HTTP Parameter Pollution
Vulnerabilities in Web Applications

Marco `embyte` Balduzzi
Who am I?

- From Bergamo (IT) to the French Riviera
- MSc in Computer Engineering
- PhD at EURECOM
- 8+ years experience in IT Security
- Engineer and consultant for different international firms
- Co-founder of BGLug, Applied Uni Lab, (ex) SPINE Group, Nast, etc…

http://www.iseclab.org/people/embyte
The Web as We Know It

- Has evolved from being a collection of simple and static pages to fully dynamic applications
- Applications are more complex than they used to be
- Multi-tier architecture is the normal
- Many complex systems have web interfaces
The Web before

Mosaic Communications Internet Directory


This is a large subject directory that also allows you to search for subjects you may be interested in.

Yahoo
Stanford University offers a directory of Internet services that features over 13,000 entries. Yahoo also offers listings of the most popular pages, the ability to search the directory of services, and (if you're up for adventure) a way to visit a random page.

Subject Catalog
Type of Service
The World-Wide Web Virtual Library offers a broad range of topics searchable by subject as well as service type.

Best of the Web '94
This directory is a collection its voters call the best Internet services available in 1994.

http://akebono.stanford.edu/yahoo/
Increased Importance of Web Security

As a consequence:

- Web security has increased in importance
- OWASP, the Top Ten Project
- Attack against web apps constitute 60% of attacks on the Internet (SANS’s The Top Cyber Security Risks)
- Application being targeted for hosting drive-by-download content or C&C servers
- Malware targeting browsers (e.g. key and network loggers)
Increased Importance of Web Security

- A lot of work done to detect injection type flaws:
  - SQL Injection
  - Cross Site Scripting
  - Command Injection

- Injection vulnerabilities have been well-studied, and tools exist
  - Sanitization routines in languages (e.g., PHP)
  - Static code analysis (e.g., Pixy, OWASP Orizon)
  - Dynamic techniques (e.g., Huang et al.)
  - Web Application Firewalls (WAF)
HTTP Parameter Pollution

- A new class of Injection Vulnerability called HTTP Parameter Pollution (HPP) is less known
  - Has not received much attention
  - First presented by S. di Paola and L. Carettoni at OWASP 2009
- Attack consists of injecting encoded query string delimiters into existing HTTP parameters (e.g. GET/POST/Cookie)
  - If application does not sanitize its inputs, HPP can be used to launch client-side or server-side attacks
  - Attacker may be able to override existing parameter values, inject a new parameter or exploit variables out of a direct reach
HTTP Parameter Handling

- During interaction with web application, client provides parameters via GET/POST/Cookie
- HTTP allows the same parameter to be provided twice
  - E.g., in a form checkbox
    http://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_checkbox
- What happens when the same parameter is provided twice?
  - http://www.google.com/search?q=italy&q=china
  - http://www.site.com/login?user=alice&user=bob
Google example

http://www.google.com/search?q=italy&q=china

Circa 784.000.000 risultati (0,08 secondi)

Fondazione Italia Cina
La Fondazione promuove la realizzazione di una Cabina di Regia con riferimento al garantire un efficace raccordo tra pubblico e privato e dare ...
Contatti - Chi Siamo - CV Online - Il Presidente
italychina.org/ - Copia cache - Simili

Zhongguo Cina :: Associazione Italia-Cina
Presenta informazioni sull'associazione, notizie sulla Cina, appuntamenti, pagine e turismo.
italiacina.org/ - Copia cache - Simili
Yahoo example

http://search.yahoo.com/search;_ylt=AjjaXqxBvy_VBz8JmVbmegebvZx4?p=italy&p=china

Also try: china ambassador, china airlines, china ufo, ...

