HTTP Parameter Pollution

Luca Carettoni
Independent Researcher
luca.carettoni@ikkisoft.com

Stefano di Paola
CTO @ Minded Security
stefano.dipaola@mindedsecurity.com

OWASP
EU09 Poland

Copyright © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document under the terms of the OWASP License.

The OWASP Foundation
http://www.owasp.org

About us

- Luca “ikki” Carettoni
  - Penetration Testing Specialist in a worldwide financial institution
  - Security researcher for fun (and profit)
  - OWASP Italy contributor
  - I blog @ http://blog.nibblesec.org
  - Keywords: web application security, ethical hacking, Java security

- Stefano “wisec” Di Paola
  - CTO @ Minded Security Application Security Consulting
  - Director of Research @ Minded Security Labs
  - Lead of WAPT & Code Review Activities
  - OWASP Italy R&D Director
  - Sec Research (Flash Security, SWFIntruder…)
  - WebLogs http://www.wisec.it, http://blog.mindedsecurity.com

Minded security

Agenda

- Introduction
  - Server enumeration
- HPP in a nutshell
  - HPP Categories
- Server side attacks
  - Concept
  - Real world examples
- Client side attacks
  - Concept
  - Real world examples

Fact

- In modern web apps, several application layers are involved

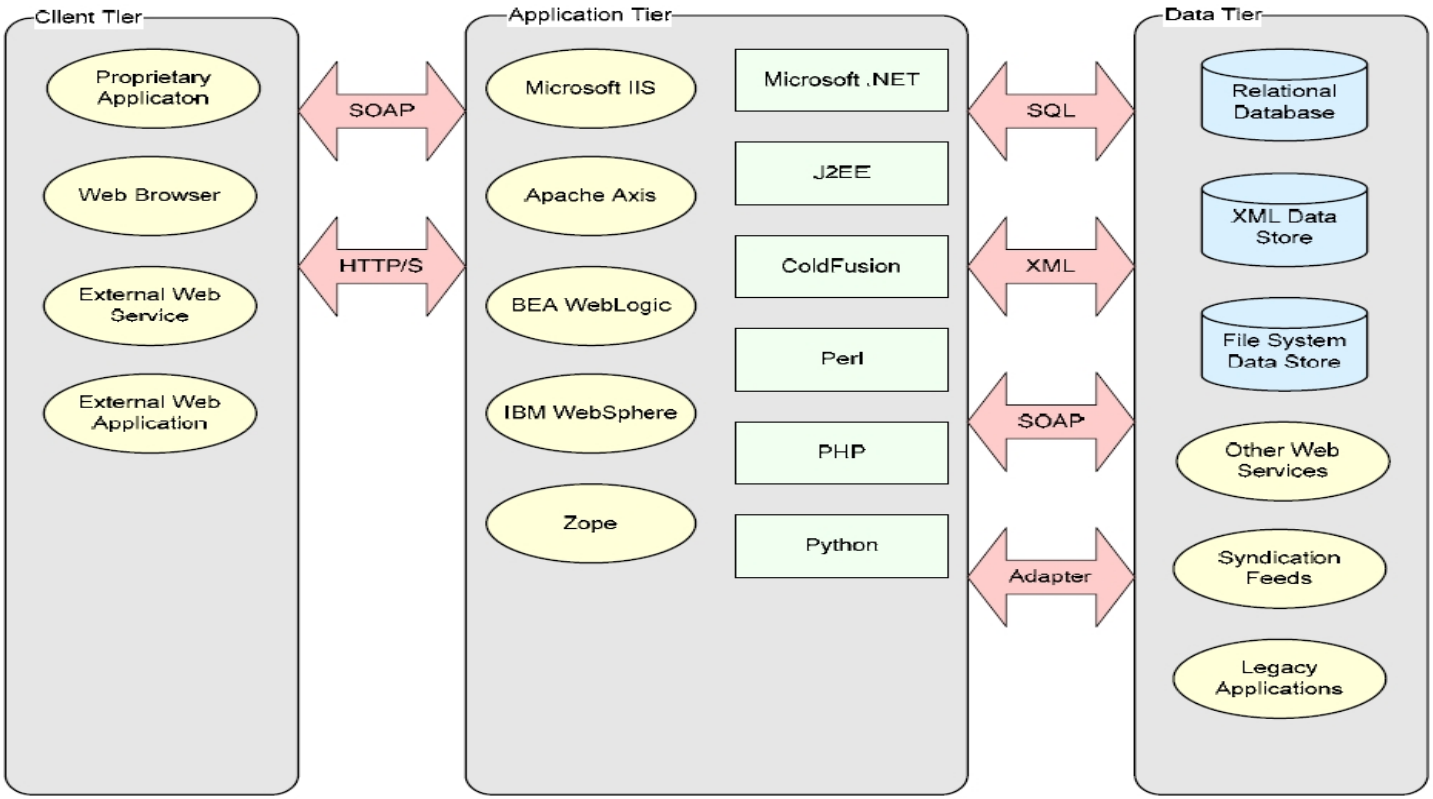


Figure 1: Web Service Deployment Tiers

Consequence

- Different input validation vulnerabilities exist
  - SQL Injection
  - LDAP Injection
  - XML Injection
  - XPath Injection
  - Command Injection
- All input validation flaws are caused by unsanitized data flows between the front-end and the several back-ends of a web application
- Anyway, we still miss something here !?!
  - _ _ _ Injection

