Automated Detection of HPP Vulnerabilities in Web Applications

Marco `embyte` Balduzzi

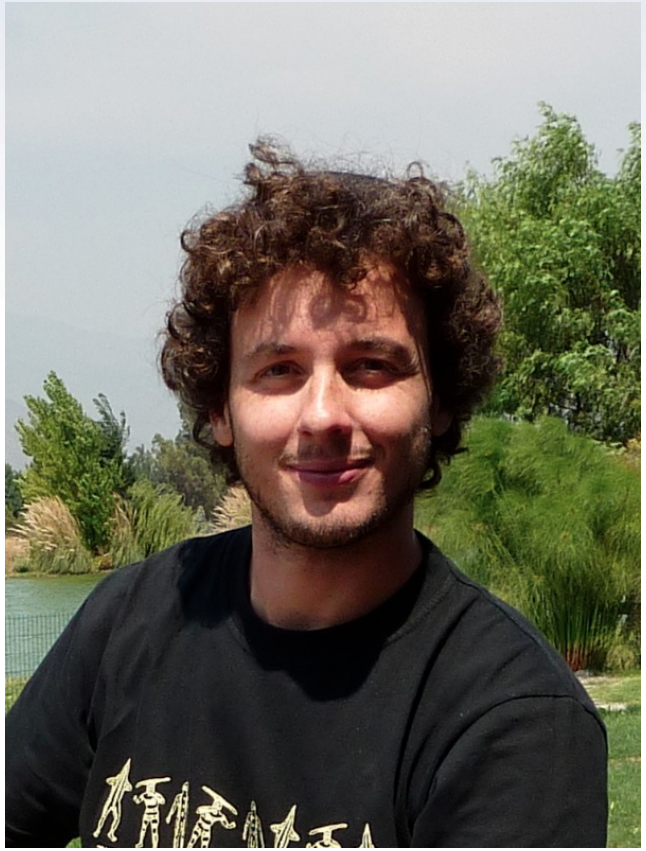
Roadmap

- Introduction
- HTTP Parameter Pollution
  - Client-Side
  - Server-Side
  - Other Uses
- Detection
  - Approach
  - Tool
  - Demo
- Experiments
- Results
- Prevention