China - Wikipedia, the free encyclopedia

With nearly 4,000 years of continuous history, China is one of the world's oldest civilizations. Prior to the 19th century, it possessed one of the most advanced societies and economies...

en.wikipedia.org/wiki/China - Cached
HTTP Parameter Handling

- We manually tested common methods of 5 different languages

<table>
<thead>
<tr>
<th>Technology/Server</th>
<th>Tested Method</th>
<th>Parameter Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP/IIS</td>
<td>Request.QueryString(&quot;par&quot;)</td>
<td>All (comma-delimited string)</td>
</tr>
<tr>
<td>PHP/Apache</td>
<td>$_GET(&quot;par&quot;)</td>
<td>Last</td>
</tr>
<tr>
<td>JSP/Tomcat</td>
<td>Request.getParameter(&quot;par&quot;)</td>
<td>First</td>
</tr>
<tr>
<td>Perl(CGI)/Apache</td>
<td>Param(&quot;par&quot;)</td>
<td>First</td>
</tr>
<tr>
<td>Python/Apache</td>
<td>getvalue(&quot;par&quot;)</td>
<td>All (List)</td>
</tr>
</tbody>
</table>

- There is nothing bad with it, if the developer is aware of this behavior
  - Languages provide secure functions (python’s getfirst())
HTTP Parameter Pollution (Client-Side)

Attacker generates The *Trigger URL* sent to the victims

Site vulnerable to HTTP Parameter Pollution

Examples:
- Deletion of personal emails
- Generation of custom friend-requests
- Posting malicious wall posts
- Purchasing unintended products
- Unintended voting

Blackhat Webcast Series - 28 May 2011
Example: Unintended voting

- An application for voting between two candidates
- The two links are built from the URL

**Url**: http://host/election.jsp?poll_id=4568

**Link1**: <a href="vote.jsp?poll_id=4568&candidate=white">Vote for Mr.White</a>
**Link2**: <a href="vote.jsp?poll_id=4568&candidate=green">Vote for Mrs.Green</a>

- No sanitization

**ID** = Request.getParameter("poll_id")
**href_link** = “vote.jsp?poll_id=” + ID + ”&candidate=xyz”
Example: Unintended voting

- poll_id is vulnerable
- Attacker generate a Trigger URL to be sent to his victims:
  - http://host/election.jsp?p0ll_id=4568&candidate=green

- The resulting page now contains injected links:

```html
<a href=vote.jsp?pool_id=4568&candidate=green&candidate=white>
  Vote for Mr. White </a>
<a href=vote.jsp?pool_id=4568&candidate=green&candidate=green>
  Vote for Mrs. Green </a>
```

- Candidate Mrs. Green is always voted!
Example: Misleading shopping users
HTTP Parameter Pollution (Server-Side)

Attacker generates The Trigger URL

- Used to exploit the server-side logic of the web-application
- The attacker sends the Trigger URL to the vulnerable application
HPP Server-Side: Example #1

- E.g., Payment system (di Paola / Carettoni)

```java
void private executeBackendRequest(HTTPRequest request){
    String amount=request.getParameter("amount");
    String beneficiary=request.getParameter("recipient");
    HttpRequest("http://backendServer.com/servlet/actions","POST",
            action=transfer&amount="+amount+"&recipient="+beneficiary);
}
```

Trigger URL: http://frontendHost.com/page?amount=1000&recipient=Mat%26action%3dwithdraw

Injected query on the backend:
HttpRequest("http://backendServer.com/servlet/actions","POST",
        action=transfer&amount=1000&recipient=Mat&action=withdraw");
HPP Server-Side: Example #2

- E.g., Access the user passwords
- ASP concatenates the values of two parameters with the same name with a comma
- This permits to inject and modify the query on the database

Normal requests:
URL: printEmploys?department=engineering
Back-end: dbconnect.asp?what=users&department=engineering
Database: select users from table where department=engineering

HPP injected requests:
URL: printEmploys?department=engineering%26what%3Dpasswd
Database: select users, passwd from table where department=engineering
Parameter Pollution – More uses

1) Cross-channel pollution
- Override parameters between different input channels (GET/POST/Cookie)
- Good security practice: accept parameters only from where they are supposed to be supplied

2) Bypass CSRF tokens
- E.g. Yahoo Mail client-side attack (di Paola & Carettoni)
- The user’s mails get automatically deleted!