An unbelievable story…

- There is no formal definition of an injection triggered by query string delimiters
- As far as we know, no one has never formalized an injection based attack against delimiters of the most used protocol on the web: HTTP
- HPP is surely around since many years, however it is definitely underestimated
- As a result, several vulnerabilities have been discovered in real-world applications

Introduction 1/2

- The term *Query String* is commonly used to refer to the part between the “?” and the end of the URI
- As defined in the RFC 3986, it is a series of field-value pairs
- Pairs are separated by “&” or “;”
- The usage of semicolon is a W3C recommendation in order to avoid escaping
- RFC 2396 defines two classes of characters:
  - *Unreserved:* a-z, A-Z, 0-9 and _ . ! ~ * ' ( )
  - *Reserved:* ; / ? : @ & = + $,

Introduction 2/2

- GET and POST HTTP request

```
GET /foo?par1=val1&par2=val2 HTTP/1.1
User-Agent: Mozilla/5.0
Host: Host
Accept: */*
```

```
POST /foo HTTP/1.1
User-Agent: Mozilla/5.0
Host: Host
Accept: */*
Content-Length: 19

par1=val1&par2=val2c
```

- Query String meta characters are &, ?, #, ; , = and equivalent (e.g. using encoding)
- In case of multiple parameters with the same name, HTTP back-ends behave in several ways

Server enumeration - List

Technology/HTTP back-end	Overall Parsing Result	Example
ASP.NET/IIS	All occurrences of the specific parameter	par1=val1,val2
ASP/IIS	All occurrences of the specific parameter	par1=val1,val2
PHP/Apache	Last occurrence	par1=val2
PHP/Zeus	Last occurrence	par1=val2
JSP,Servlet/Apache Tomcat	First occurrence	par1=val1
JSP,Servlet/Oracle Application Server 10g	First occurrence	par1=val1
JSP,Servlet/Jetty	First occurrence	par1=val1
IBM Lotus Domino	Last occurrence	par1=val2
IBM HTTP Server	First occurrence	par1=val1
mod_perl,libapreq2/Apache	First occurrence	par1=val1
Perl CGI/Apache	First occurrence	par1=val1
mod_perl,lib???/Apache	Becomes an array	ARRAY(0x8b9059c)
mod_wsgi (Python)/Apache	First occurrence	par1=val1
Python/Zope	Becomes an array	['val1', 'val2']
IceWarp	Last occurrence	par1=val2
AXIS 2400	All occurrences of the specific parameter	par1=val1,val2
Linksys Wireless-G PTZ Internet Camera	Last occurrence	par1=val2
Ricoh Aficio 1022 Printer	First occurrence	par1=val1
webcamXP PRO	First occurrence	par1=val1
DBMan	All occurrences of the specific parameter	par1=val1~~val2

Server enumeration - Summing up

- Different web servers manage multiple occurrences in several ways
- Some behaviors are quite bizarre
- Whenever protocol details are not <u>strongly</u> defined, implementations may <u>strongly</u> differ
- <u>Unusual</u> behaviors are a <u>usual</u> source of security weaknesses *(MANTRA!)*

Additional considerations 1/2

- As mentioned, ASP and ASP.NET concatenate the values with a comma in between
- This applies to the Query String and form parameters in ASP and ASP.NET
  - *Request.QueryString*
  - *Request.Form*
- Cookies have similar property in ASP.NET
  - *Request.Params["par"]*
  - *par = 1,2,3,4,5,6*

```
POST /index.aspx?par=1&par=2 HTTP/1.1
User-Agent: Mozilla/5.0
Host: Host
Cookie: par=5; par=6
Content-Length: 19

par=3&par=4
```

Additional considerations 2/2

- Unfortunately, application behaviors in case of multiple occurrences may differ as well
- This is strongly connected with the specific API used by our code
- In Java, for example:
  - *javax.servlet.ServletRequest Interface* (Query String direct parsing)
  - *java.lang.String* ***getParameter****(java.lang.String name)*

    Returns the value of a request parameter as a String, or null if the parameter does not exist
  - *java.lang.String[]* ***getParameterValues****(java.lang.String name)*

    Returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist

- As a result, the applications may react in unexpected ways...as you will see!

