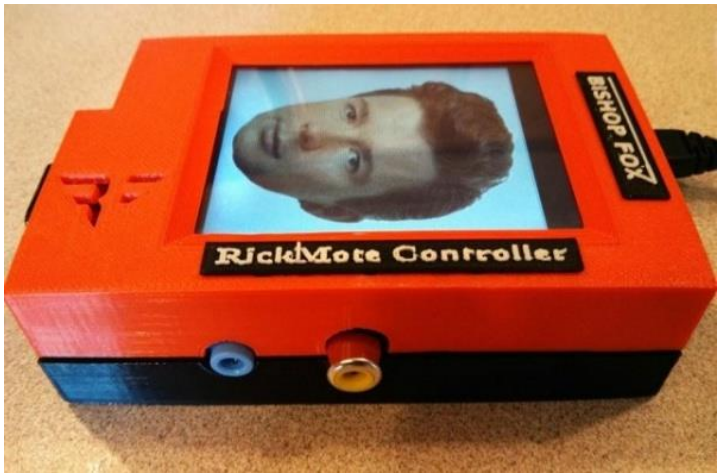
Protecting the Customer-Facing Website

October 27, 2014

Who am I?

Bishop Fox

- Dan Petro
- *Senior Security Analyst*



Rickmote

Categorizing Attacks

Server Attacks – Compromising the server or related infrastructure

- Web servers
- Database Servers
- Load Balancers

Client Attacks – Tricking users into performing malicious actions

Common Vulnerabilities

(NOT A COMPLETE LIST)

Server Attacks

- Insecure File Upload
  - Place malicious file on server
- SQL Injection
- Command Injection
- DDoS — Distributed Denial of Service

Client Attacks

- Cross-site Scripting (XSS)
  - Persistent and reflective
- Cross-site Request Forgery
  - Trick users into making requests to other sites

INSECURE FILE UPLOAD

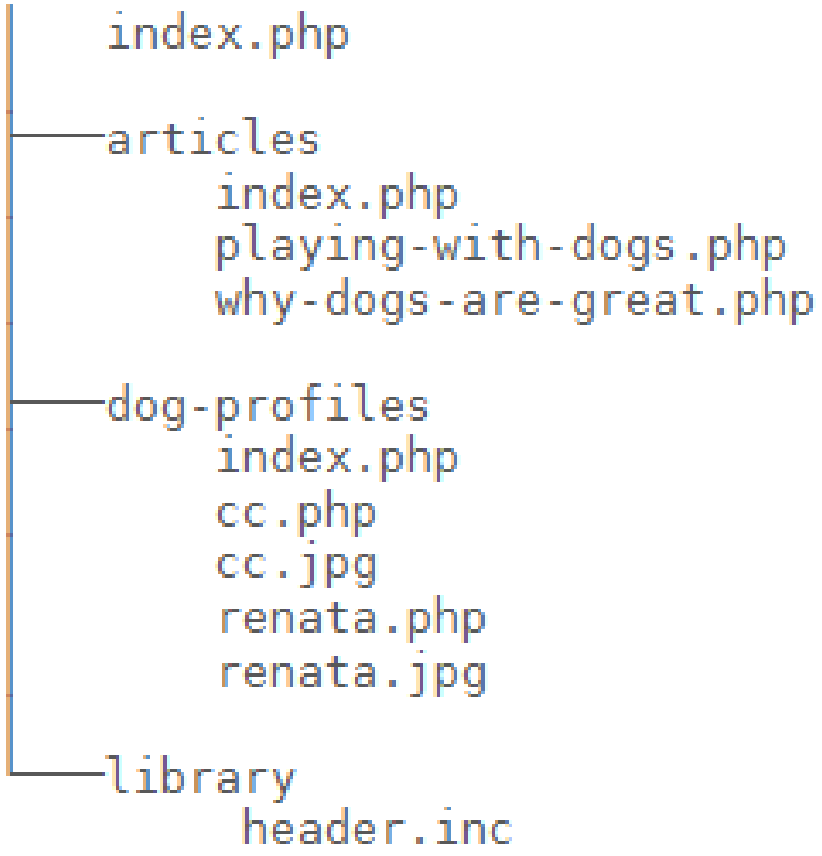
EASY ROAD TO OWNING SERVERS

Webserver Filesystem

DIRECTORY TREE

- Serves documents from the server webroot
- "**Static**" files like JPEGs are simply sent to the user
- "**Dynamic**" files like PHP are interpreted as code

- ...what if an attacker could place their own PHP file here?
  - They could **execute code** on the vulnerable server!