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…

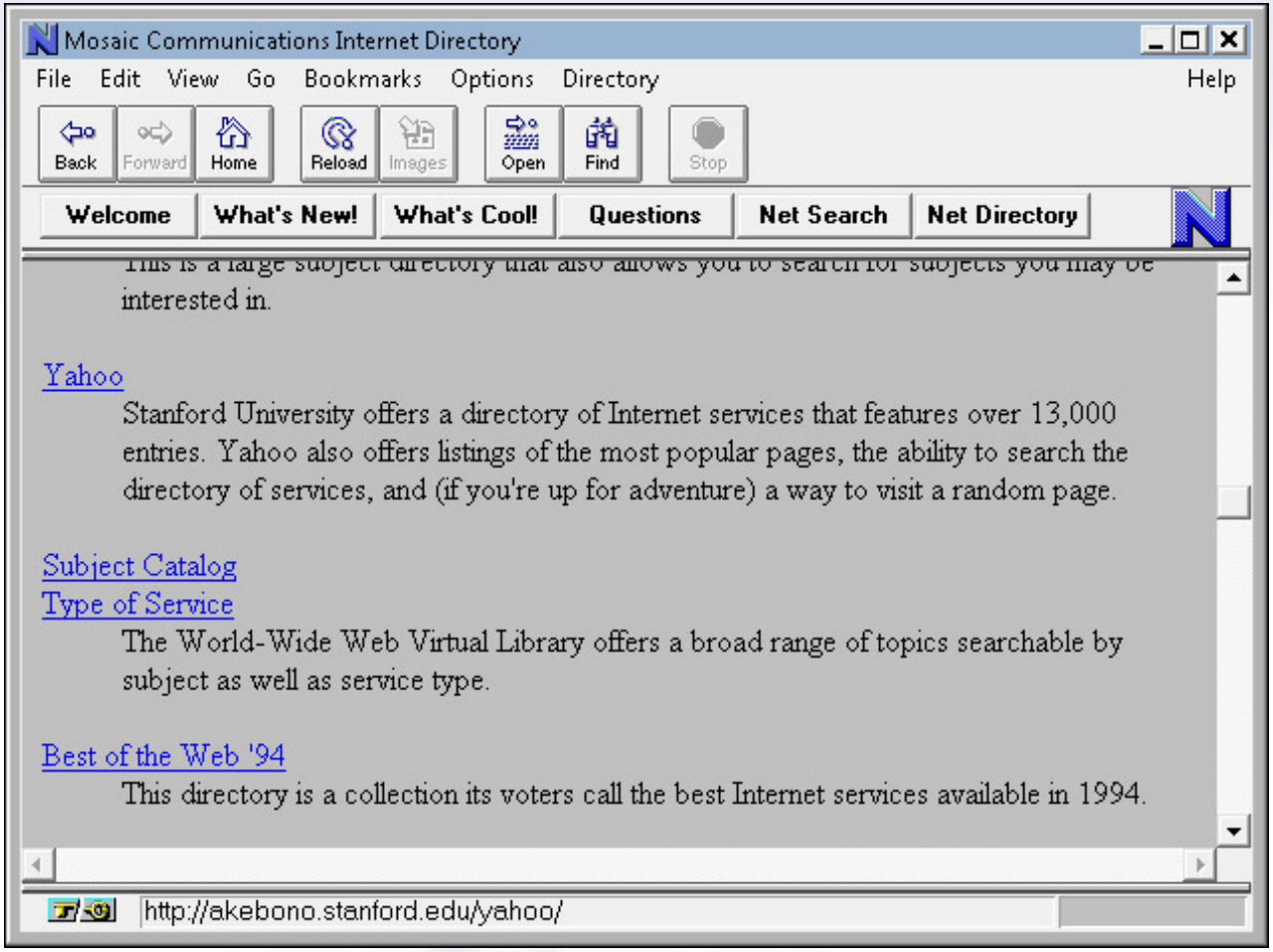
- http://www.iseclab.org/people/embyte



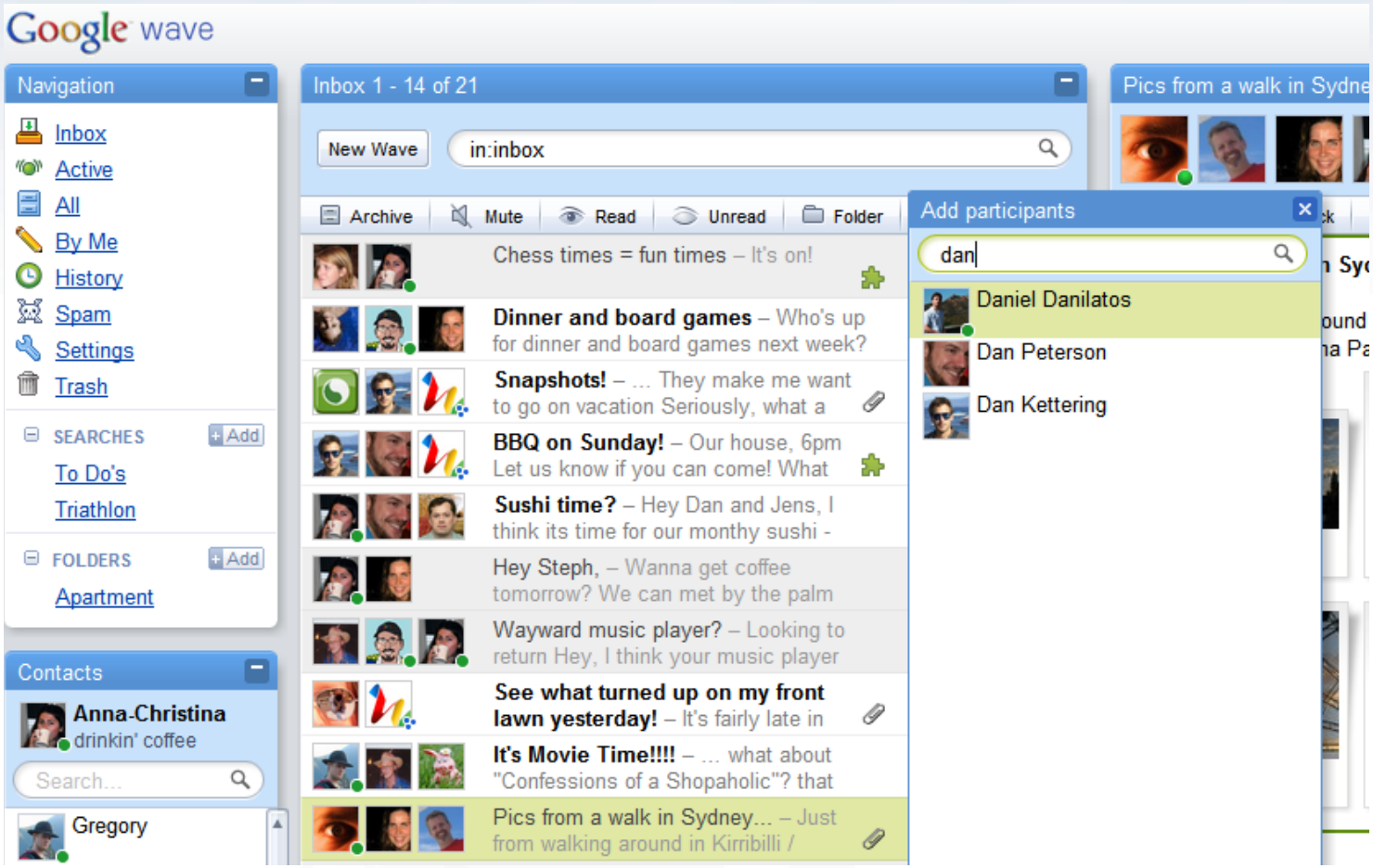
The Web as We Know It

- 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



Now



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

Research Objectives

- To create the first automated system for detecting HPP flaws
  - Blackbox approach, consists of a set of tests and heuristics
