Black Vectors of Web Exploitation
[Craniological Dissection of Web 2.0 Attacks.]
Analysis Through Live Case Studies.

Aditya K Sood aka 0kn0ck
Sec Niche Security
http://www.secniche.org

OWASP
6 September 2007
[] Who Am I?

- Independent Security Researcher.
- Founder, Sec Niche Security. [http://www.secniche.org]
- Active Speaker: CERT-IN, XFocus-XCON (China) etc.
- IS Author Hakin9, Hakin9 Linux+ etc. Authored Number of Security Related Papers. Research Featured as Global Security Perspective at FIRST.
- Released Advisories: Yahoo, AOL, MSN, Google, Verisign, Microsoft etc.

Projects:

- M-Labs: Digital Intelligence [http://mlabs.secniche.org]
- Trio Sec: An Active Penetration Testing Arena.[http://triosec.org]
Traversing Through Talk.

- Technology Variance.
- Application Bug Anatomy.
- Live Case Studies.
  - Double Trapping Injections.
  - Untamed Phishing.
  - Simulated Web Third Party Attacks.

Summary: Technology Variance.
Technology Variance!

- Flourishing of Development Matrix.
- Innovation for more Efficient Products.
- Cross Platform Interdependencies.
- Paradigm of Exploitation.
- Workflow and Custom Deadlines.
- Survivability with Ever Changing Requirements.
- Cross Referenced Matrix of Technologies.
Shifting Vector Towards Web

- System Bugs are Hard to Exploit.
- Internally Structured Protection Mechanisms.
- Enhanced Security Features To Dethrone System Bugs.
- Organizations are on RED ALERT.
- Web: The Hottest Place of Attackers.
- Web Application: An Easy Interface To Exploitation.
Squared Approach: Application Security

The Realm of Application Security

The Paradigm: Squared Approach

Privacy

Application Quality

Reliability

Security

(C) See Niche
[] Application Bug Anatomy

- Scratching the Cause of Bugs.
- Too Many Eye Balls Misses The Point All Together.
- The Application Development Matrix.
- The Exploitation Vector.
Injection Mechanism

Client Side Injection Mechanism

[ JSON / DOM / XML Injections]

MVC Framework

DWR

View

Controller

Model

Client Side Trust Zone

Server Side Trust Zone

JSON / DOM / XML Injections Written into DOMgie Browser - Client Side

© Sec Niche Security
Case Studies: Vulnerabilities

1. Double Trap Injections [One Step Ahead]
   1.1 Case Study of a Company.
   1.2 Learning Through Hard Knocking.

2. Untamed Phishing [Digging Deeper]
   2.1 Yahoo Search Engine Vulnerability.
   2.2 Yahoo Network Redirection and Phishing Vulnerability.
   2.3 Verisign Phishing Stringent Cases.
Case Studies: Double Trap XSS
[] Double Trap Injections : Core

1. **URL Banging**

   Injecting input parameters in the uniform resource locator's.

2. **Form Splitting**

   Injecting false arguments in the form values.

3. Input Validation points.


5. Insecure Coding.

6. Use of Obsolete Scripts.
Double Trap Injections: Model

Simplistic View: Duo Script Injection

First Injection Block

Second Injection Block

Injection Specific Parameters [4]
[] Double Trap Injections: Live Case Study

1. Attack Simulated in Security Consultation Website.

The Injection View:
[] Double Trap Injections: Live Case Study

<p>Please fill in the following fields:</p>
<form action="contact.cgi" method="POST">
  Your name:<input type="text" name="login" style="width: 300px; maxlength: 100" value="&lt;script&gt;alert("XSS Says: Let Me IN!");&lt;/script&gt;">
</form>

**Attack Undertaken:**

```html
<script>alert("XSS Says: Let Me In");</script><h1>XSS: I am in!</h1>
```

- Single Trap XSS
- Double Trap XSS
Double Trap Injections: Live Case Study

References:
2. http://cera.secniche.org/dbltrap.xhtml

Detailed Papers:
[Double Trap]
http://www.xssed.com/article/3/Paper_Double_Trap_XSS_Injection_An_Analysis/