Url: showFolder?fid=Inbox&order=down&tt=245&pSize=25&startMid=0%2526cmd=fmgt.emptytrash%26DEL=1%26DelFID=Inbox%26cmd=fmgt.delete

Link: showMessage?sort=date&order=down&startMid=0%26cmd%3Dfmgt.emptytrash&DEL=1&DelFID=Inbox&cmd=fmgt.delete&.rand=1076957714
Parameter Pollution – More uses

3) Bypass WAFs input validation checks
   • Split & Join the attack payload
   • E.g., SQL injection via parameter replication
   • Exploit ASP concatenation behavior and inline comments

Standard: show_user.aspx?id=5;select1,2,3+from+users+where+id=1–
Over HPP: show_user.aspx?id=5;select+1&i$d=2&i$d=3+from+users+where+id=1–

Standard: show_user.aspx?id=5+union+select++from+users–
Over HPP: show_user.aspx?id=5/*&i$d=/union/*&i$d=/select++/*/&i$d=/from+users--
System for HPP Detection

- Four main components: browser, crawler, two scanners

- Diagram showing the system with components: DB, Stat Generator, Crawler, Instrumented Browser, V-Scan, P-Scan, Browser Extension, reports, and internet connection.
P-Scan: Analysis of the Parameter Precedence

- Analyzes a page to determine the precedence of parameters, when multiple occurrences of the same parameter are submitted.
- Take parameter par1=val1, generate a similar value par1=new_val
  - Page0 (original): app.php?par1=val1
  - Page1 (test 1) : app.php?par1=new_val
  - Page2 (test 2) : app.php?par1=val1&par1=new_val

- We compare how similar are the pages
  - Page0==Page2 -> precedence on first parameter
  - Page1==Page2 -> precedence on second parameter

- Check for errors on test 2
V-Scan: Testing for HPP vulnerabilities

- For every page, an innocuous URL-encoded parameter (nonce) is injected in the page’s parameters
  - E.g., ?q=italy%26foo%3Dbar

- The page is submitted (GET/POST)

- Then, the answered page is checked for containing the decoded version of the nonce (&foo=bar):
  - In links or forms (action)
Where to inject the nonce

- \( P_A = P_{URL} \cap P_{Body} \): set of parameters that appear unmodified in the URL and in the page content (links, forms)
- \( P_B = p \mid p \in P_{URL} \land p \notin P_{Body} \): URL parameters that do not appear in the page. Some of these parameters may appear in the page under a different name
- \( P_C = p \mid p \notin P_{URL} \land p \in P_{Body} \): set of parameters that appear somewhere in the page, but that are not present in the URL
Reducing the False Positives

- E.g., one of the URL parameters (or part of it) is used as the entire target of a link

  **Url:** index.php?v1=pl&uri=apps%2Femail.jsp%3Fvar1%3Dpar1%26foo%3Dbar
  **Link:** apps/email.jsp?var1=par1&foo=bar

- Self-referencing links

  **Url:** search.html?session_id=jKAmsZx5%26foo%3Dbar&q=shoes
  **Link:** service_request.html?page=search%2Ehtml%3Fs session_id%3DjKAmsZx5&foo=bar&q=shoes

- Similar issues with printing, sharing functionalities

- To reduce false positives, we use heuristics
  - E.g., the injected parameter does not start with http://
  - Injection without URL-encoding
Implementation – The PAPAS tool

- PAPAS: Parameter Pollution Analysis System
- The components communicate via TCP/IP sockets
  - Crawler and Scanner are in Python
  - The browser component has been implemented as a Firefox extension
  - Advantage: We can see exactly how pages are rendered (cope with client-side scripts, e.g. Javascript)
  - Support for multiple sessions (parallelization)