- To find out how prevalent HPP problems are on the web
  - Is the problem being exaggerated?
  - Is this problem known by developers?
  - Does this problem occur more in smaller sites than larger sites?
  - What is the significance of the problem?

Roadmap

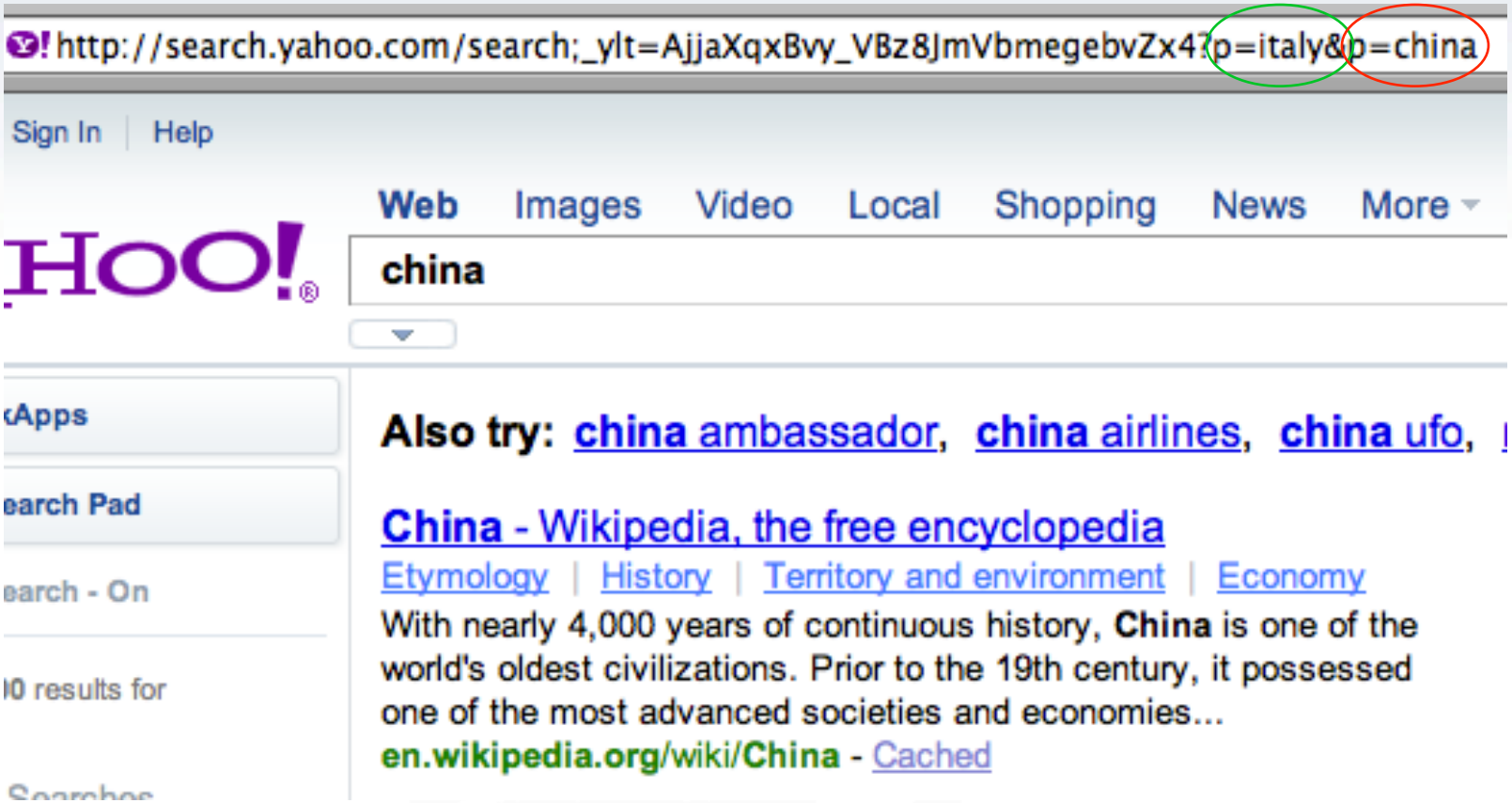
- Introduction
- HTTP Parameter Pollution
  - Client-Side
  - Server-Side
  - Other Uses
- Detection
  - Approach
  - Tool
  - Demo
- Experiments
- Results
- Prevention