A bizarre behavior 1/4 - HPP*ed !*

Help - CUPS 1.3.9 - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://127.0.0.1:631/help/?QUERY=compatibility&QUERY=overview

UNIX PRINTING SYSTEM **Help**

Home | Administration | Classes | Documentation/Help | Jobs | Printers

Search in All Documents: compatibility

Search Results in All Documents:

- **Overview of CUPS**

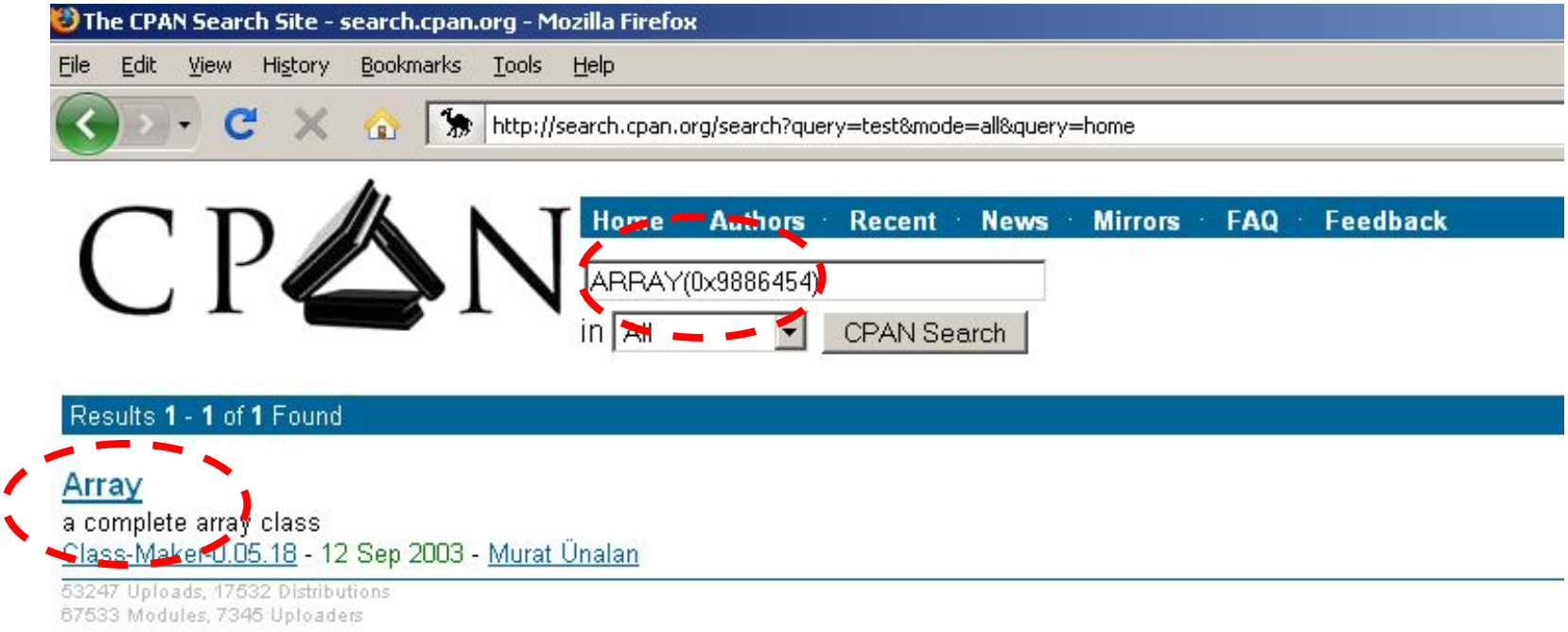
CUPS Help Pages

This is the CUPS on-line help interface. Enter search words above or click on any of the documentation links display on-line help information.

If you are new to CUPS, read the "**Overview of CUPS**" page. Veteran users should read the "**What's New CUPS 1.3**" page.

The **CUPS Home Page** also provides many resources including user discussion forums, answers to frequently-asked questions, and a form for submitting bug reports and feature requests.

A bizarre behavior 2/4 - HPP*ed !*



A bizarre behavior 3/4 - HPP*ed !*

Database Manager Demo: Login Error. - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://127.0.0.1/cgi-bin/dbman/db.cgi?db=default&uid=aaaa&uid=bbbb

Database Manager Demo: Login Error

Log On Error

Oops, there was a problem logging into the system: **invalid/expired user session**.

Please try logging in again, or contact the system administrator.

User ID:

Password:

Logon Cancel

DBMan encountered an internal error.

```
CGI ERROR
==========================================
Error Message        : Debug Information
Script Location      : /var/www/cgi-bin/dbman/db.cgi
Perl Version         : 5.010000
Setup File           : default.cfg
Session ID           : aaaa~~bbbb

Form Variables
------------------------------------------
db                   : default
uid                  : aaaa~~bbbb

Environment Variables
------------------------------------------
DOCUMENT_ROOT        : /var/www/
GATEWAY_INTERFACE    : CGI/1.1
```

A bizarre behavior 4/4 - HPPed !