Vulnerable Ingredients

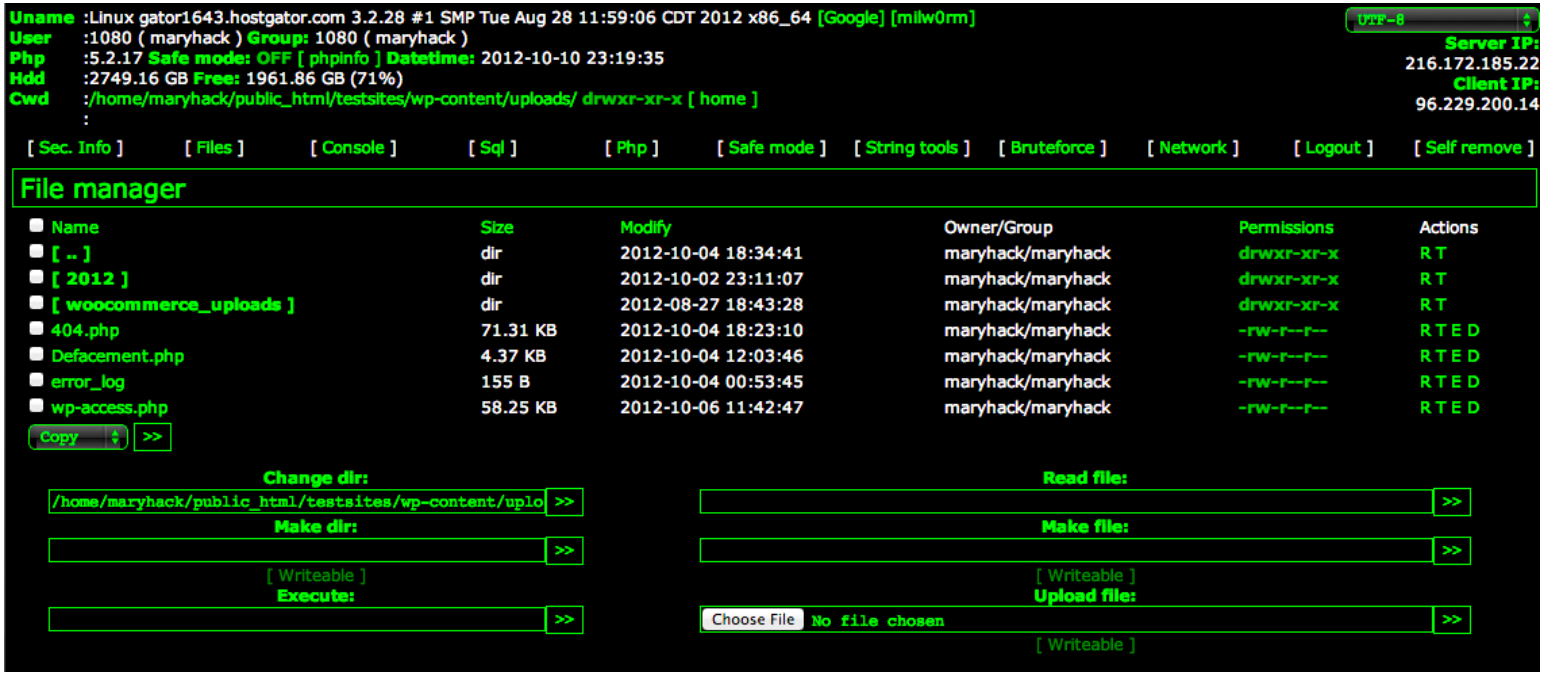
TOIL AND TROUBLE

- **File Upload Functionality** – Avatars, documents, pictures, etc…
- **Upload to webroot** – Some servers upload directly to S3, a file server, or a segmented file directory (outside of webroot)
- **Insufficient Input Validation** – File extension checking for some frameworks, content checking for others

Result

YOU WON'T LIKE IT

- Code execution
  - Rewriting application logic
  - Accessing sensitive resources