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

Yahoo example



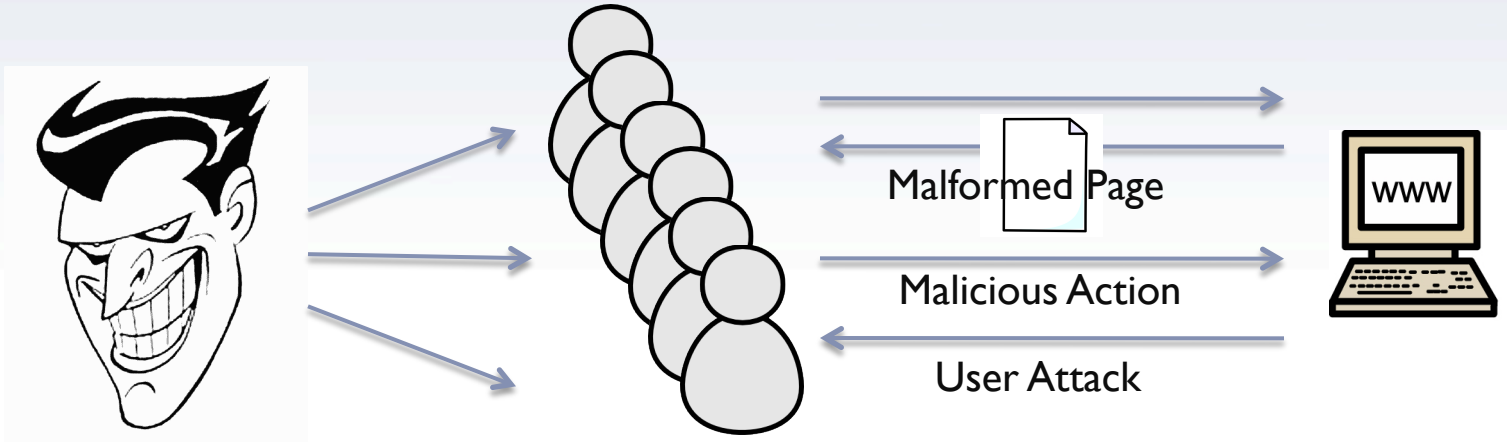
HTTP Parameter Handling

- We manually tested common methods of 5 different languages

Technology/Server	Tested Method	Parameter Precedence
ASP/IIS	Request.QueryString("par")	All (comma-delimited string)
PHP/Apache	$_GET("par")	Last
JSP/Tomcat	Request.getParameter("par")	First
Perl(CGI)/Apache	Param("par")	First
Python/Apache	getvalue("par")	All (List)

- 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*

Trigger URL are
sent to the victims

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

Site vulnerable to
HTTP Parameter Pollution

Client-Side #1: 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(“pool_id”)
href_link = “vote.jsp?poll_id=” + ID + ”&candidate=xyz”
```

Client-Side #1: Unintended voting

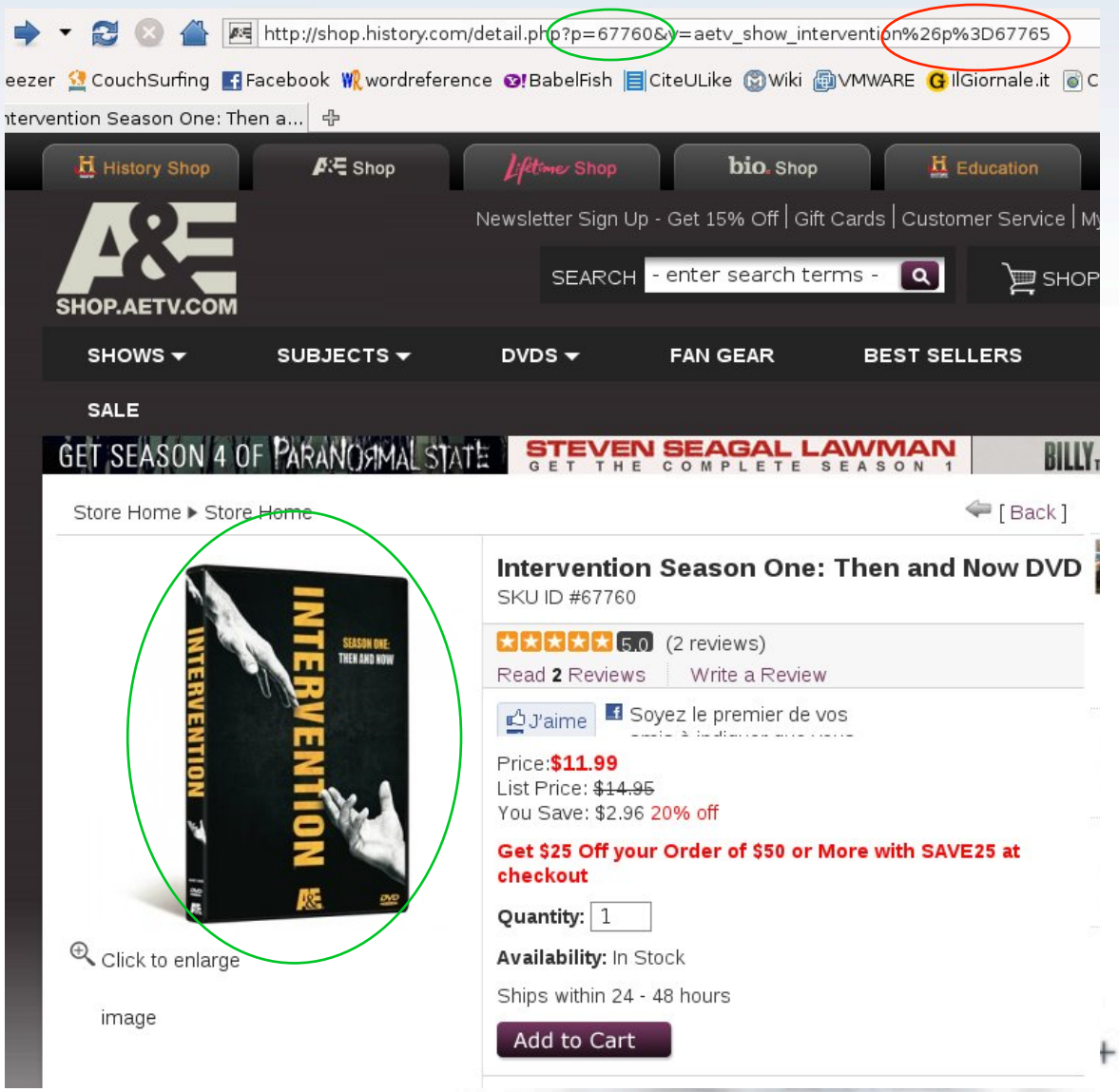
- poll_id is vulnerable
- Attacker generate a *Trigger URL* to be sent to his victims:
  - http://host/election.jsp?poll_id=4568%26candidate%3Dgreen

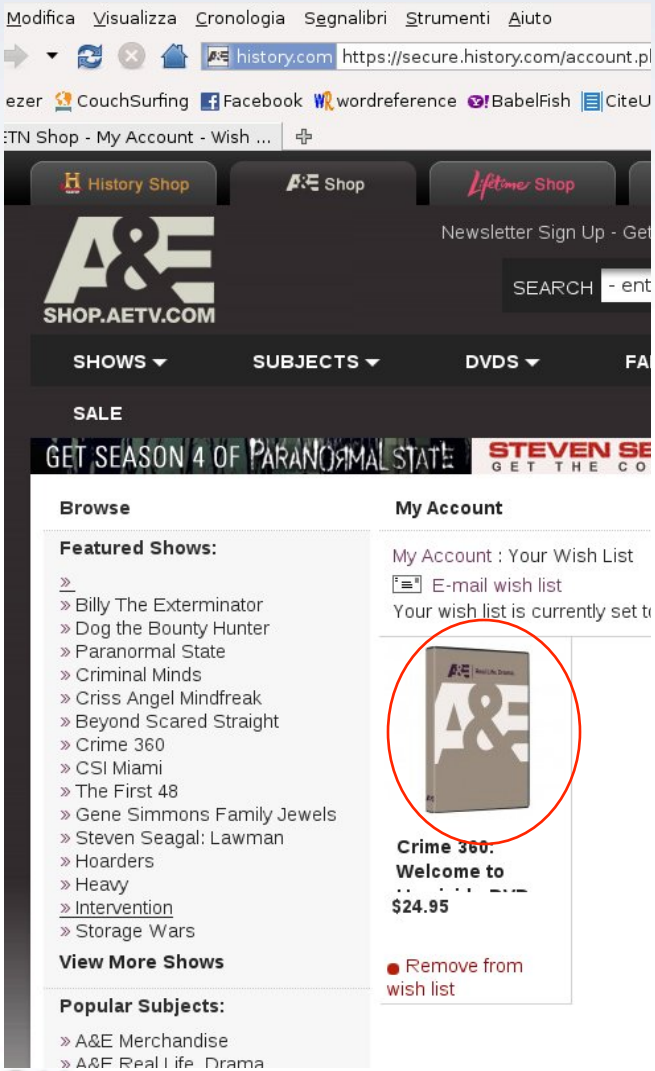
- The resulting page now contains injected links:

```
<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!

Client-Side #2: Misleading shopping users

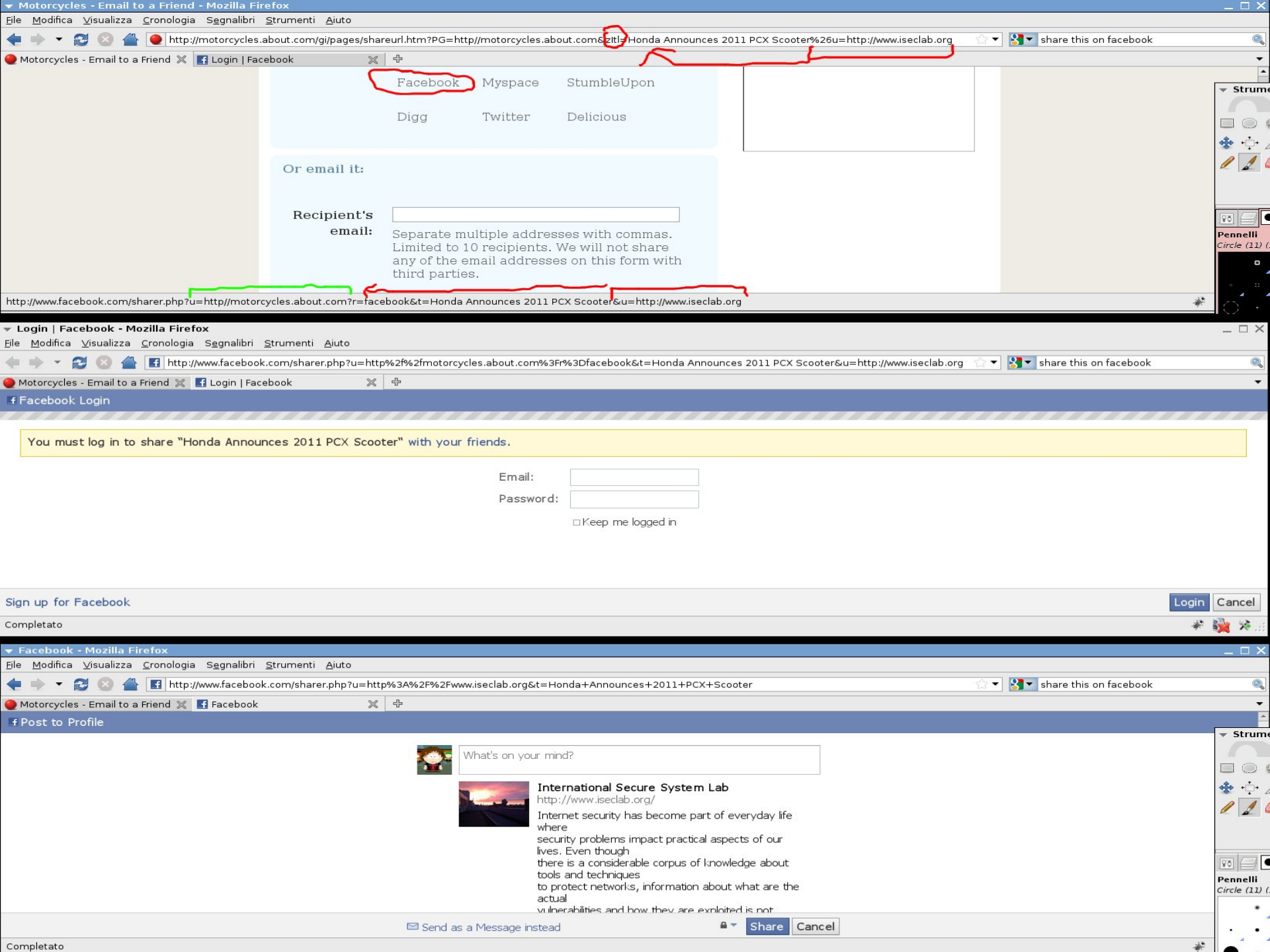




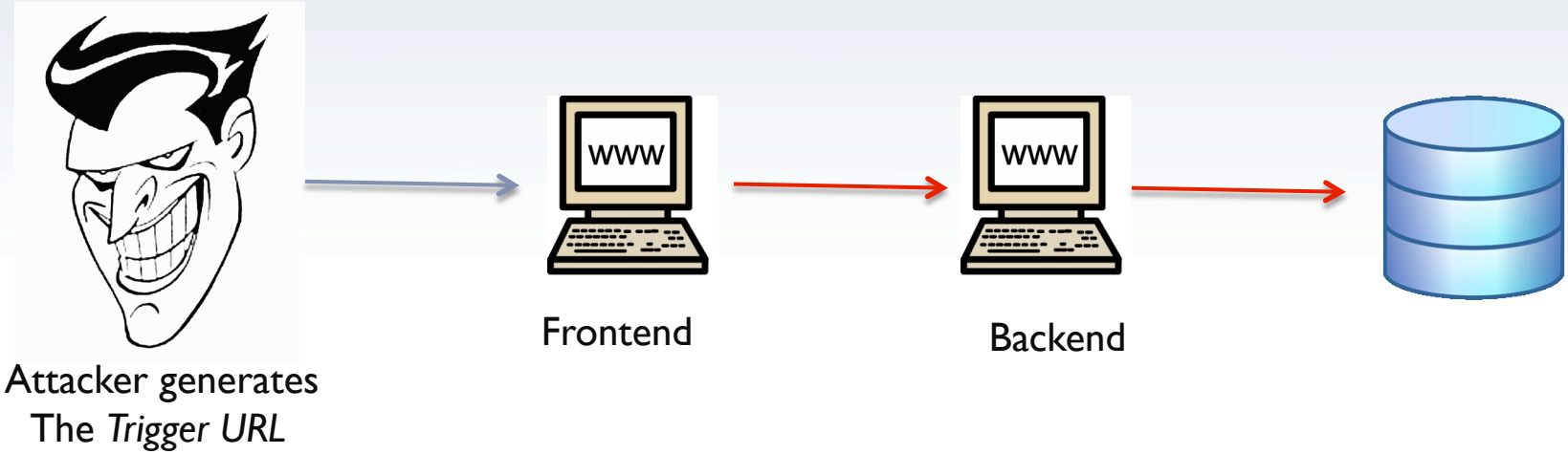
+ 2011

Client-Side #3: Sharing components

- Sharing functionalities can be attacked
- No validation in the sharer API (Facebook, Twitter, …)
- Injection on the customer side (e.g. blog post)
- Client-side attack
  - Posting of unintended data


Motorcycles - Email to a Friend - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://motorcycles.about.com/gi/pages/shareurl.htm?PG=http//motorcycles.about.com&ztl=Honda Announces 2011 PCX Scooter%26u=http://www.iseclab.org
share this on facebook
Motorcycles - Email to a Friend
Login | Facebook
Facebook
Myspace
StumbleUpon
Digg
Twitter
Delicious
Or email it:
Recipient's email:
Separate multiple addresses with commas. Limited to 10 recipients. We will not share any of the email addresses on this form with third parties.
http://www.facebook.com/sharer.php?u=http//motorcycles.about.com?r=facebook&t=Honda Announces 2011 PCX Scooter&u=http://www.iseclab.org
Login | Facebook - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://www.facebook.com/sharer.php?u=http%2f%2fmotorcycles.about.com%3Fr%3Dfacebook&t=Honda Announces 2011 PCX Scooter&u=http://www.iseclab.org
share this on facebook
Motorcycles - Email to a Friend
Login | Facebook
Facebook Login
You must log in to share "Honda Announces 2011 PCX Scooter" with your friends.
Email:
Password:
Keep me logged in
Sign up for Facebook
Login
Cancel
Completato
Facebook - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://www.facebook.com/sharer.php?u=http%3A%2F%2Fwww.iseclab.org&t=Honda+Announces+2011+PCX+Scooter
share this on facebook
Motorcycles - Email to a Friend
Facebook
Post to Profile
What's on your mind?
International Secure System Lab
http://www.iseclab.org/
Internet security has become part of everyday life where
security problems impact practical aspects of our lives. Even though
there is a considerable corpus of knowledge about tools and techniques
to protect networks, information about what are the actual
vulnerabilities and how they are exploited is not
Send as a Message instead
Share
Cancel
Completato

HTTP Parameter Pollution (Server-Side)



- Used to exploit the server-side logic of the web-application
- The attacker sends the *Trigger URL* to the vulnerable application

Server-Side #1: Payment System

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

```
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);
```

Server-Side #2: Database hijacking

- 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
Back-end: dbconnect.asp?what=users&department=engineering&what=passwd
Database: select users,passwd from table where department=engineering
```

Server-Side #3: Authorization Bypass

- Google Blogger exploited by *Nir Goldshlager*
- Get administrator privilege over any blogger account
- Attacker uses the add authors functionality
  - The server checks the 1st blogid value but executes the 2nd blogid value of the attacker
- When the attacker is added as user to the victim's blogger, he raises his privileges to administrator

```
POST /add-authors.do HTTP/1.1

security_token=attacker_token&blogID=attacker_blogidvalue&
blogID=victim_blogidvalue&authorsList=attacker_email&ok=Invite
```

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

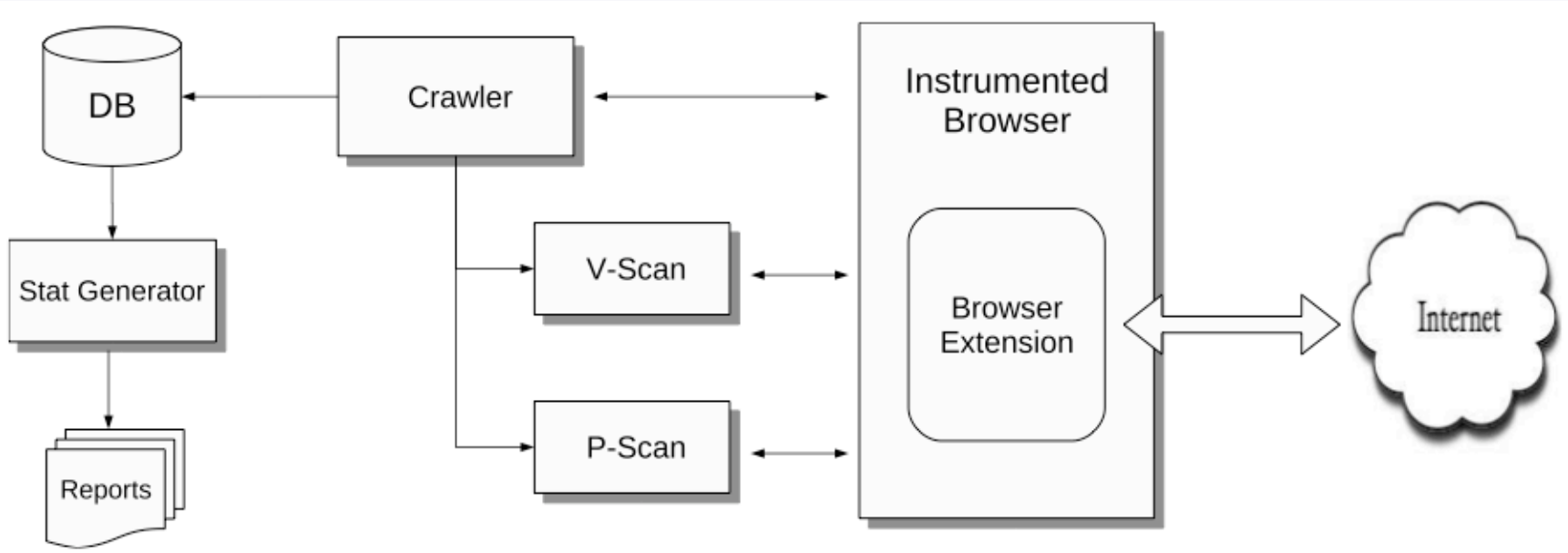
```
Standard: show_user.aspx?id=5;select+1,2,3+from+users+where+id=1-
Over HPP: show_user.aspx?id=5;select+1&id=2&id=3+from+users+where+id=1-