Portalaa - Mozilla Firefox

File Modifica Visualizza Cronologia Segnalibri Strumenti Guida

http://localhost:8080/Plone/search?SearchableText=aaaa;SearchableText=bbbb

Più visitati Getting Started Latest Headlines

Site Map Accessibility Contact

['aaaa', 'bbbb'] Search

Only in current section

Home Users News Events

Log in

You are here: Home

We're sorry, but there seems to be an error...

The error has been logged as entry number 1241443579.40.25786329

If you need to report this to the site administration, please include this entry numb

The Plone® CMS — Open Source Content

Plone® and the Plone logo are registered tradem

Powered by Plone

plone@plone-ruggine: ~

File Modifica Visualizza Terminale Schede Aiuto

```
        'root': <Application at >,
        'template': <FSPageTemplate at /Plone/search>,
        'traverse_subpath': [],
        'user': <SpecialUser 'Anonymous User'>}
  Module Products.PageTemplates.ZRPythonExpr, line 49, in __call_
   - __traceback_info__: here.queryCatalog(REQUEST=request,use_ty
se_types_blacklist, use_navigation_root=use_navigation_root)
  Module PythonExpr, line 1, in <expression>
  Module Products.CMFCore.FSPythonScript, line 140, in __call__
  Module Shared.DC.Scripts.Bindings, line 313, in __call__
  Module Shared.DC.Scripts.Bindings, line 350, in _bindAndExec
  Module Products.CMFCore.FSPythonScript, line 196, in _exec
  Module None, line 72, in queryCatalog
   - <FSPythonScript at /Plone/queryCatalog>
   - Line 72
  Module None, line 45, in quote_bad_chars
   - <FSPythonScript at /Plone/queryCatalog>
   - Line 45
AttributeError: 'list' object has no attribute 'replace'
```

- Since this error generates ~100 lines in the log file, it may be used to obfuscate other attacks

HPP in a nutshell

- **HTTP Parameter Pollution** (**HPP**) is a quite simple but effective hacking technique
- HPP attacks can be defined as the feasibility to override or add HTTP GET/POST parameters by injecting query string delimiters
- It affects a building block of all web technologies thus server-side and client-side attacks exist
- Exploiting HPP vulnerabilities, it may be possible to:
  - Override existing hardcoded HTTP parameters
  - Modify the application behaviors
  - Access and, potentially exploit, uncontrollable variables
  - Bypass input validation checkpoints and WAFs rules

HPP Categories

- We are not keen on inventing yet another buzzword. However, the standard vulnerability nomenclature seems lacking this concept

- Classification:
  - Client-side
    1. *First order HPP or Reflected HPP*
    2. *Second order HPP or Stored HPP*
    3. *Third order HPP or DOM Based HPP*
  - Server-side
    1. *Standard HPP*
    2. *Second order HPP*

- According to our classification, *Flash Parameter Injection** may be considered as a particular subcategory of the HPP client-side attack

* http://blog.watchfire.com/FPI.ppt

Encoding & GET/POST/Cookie precedence

- Several well-known encoding techniques may be used to inject malicious payloads

Encoding Type	Value
URL Encode	*%26*
Double URL Encode	*%2526*
UTF-8 (2 bytes)	*%c0%a6*
UTF-8 (Java style)	*\uc0a6*
HTML Entity	*&*
HTML Entity number	*&*
Unicode URL Encode	*%u0026*

- The precedence of GET/POST/Cookie may influence the application behaviors and it can also be used to override parameters

Apache Tomcat/6.0.18

POST /foo?par1=val1&par1=val2 HTTP/1.1
Host: 127.0.0.1

par1=val3&par1=val4

FIRST occurrence, GET parameter first

HPP Server Side Attacks 1/2

- Suppose some code as the following:

```
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);
}
```

- Which is the attack surface?

HPP Server Side Attacks 2/2

- A malicious user may send a request like:

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

- Then, the frontend will build the following back-end request:

```
HttpRequest("http://backendServer.com/servlet/actions","POST",
      "action=transfer&amount="+amount+"&recipient="+beneficiary);
```