SQL INJECTION

DATABASE TAKEOVER

SQL Injection

SLIDE SUBTITLE

- Confusion of **data** and **code**
  - Database SQL statements
  - User-supplied data
  - Hijacks statements, thus database

- `SELECT * FROM Customers WHERE Country='QUERY';`

Vulnerable Queries

DON'T DO THESE

Original Code

- `SELECT * FROM Customers WHERE Country='QUERY';`

Resulting Query

- `SELECT * FROM Customers WHERE Country='America';`

Normal Input

America

Injection Example

CHECK THIS OUT

Original Code

- `SELECT * FROM Customers WHERE Country='QUERY';`

Injection Input

```
a' UNION SELECT *
FROM cc_numbers;--
```

Resulting Query

- `SELECT * FROM Customers WHERE Country='a' or UNION SELECT * FROM cc_numbers;-- ';`

Or Even Worse!

CODE EXECUTION

To also take over the webserver:

- **OUTFILE** in MySQL
  - Can direct output of a statement to file
  - Can possibly write malicious content to webroot
  - Just as in Insecure File Upload
- **XP_CMDSHELL** in MSSQL
  - Direct terminal commands
  - Executed on the webserver

Injection Immunity

TAKE YOUR SHOTS

Parameterized Queries

- **Example in PHP**

```
$stmt = $dbh->prepare("INSERT INTO REGISTRY
(name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);
```

- Inputs are not merely inserted
- Safely accepted as parameters

CROSS-SITE SCRIPTING (XSS)

JUST WHEN YOU THOUGHT JAVASCRIPT WAS SAFE

Cross-site Scripting

XSS

Another confusion of data and code

- This time in HTML
- Attacks users, not servers
- Allows an attacker to hijack user accounts

Example: (normal use)

- `www.bank.com/account?name=`**`John`**

```
<input type="text" name="state" value="John">
```

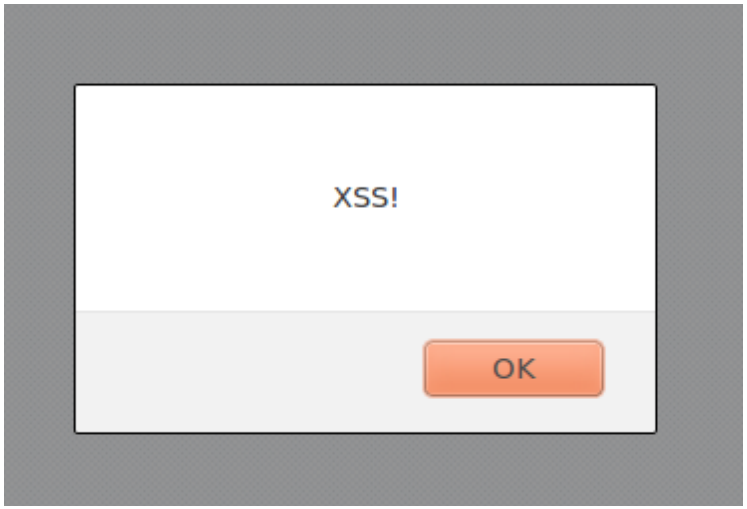
Cross-site Scripting

XSS

Reflective XSS Example:

```
www.bank.com/account?name=a"><script>alert('XSS!')</script>
```

```
<input type="text" name="state“ value=“a"><script>alert('XSS!')</script>
```



Cross-site Scripting (Fixed)

Escaped Example:

www.bank.com/account?name=**a"><script>alert('XSS!')</script>**

```
<input type="text" name="state“ value=“a&quot; >&lt;script&gt;alert('XSS!') &lt; /script&gt;”>
```

---

a"><script>alert('XSS!') < /script>

STAYING SAFE

SECURE CODING PRACTICES

Good News!

SECURITY DOESN'T HAVE TO BE EXPENSIVE

Firewalls are nice, but good code is better

- All of the previous issues can be avoided with secure coding practices – not expensive firewalls or products

Recommendations:

- Establish a **secure development lifecycle** practice within your organization
- Instill a culture of **caring for the code** (take it personally!)
- Peer review your code, **open source** it!

Thank you!

BISHOP FOX®

We’re Hiring

www.bishopfox.com

contact@bishopfox.com

BISHOP FOX®