Standard: show_user.aspx?id=5+union+select+*+from+users-
Over HPP: show_user.aspx?id=5/*&id=*/union/*&id=*/select+*/*&id=*/from+users--
```

Roadmap

- Introduction
- HTTP Parameter Pollution
  - Client-Side
  - Server-Side
  - Other Uses
- Detection
  - Approach
  - Tool
  - Demo
- Experiments
- Results
- Prevention

System for HPP Detection

- Four main components: browser, crawler, two scanners



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
- How do we determine precedence? Naïve approach:
  - Page0==Page2 -> precedence on first parameter
  - Page1==Page2 -> precedence on second parameter

P-Scan: Problem with the naïve approach

- In practice, naïve technique does not work well
  - Applications are complex, much dynamic content (publicity banners, RSS feeds, ads, etc.)

- Hence, we perform pre-filtering to eliminate dynamic components (embedded content, applets, IFRAMES, style sheets, etc.)
  - Remove all self-referencing URLs (as these change when parameters are inserted)
  - We then perform different tests to determine similarity

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)
- $P_B = p \mid p \in P_{URL} \wedge p \notin P_{Body}$: URL parameters that do not appear in the page. Some of these parameters may appear in the page under a different name
- $P_C = p \mid p \notin P_{URL} \wedge p \in P_{Body}$: 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=p1&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%3fsession_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)

Implementation – The PAPAS tool

- PAPAS is fully customizable
  - E.g., scanning depth, number of performed injections, page loading timeouts, etc.
- Three modes are supported
  - Fast mode, extensive mode, assisted mode
  - In assisted mode, authenticated areas of a site can be scanned as well
- Now, as a free-to-use-service:
  - http://papas.iseclab.org

Possible improvements

- PAPAS does not support the crawling of links embedded in active content
  - E.g., flash
- Support additional encoding schemas (UTF-8, Double URL)
- PAPAS currently only focuses on client-side exploits where user needs to click on a link
  - HPP is also possible on the server side – but this is more difficult to detect
  - Analogous to detecting stored XSS

Roadmap

- Introduction
- HTTP Parameter Pollution
  - Client-Side
  - Server-Side
  - Other Uses
- Detection
  - Approach
  - Tool
  - Demo
- Experiments
- Results
- Prevention

Ethical Considerations

- Only client-side attacks. The server-side have the potential to cause harm
- We provided the applications with innocuous parameters (`&foo=bar`). No malicious code.
- Limited scan time (15min) and activity
- We immediately informed, when possible, the security engineers of the affected applications
  - Thankful feedbacks

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

Categories	# of Tested Applications	Categories	# of Tested Applications
Financial	110	Shopping	460
Games	300	Social Networking	117
Government	132	Sports	256
Health	235	Travel	175
Internet	698	University	91
News	599	Video	114
Organization	106	Others	1,401
Science	222		

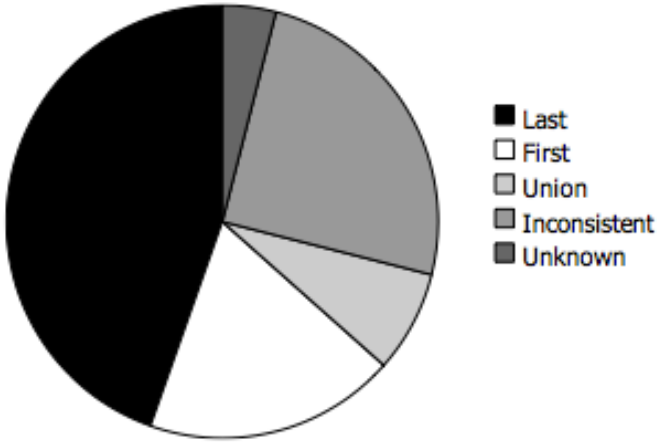