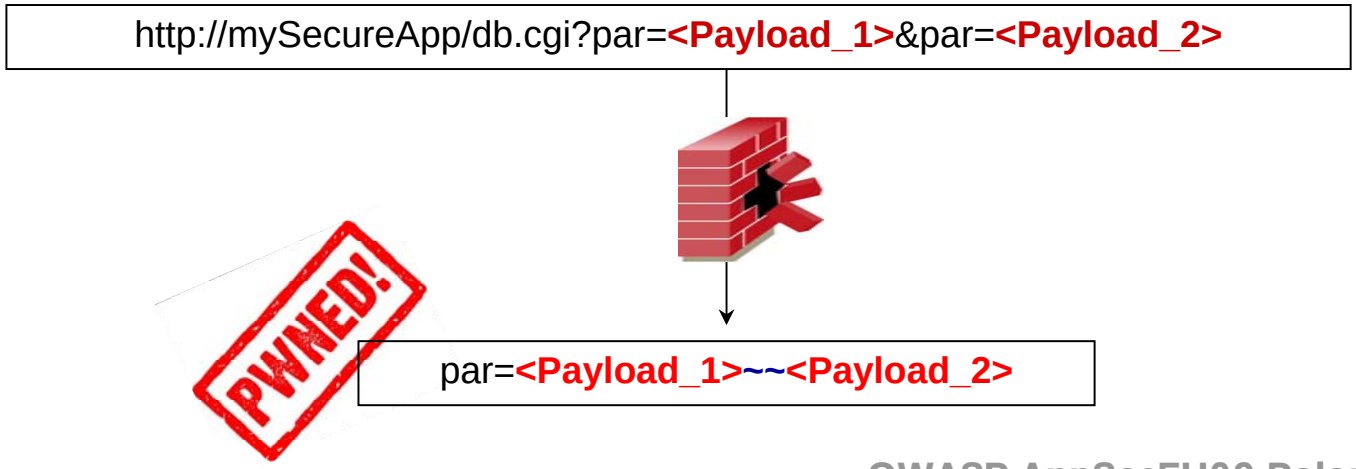
↓

action=transfer&amount=1000&recipient=Mat**&action=withdraw**

- Obviously depends on how the application will manage the occurrence

HPP Server Side - WebApp Firewalls

- What would happen with WAFs that do Query String parsing before applying filters?
- HPP can be used even to bypass WAFs ☺
- Some loose WAFs may analyze and validate a single parameter occurrence only (first or last one)
- Whenever the devel environment concatenates multiple occurrences (e.g. ASP, ASP.NET, AXIS IP Cameras, DBMan, ...), an aggressor can split the malicious payload.



HPP Server Side – URL Rewriting

- URL Rewriting could be affected as well if regexp are too permissive:

```
RewriteCond %{THE_REQUEST} ^[A-Z]{3,9}\ .+page\.php.*\ HTTP/
RewriteRule ^page\.php.*$ - [F,L]

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^([^/]+)$ page.php?action=view&page=$1&id=0 [L]
```

http://host/abc

becomes:

http://host/page.php?action=view&page=abc&id=0

HPP Server Side – URL Rewriting issues

- An attacker may try to inject:

  http://host/abc%26action%3dedit

- and the url will be rewritten as:

  http://host/page.php?action=view&page=abc&action=edit&id=0

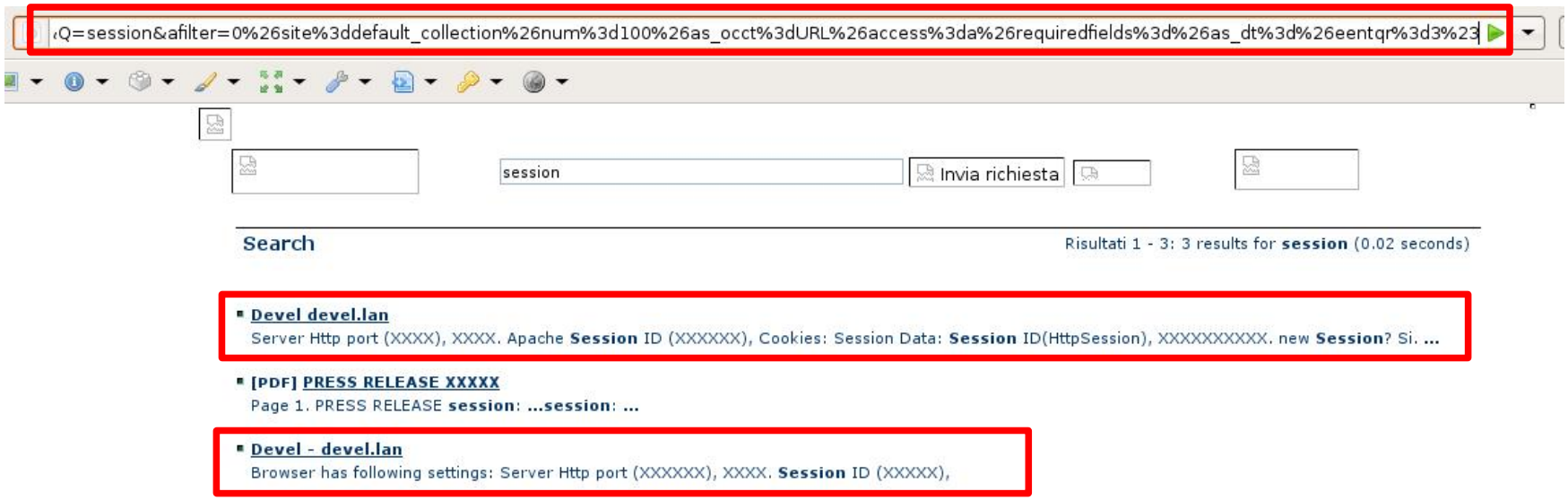
- Obviously, the impact depends on the functionality exposed