- Now, as a free-to-use-service:
  - [http://papas.iseclab.org](http://papas.iseclab.org)
Two set of experiments

1) We used PAPAS to scan a set of popular websites
   About 5,000 sites collected by the first 500 of Alexa’s main categories
   The aim: To quickly scan as many websites as possible and to see how common HPP flaws are

2) We then analyzed some of the sites we identified to be HPP-vulnerable in more detail
The 5,016 tested sites

<table>
<thead>
<tr>
<th>Categories</th>
<th># of Tested Applications</th>
<th>Categories</th>
<th># of Tested Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial</td>
<td>110</td>
<td>Shopping</td>
<td>460</td>
</tr>
<tr>
<td>Games</td>
<td>300</td>
<td>Social Networking</td>
<td>117</td>
</tr>
<tr>
<td>Government</td>
<td>132</td>
<td>Sports</td>
<td>256</td>
</tr>
<tr>
<td>Health</td>
<td>235</td>
<td>Travel</td>
<td>175</td>
</tr>
<tr>
<td>Internet</td>
<td>698</td>
<td>University</td>
<td>91</td>
</tr>
<tr>
<td>News</td>
<td>599</td>
<td>Video</td>
<td>114</td>
</tr>
<tr>
<td>Organization</td>
<td>106</td>
<td>Others</td>
<td>1,401</td>
</tr>
<tr>
<td>Science</td>
<td>222</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Efficient assessment

- In 13 days, we tested 5,016 sites and more than 149,000 unique pages
- To maximize the speed, the scanner
  - Crawled pages up to a distance of 3 from the homepage
  - Considered links with at least one parameter (except for the homepage)
  - Considered at max 5 instances for page (same page, different query string)
  - We disabled pop-ups, images, plug-ins for active content technologies
Evaluation – Parameter Precedence

- **Database Errors**
  - Web developers do not seem conscious of the possibility to duplicate GET/POST parameter
  - No sanitization is in place

<table>
<thead>
<tr>
<th>Parameter Precedence</th>
<th>WebSites</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last</td>
<td>2,237</td>
<td>44.60%</td>
</tr>
<tr>
<td>First</td>
<td>946</td>
<td>18.86%</td>
</tr>
<tr>
<td>Union</td>
<td>381</td>
<td>7.60%</td>
</tr>
<tr>
<td>Inconsistent</td>
<td>1,251</td>
<td>24.94%</td>
</tr>
<tr>
<td>Unknown</td>
<td>201</td>
<td>4.00%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>5,016</strong></td>
<td><strong>100.00%</strong></td>
</tr>
</tbody>
</table>

Database Errors: 238 (4.74%)
Nasa.gov: coldfusion SQL Error

The web site you are accessing has experienced an unexpected error. Please contact the website administrator.

The following information is meant for the website developer for debugging purposes.

Error Occurred While Processing Request

The cause of this output exception was that:
coldfusion.tagext.sql.QueryParamTag$InvalidDataException:
Invalid data value 23,24 exceeds maxlength setting 4..

Resources:
- Enable Robust Exception Information to provide greater detail about the source of errors. In the Administrator, click Debugging & Logging > Debug Output Settings, and select the Robust Exception Information option.
- Check the ColdFusion documentation to verify that you are using the correct syntax.
- Search the Knowledge Base to find a solution to your problem.