Efficient assessment

- 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 does not seem conscious of the possibility to duplicate GET/POST parameter
  - No sanitization is in place

Parameter Precedence	WebSites	
Last	2,237	(44.60%)
First	946	(18.86%)
Union	381	(7.60%)
Inconsistent	1,251	(24.94%)
Unknown	201	(4.00%)
Total	5,016	(100.00%)
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:

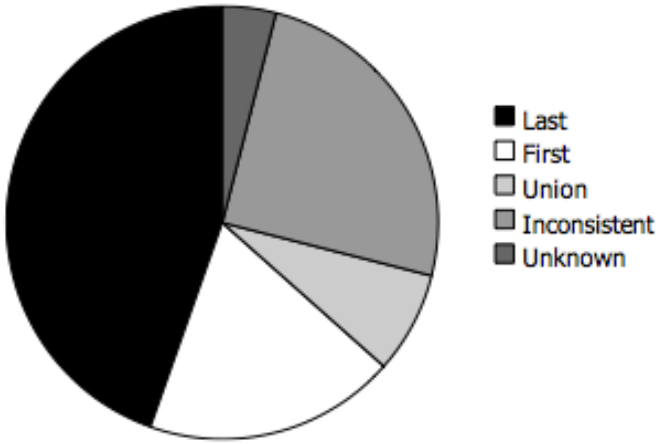
- Enable Robust Exception Information to provide greater detail about the source of errors. In the Administrator, click Debugging & Logging > Debug Output Settings, and select the Robust Exception Information option.
- Check the ColdFusion documentation to verify that you are using the correct syntax.
- 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

Parameter Precedence	WebSites	
Last	2,237	(44.60%)
First	946	(18.86%)
Union	381	(7.60%)
Inconsistent	1,251	(24.94%)
Unknown	201	(4.00%)
Total	5,016	(100.00%)
Database Errors	238	(4.74%)



Evaluation – HPP Vulnerabilities

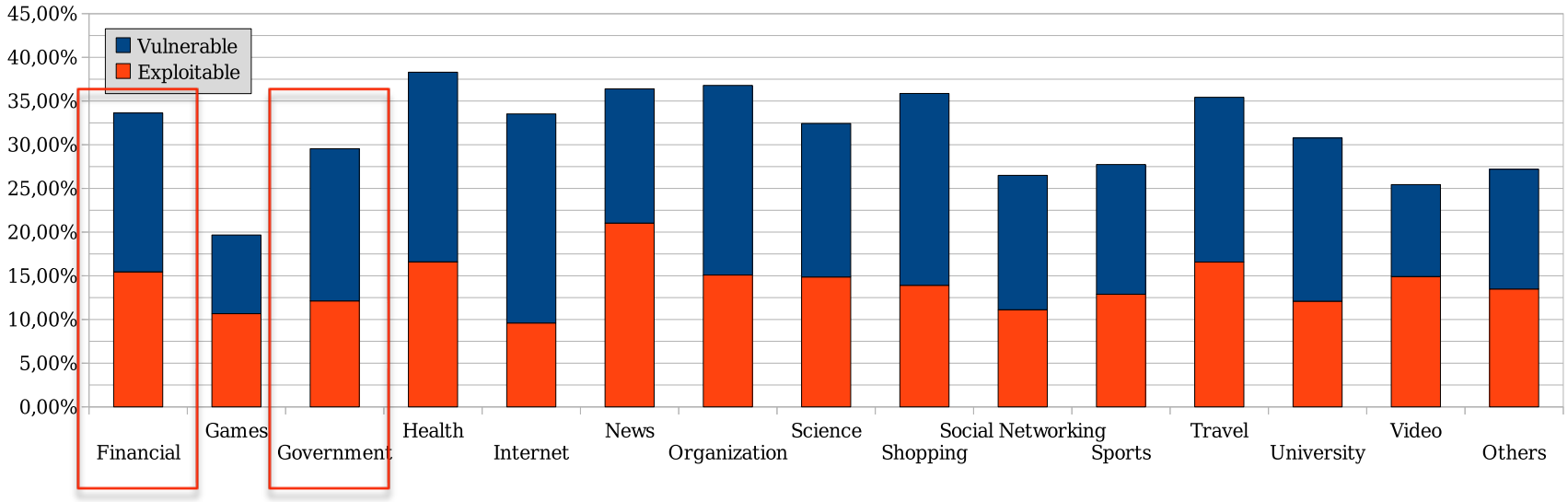
- PAPAS discovered that about 1,500 (30%) websites contained at least one page vulnerable to HTTP Parameter Injection
  - The tool was able to inject (and verify) an encoded parameter
- Vulnerable != Exploitable
  - Is the parameter precedence consistent?
  - Can a possible attacker override existing parameter values?

Vulnerable or Exploitable?

- Injection on link
- Read a mail: http://site.com/script?mail_id=10&action=read
  - 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%)

Evaluation



- More sensitive sites are equally (or even more) affected by the problem

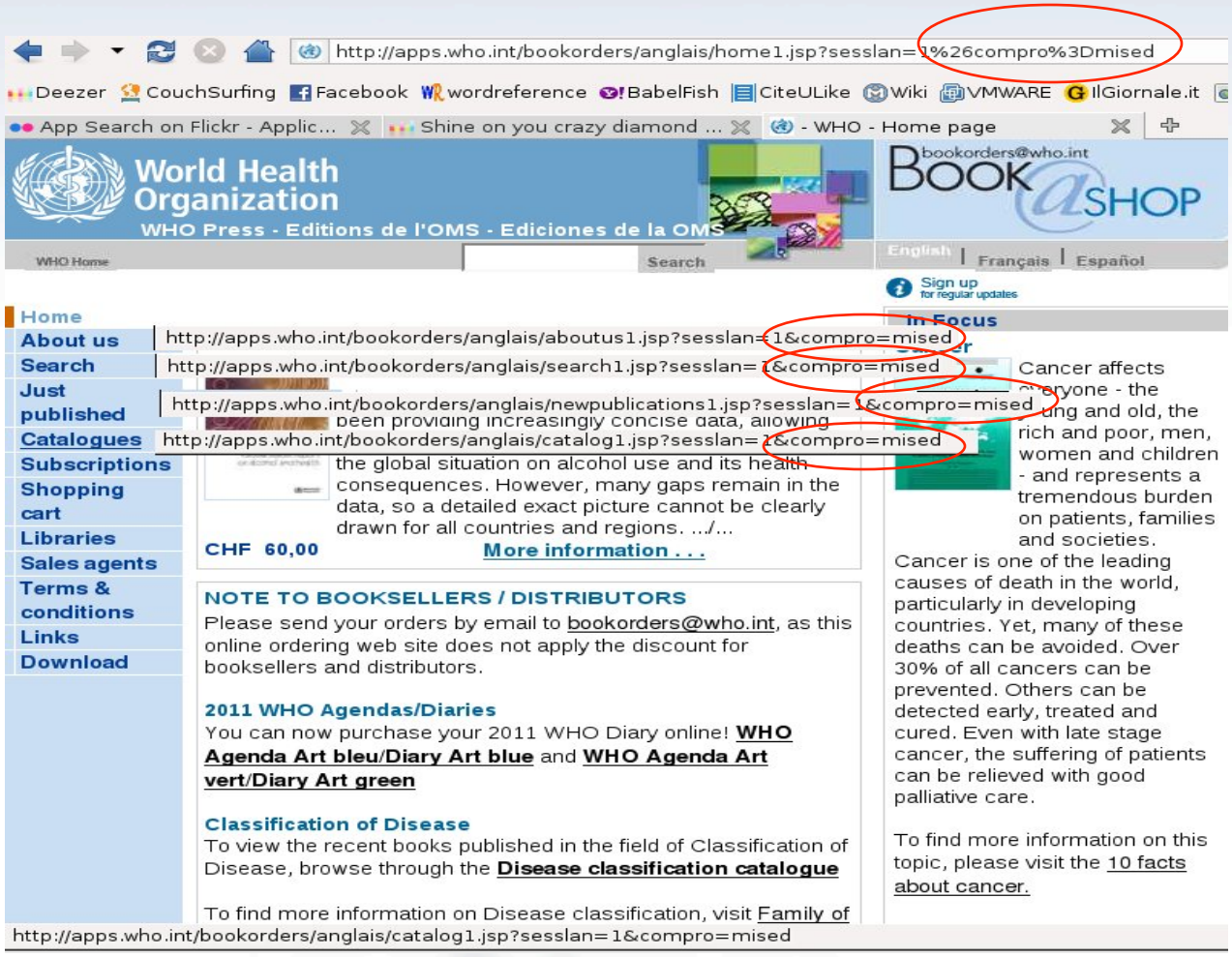
False Positives

- 10 applications (1.12%) use the injected parameter as entire target for one link
- Variation of the special case we saw in previous slide (V-Scan: special cases)
  - The application applied a transformation to the parameter before using it as a link's URL

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

World Health Organization



Your (secured) home banking

Raise your account limits and get Verified - PayPal - Mozilla Firefox

File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto

PayPal, Inc. (US) https://www.paypal.com/fr/cgi-bin/webscr?cmd=xpt/FinancialInstrument/popup/VerificationCC&EUPCCFee=1,50 EUR%26EUPCCFee%3D500,00 USD&ConfirmationDepositCurrency=EU

Link and confirm your card - ... | Raise your account limits an...

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 Pay

How do I confirm my card?

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

Sample card statement

Date	Description	Amount
01/06/2009	PP* 1234 EXPUSE	1,50 EUR

1234

Sample PayPal code

1. Enter your card information or select an existing card and click **Continue**. We'll charge your card 1,50 EUR&EUPCCFee=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 in to your PayPal account and enter the code from your statement.

We'll refund the 1,50 EUR&EUPCCFee=500,00 USD charge into your PayPal account within 24 hours after you confirm your card.

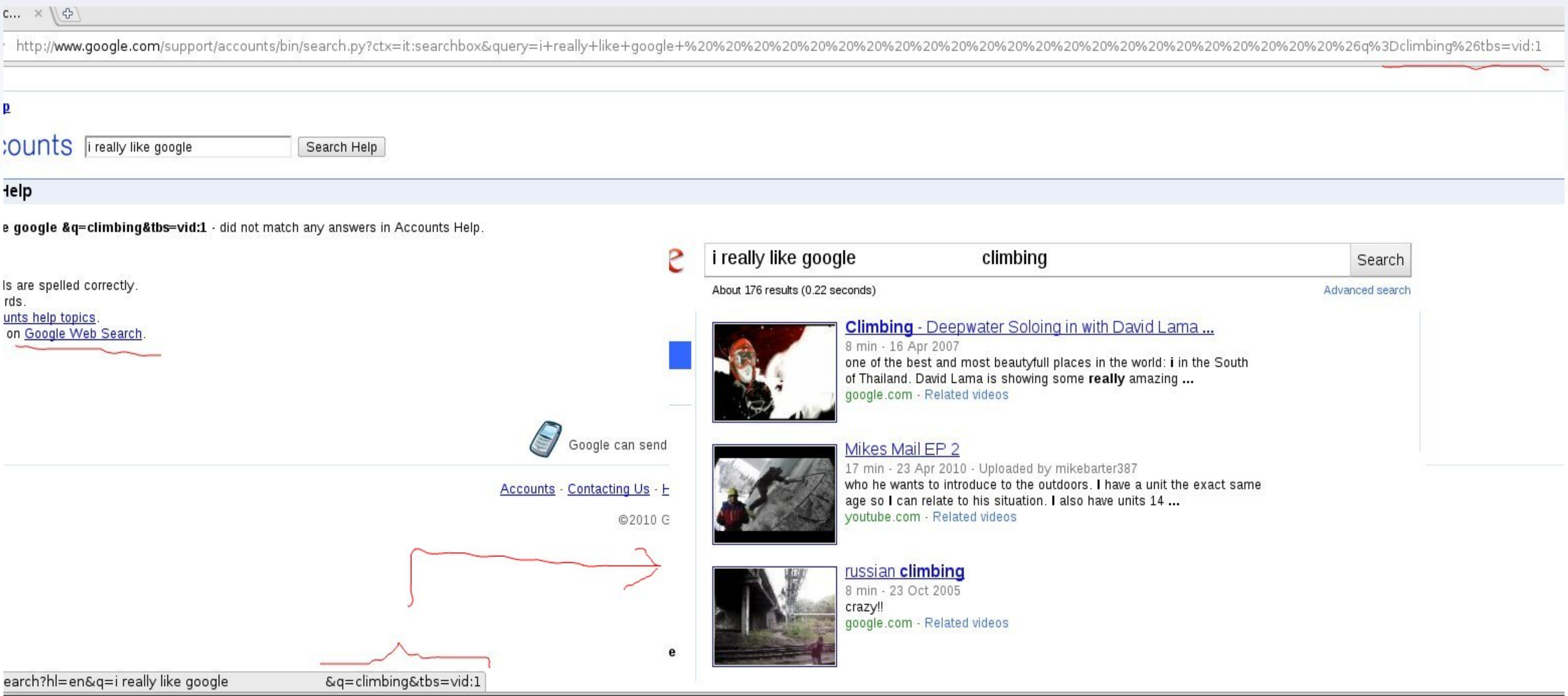
Console HTML CSS Script DOM Net

Modifica body < html.