Real World Examples

Server Side Attacks

Google Search Appliance - HPP*ed !*

- Once upon a time, during an assessment for XXX...
- GSA was the LAN search engine exposed for public search as well, with only three controllable values
- The parameter named “afilter” is used unencoded
- By polluting GSA parameters, appending %23 (“#”), we got full access to internal results



ModSecurity - HPP*ed !*

- *ModSecurity SQL Injection filter bypass*
- While the following query is properly detected

/index.aspx?page=select 1,2,3 from table where id=1

- Using HPP, it is possible to bypass the filter

/index.aspx?page=select 1&page=2,3 from table where id=1

- Other vendors may be affected as well
- This technique could potentially be extended to obfuscate attack payloads
- Lavakumar Kuppan is credited for this finding

HPP Client Side attacks 1/2

- HPP Client Side is about injecting additional parameters to links and other src attributes
- Suppose the following code:

```
<? $val=htmlspecialchars($_GET['par'],ENT_QUOTES); ?>
<a href="/page.php?action=view&par='.<?=$val?>.'">View Me!</a>
```

- There's no XSS, but what about HPP?
- It’s just necessary to send a request like

```
http:/host/page.php?par=123%26action=edit
```

- To obtain

```
<a href="/page.php?action=view&par=123&amp;action=edit">View Me!</a>
```

HPP Client Side attacks 2/2

- Once again, it strongly depends on the functionalities of a link
- It's more about
  - Anti-CSRF
  - Functional UI Redressing
- It could be applied on every tag with
  - Data, src, href attributes
  - Action forms with POST method

HPP Client Side - DOM based

- It's about parsing unexpected parameters
- It's about the interaction between IDSs and the application
- It's about the generation of client side HPP via JavaScript
- It's about the use of (XMLHttp)Requests on polluted parameters

```
// First Occurrence
function gup( name )
{
  name = name.replace(/[\[]/,"\\\[").replace(/[\]]/,"\\\]");
  var regexS = "[\\?&]"+name+"=([^&#]*)";
  var regex = new RegExp( regexS );
  var results = regex.exec( window.location.href );
  if( results == null )
    return "";
  else
    return results[1];
}
```

```
// Last Occurrence
function argToObject () {
 var sArgs = location.search.slice(1).split('&');
 var argObj={};
 for (var i = 0; i < sArgs.length; i++) {
  var r=sArgs[i].split('=')
  argObj[r[0]]=r[1]
   }
 return  argObj
}
```

HPP Client Side - FPI, the HPP way

- As mentioned, an interesting case of HPP is the *Flash Parameter Injection* by Ayal Yogev and Adi Sharabani @ Watchfire
- FPI is about including *FlashVars* in the html itself when the vulnerable flash is directly dependent on the page itself
- A FPI will result in the injection of additional parameters in the *param* tag
- E.g. Piggybacking FlashVars