Browser: Mozilla/5.0 (X11; U; Linux i686; it; rv:1.9.2.3) Gecko/20100401
Firefox/3.6.3
Remote Address: 193.253.230.214
Referrer: http://www.jpl.nasa.gov/multimedia/slideshows/index.cfm?id=23&page=1%26id%3D24
Date/Time: 07-Jun-10 07:44 AM
Evaluation – Parameter Precedence

- **Parameter Inconsistency**
  - Sites developed using a combination of heterogeneous technologies (e.g. PHP and Perl)
  - This is perfectly safe if the developer is aware of the HPP threat… this is not always the case

<table>
<thead>
<tr>
<th>Parameter Precedence</th>
<th>WebSites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last</td>
<td>2,237</td>
</tr>
<tr>
<td>First</td>
<td>946</td>
</tr>
<tr>
<td>Union</td>
<td>381</td>
</tr>
<tr>
<td>Inconsistent</td>
<td>1,251</td>
</tr>
<tr>
<td>Unknown</td>
<td>201</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>5,016</strong></td>
</tr>
<tr>
<td>Database Errors</td>
<td>238</td>
</tr>
</tbody>
</table>

Blackhat Webcast Series - 28 May 2011
PAPAS discovered that about 1,500 (30%) websites contained at least one page vulnerable to HTTP Parameter Injection

- The tool was able to inject (and verify) an encoded parameter

- Vulnerable != Exploitable
  - Is the parameter precedence consistent?
  - Can a possible attacker override existing parameter values?
Vulnerable or Exploitable?

- Injection on link
  - Read a mail: [http://site.com/script?mail_id=10&action=read](http://site.com/script?mail_id=10&action=read)
    - Parameter in the middle -> always overriding
      - ?mail_id=10&action=delete&action=read
    - Parameter at the begin/end -> automated check via P-Scan
      - ?action=read&mail_id=10&action=delete

- Injection on form:
  - The injected value is automatically encoded by the browser
  - Still, someone may be able to run a two-step attack (client-side) or a server-side attack

- 702 applications are exploitable (14%)
More sensitive sites are equally (or even more) affected by the problem
Some Case Studies

- We investigated some of the websites in more detail
  - Among our “victims”: Facebook, Google, Symantec, Microsoft, PayPal, Flickr, FOX Video, VMWare, …
  - We notified security officers and some of the problems were fixed
  - Facebook: share component
  - Several shopping cart applications could be manipulated to change the price of an item
  - Some banks were vulnerable and we could play around with parameters
  - Google: search engine results could be manipulated
Your (secured) home banking

[Image: U.S. Bank Secured Visa® Credit Card]

[Image: U.S. Bank Young Adult Visa® Card Application]
PayPal

Raise your account limits and get Verified

For security and legal reasons, there are initial limits on how much money customers can send, receive, and withdraw using their PayPal accounts. Confirming your card raises some of these limits and makes you a Verified PayPal user.

How do I confirm my card?

Follow the steps below unless you are confirming domestic bank card.

Sample card statement

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/06/2008</td>
<td>PP (1234) EXPENSE</td>
<td>1,50 EUR</td>
</tr>
<tr>
<td>1234</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sample PayPal code

1. Enter your card information or select an existing card and click Continue. We’ll charge your card 1,50 EUR & EUPFCCFee=500.00 USD to make sure it’s yours.
2. In 2-3 days, check your bank statement (or card statement) for the unique 4-digit code that we sent along with the charge.
3. Log back into your PayPal account and enter the code from your statement.

We’ll refund the 1,50 EUR & EUPFCCFee=500.00 USD charge into your PayPal account within 24 hours after you confirm your card.
HPP Prevention

- Input validation
  - Encoded query string delimiters
- Use safe methods
  - Handle the parameter precedence
  - Channel (GET/POST/Cookie) validation
- Raise awareness
  - The client can provide the same parameter twice (or more)
Conclusion

- Presented the first technique and system to detect HPP vulnerabilities in web applications.
  - We call it PAPAS, http://papas.iseclab.org

- Conducted a large-scale study of the Internet
  - About 5,000 web sites

- Our results suggest that Parameter Pollution is a largely unknown, and wide-spread problem

- We hope that this work will help raise awareness about HPP!
References

- Minded Security Blog, S. di Paola & L. Carettoni

- White paper

- I collected a bunch of resources here:
  - http://papas.iseclab.org/cgi-bin/resources.py
Thanks for your attention.

I love you too, pollution!