```
<html class=" jsEnabled" lang="en">
  <head>
  <body>
    <div id="popup">
      <div id="logo">
      <div id="content">
        <div id="headline">
        <div id="main">
          <p>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 member.</p>
          <div class="boxemail">
            <h3>How do I confirm my card?</h3>
            <p>
              Follow the steps below unless you are confirming
              <a href="https://www.paypal.com/fr/cgi-bin/webscr?cmd=xpt/FinancialInstrument/popup/VerificationCC-RandomCharge&EUPCCFee=1,50%20EUR&EUPCCFee=500,00%20USD&ConfirmationDepositCurrency=EUR">domestic
              bank card</a>
```

https://www.paypal.com/fr/cgi-bin/webscr?cmd=xpt/FinancialInstrument/popup/VerificationCC-RandomCharge&EUPCCFee=1,50 EUR&EUPCCFee=500,00 USD&ConfirmationDepositCurrency=EUR

And Google ☺



http://www.google.com/support/accounts/bin/search.py?ctx=it:searchbox&query=i+really+like+google+%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%26q%3Dclimbing%26tbs=vid:1
i really like google
Search Help
e google &q=climbing&tbs=vid:1 - did not match any answers in Accounts Help.
ls are spelled correctly.
rds.
unts help topics.
on Google Web Search.
Google can send
Accounts · Contacting Us
©2010 G
earch?hl=en&q=i really like google &q=climbing&tbs=vid:1
i really like google climbing
Search
About 176 results (0.22 seconds)
Advanced search
Climbing - Deepwater Soloing in with David Lama ...
8 min - 16 Apr 2007
one of the best and most beautyfull places in the world: i in the South of Thailand. David Lama is showing some really amazing ...
google.com - Related videos
Mikes Mail EP 2
17 min - 23 Apr 2010 - Uploaded by mikebarter387
who he wants to introduce to the outdoors. I have a unit the exact same age so I can relate to his situation. I also have units 14 ...
youtube.com - Related videos
russian climbing
8 min - 23 Oct 2005
crazy!!
google.com - Related videos

What's next?

- Complementary approach: white-box (SCA)
- Server-Side flaws
- Technology: Pixy, RIPS

- Problems: Parsing, OOP support, Custom Sanitizations
  - PHP-Parser: https://github.com/nikic/PHP-Parser#readme
  - Saner
- Get in touch!

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!

Thanks for your attention.

I love you too, pollution!

Marco Balduzzi
<embyte@iseclab.org>

Acknowledgments & References

- Co-joint work:
  - M. Balduzzi, C. Torrano Gimenez, D. Balzarotti, and E. Kirda.
  - NDSS 2011, San Diego, CA.
  - *Automated discovery of parameter pollution vulnerabilities in web applications*
- Minded Security Blog, S. di Paola & L. Carettoni
  - http://blog.mindedsecurity.com/2009/05/client-side-http-parameter-pollution.html
- I collected a bunch of resources here:
  - http://papas.iseclab.org/cgi-bin/resources.py