http://myFlashApp/index.cgi?language=ENG%26globalVar=**<HPP>**

Real World Examples

Client Side Attacks

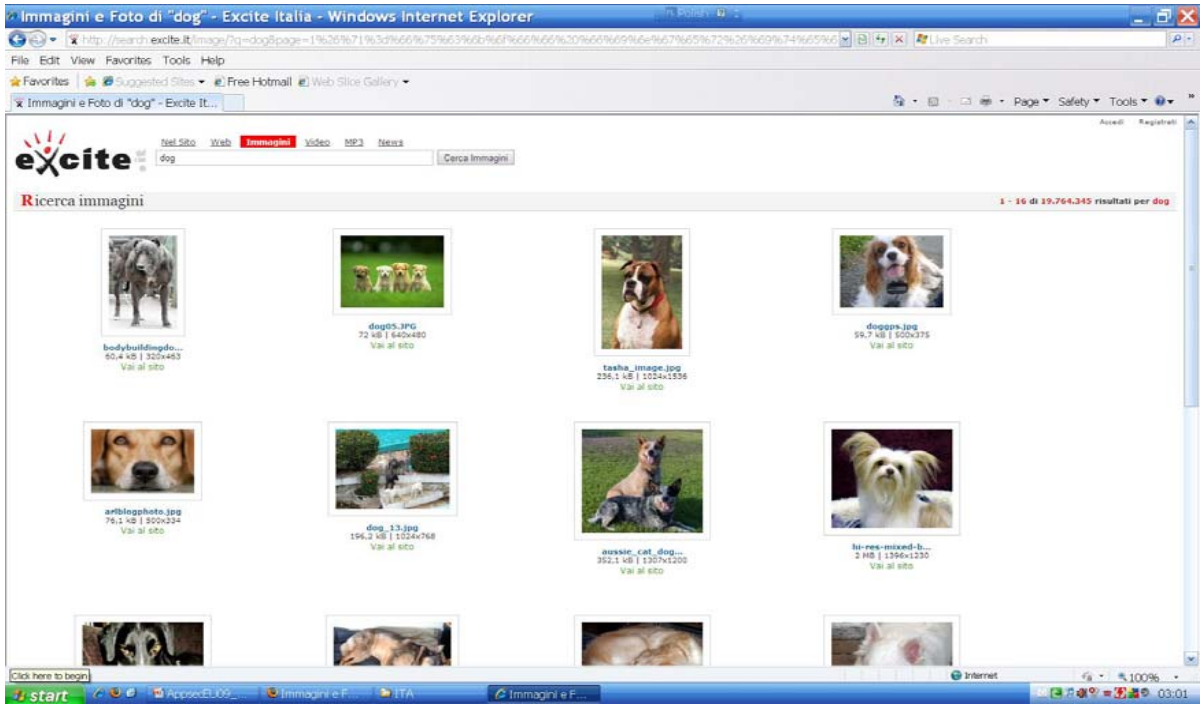
Ask.com - HPP*ed !*

- Features:
  - Anti XSS using HtmlEntities
  - DOM HPP and Client Side HPP compliant! ;)

Excite - HPP*ed !*

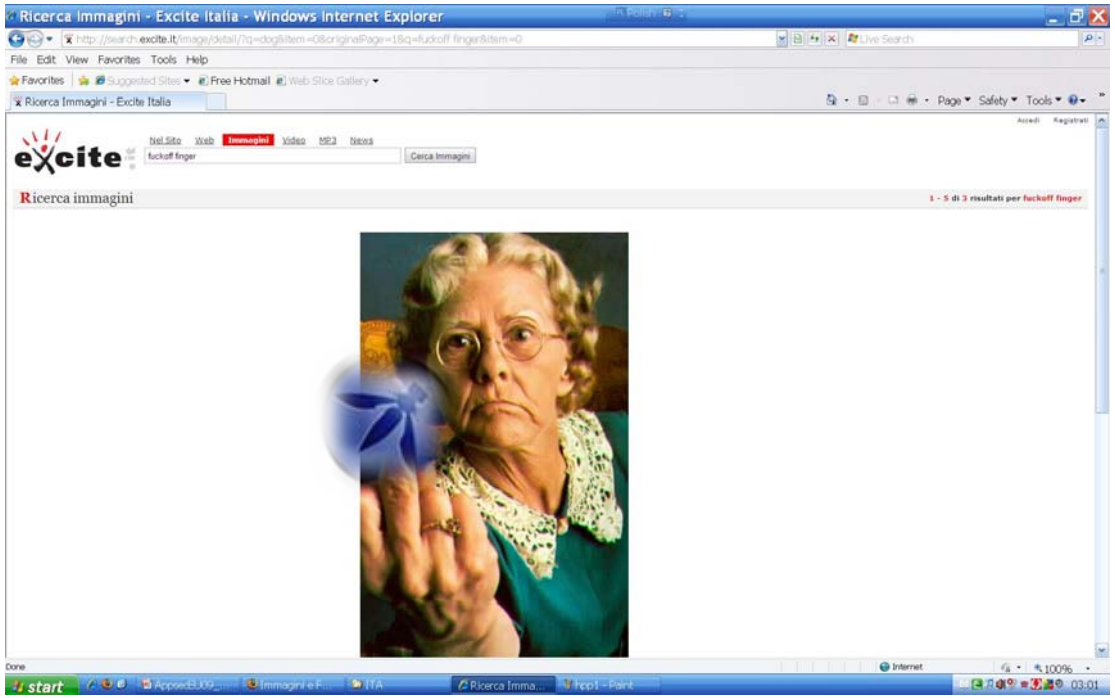
- Features:
  - Several parameters could be HPPed
  - Anti XSS using htmlEntities countermeasures
  - DOM HPP + Client Side HPP friendly!

http://search.excite.it/image/?q=dog&page=1%26%71%3d%66%75%63%6b%6f%66%66%20%66%69%6e%67%65%72%26%69%74%65%6d%3d%30



Excite - HPP*ed !*

- Sweet dogs? Click anywhere on an image...



- This is a kind of content pollution
- Even if the example seems harmless, it may help to successfully conduct social engineering attacks

MS IE8 XSS Filter Bypass - HPP*ed !*

- IE8 checks for XSS regexp in the query string parameters, as well as it searches for them in the output
- When there's a .NET application, multiple occurrences of a parameter are joined using “,”
- So *param=<script&param=src=”....”>* becomes *<script,src=”...”>* in HTML
- As you can imagine, it bypasses the IE8 XSS filter
- Alex Kuza is credited for this finding

Yahoo! Mail Classic - HPP*ed !*

- Features
  - Check antiCSRF
  - Dispatcher View
  - Html Entities filtering, antiXSS
  - HPP compliant!
- The dispatcher pattern helps the attacker
  - %26DEL=1%26DelFID=Inbox%26cmd=fmgt.delete
  - %2526cmd=fmgt.emptytrash
  - Attack payload: **http://it.mc257.mail.yahoo.com/mc/showFolder?fid=Inbox&order=down&tt=245&pSize=25&startMid=0%2526cmd=fmgt.emptytrash%26DEL=1%26DelFID=Inbox%26cmd=fmgt.delete**

Yahoo! Mail Classic - HPP*ed !*

- It’s show time!

- Yahoo! has (silently) patched this issue...

PTK Forensic - HPP*ed !*

- *PTK, an alternative Sleuthkit Interface*
- PTK is a forensic tool with a web based frontend written in PHP, included in the SANS SIFT
- The investigator can mount a DD image and then inspect files, using the Web2.0 UI
- Here, HPP is the key to exploit a critical vulnerability*

"...Once the investigator selects a specific file from the image filesystem, PTK invokes the following script:
/ptk/lib/file_content.php?arg1=null&arg2=107533&arg3=<FILENAME>&arg4=1
..."

* http://www.ikkisoft.com/stuff/LC-2008-07.txt

PTK Forensic - HPP*ed !*

- Vulnerable code:

```
$offset = $_GET['arg1'];
$inode = $_GET['arg2'];
$name = $_GET['arg3']; //filename
$partition_id = $_GET['arg4'];
$page_offset = 100;
...
$type = get_file_type($_SESSION['image_path'], $offset, $inode);
...

function get_file_type($path, $offset, $inode){
        include("../config/conf.php");
        if($offset == 'null'){
                $offset = '';
        }else{
                $offset = "-o $offset";
        }
        if($inode == 'null') $inode = '';
        $result = shell_exec("$icat_bin -r $offset $path $inode | $file_bin
-zb -");
        if(preg_match("/(image data)|(PC bitmap data)/", $result)){
                $_SESSION['is_graphic'] = 1;
        }       return $result;}
```

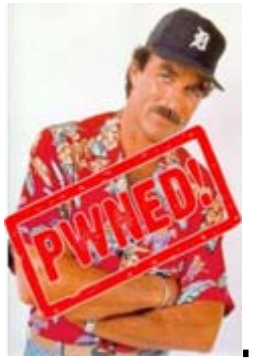
- Since filenames are contained within the DD image, they should be considered as user-supplied values

PTK Forensic - HPP*ed !*

- Crafting a filename as **Confidential.doc&arg1=;EvilShell;...**
- It is actually possible to tamper the link, leading to code execution since PHP considers the last occurrence

.../file_content.php?arg1=null&arg2=107533&arg3=**Confidential.doc&arg1=;EvilShell;...**&arg4=1

- Demonstration video of the attack: *http://www.vimeo.com/2161045*

As a result... 

...Stored HPP!

PHPIDS - HPP*ed !*

- PHPIDS is a state-of-the-art security layer for PHP web applications
- When dealing with DOM based HPP, PHPIDS could be fooled
- If the DOM based location parsing gets the first occurrence, then PHPIDS will consider only PHP behavior
- It means the last occurrence, thus no alert and XSS attacks still possible!

Countermeasures

- Speaking about HPP, several elements should be considered:
  - Application business logic
  - Technology used
  - Context
  - Data validation (as usual!)
  - Output encoding
- Filtering is the key to defend our systems!
- Don't use HtmlEntities. They're out of context!
- Instead, apply URL Encoding
- Use strict regexp in URL Rewriting
- Know your application environment!

Conclusion

- HPP is a quite simple but effective hacking technique
- HPP affects server side as well client side components
- The impact could vary depending on the affected functionality

- We are going to release a whitepaper about these and other issues, including all technical details. *Stay tuned!*
- HPP requires further researches in order to deeply understand threats and risks. Several applications are likely vulnerable to HPP

- Standard and guidelines on multiple occurrences of a parameter in the QueryString should be defined
- Awareness for application developers is crucial

Q&A

- Time is over! Thanks!

- If you have further inquiries, please contact us:
  - luca.carettoni@ikkisoft.com
  - stefano.dipaola@mindedsecurity.com