[End Points Malfeasance]
http://sla.ckers.org/forum/read.php?6,8680
Yahoo Search Engine Flaw: Live Case Study
[] Yahoo Search Engine Flaw : Explanation

1. Vulnerability persisted in Yahoo Search Engine.

2. Links can be used by Phishers for Malicious Attacks.

3. Possible Cause: Handling of Redirection Variables.

4. An Ingrained Flaw.

The Vulnerable Link:

http://rds.yahoo.com/_ylt=A0geu4qjI3hGYOEAIjJXNyoA/SIG=14oi6m38j/EXP=1182364963/**http%3A//search.yahoo.com/search%3Fp=Hacking%26y=Search%26rd=r1%26meta=vc%253Din%26fr=yfp-t-501%26fp_ip=IN%26xargs=0%26pstart=1%26b=11
Yahoo Search Engine Flaw: Explanation

Persistent Link:

http://rds.yahoo.com/_ylt=A0geu4qjI3hGYOEAljJXNyoA/SIG=14oi6m38j/EXP=1182364963/**http%3A//%5B [Phishing Website]

Exploited Link:

http://rds.yahoo.com/_ylt=A0geu4qjI3hGYOEAljJXNyoA/SIG=14oi6m38j/EXP=11823663/**http%3A//www.metasploit.com

The Traffic is Redirected to the desired Link.
Yahoo Network Flaw: Explanation

1. The specific URL linked to any further yahoo website can be manipulated by the attacker to redirect the traffic and used for phishing.

2. The critical point is the URL can be called by third party for phishing.

Vulnerable Links:

[] Yahoo Network Flaw : Explanation

The Website Network Links:

https://us.ard.yahoo.com/SIG=12gb00bbf/M=341232.9804850.11489914.6055752/D=rest/S=150001465:R2/Y=YAHOO/EXP=1182284104/A=4651436/R=0/SIG=1255of0p5/ [ Website Link ]

The Manipulated URL’s

https://us.ard.yahoo.com/SIG=12gb00bbf/M=341232.9804850.11489914.6055752/D=rest/S=150001465:R2/Y=YAHOO/EXP=1182284104/A=4651436/R=0/SIG=1255of0p5/*http://www.google.com

https://us.ard.yahoo.com/SIG=12gb00bbf/M=341232.9804850.11489914.6055752/D=rest/S=150001465:R2/Y=YAHOO/EXP=1182284104/A=4651436/R=0/SIG=1255of0p5/*http://www.hushmail.com
[] Yahoo Flaws : Response

1. Website Vulnerability was Patched in 24 hours.
2. Search Engine Patch is in Development Stage.
[] Verisign Network Flaw

VeriSign™
The Internet Trust Company™
[] Verisign Network Flaw : Explanation

The Verisign Secured Network and Verisign Weblogs network is vulnerable to phishing attacks. The problem persists in the redirection links present which allows third party redirection.

The cause :
1. Redirection of traffic directly without visiting website.
2. The website wont check the link that is being called by the phisher.
3. Third party linking is possible.
4. Looping attack is also possible.

Vulnerable Links :

http://www.verisignsecured.com/Redirect.aspx?%5B [Website Name]
http://www.weblogs.com/clickthru?url=%5B [Website Name]
[ ] Verisign Network Flaw : Explanation

Attack Examples :

[Third Party SQL Injection Check]


[ Multiple Redirections ]


[ Blind SQL Check ]

[] Google URL Flaw
[ ] Google URL Flaw : Overall

1. Time to Time Google is Vulnerable To Phishing and XSS Attacks.


[Google Redirection Flaw]

http://www.google.com/url?q= < Website Link> /&sa=D&sntz=1&usg=1

The Link : http://www.google.com/url?q=

The Link is undertaken and applied as such.

From Previous Time This Link has shown discrepancies a lot.
We're sorry...

...but your query looks similar to automated requests from a computer virus or spyware application. To protect our users, we can't process your request right now.

We'll restore your access as quickly as possible, so try again soon. In the meantime, if you suspect that your computer or network has been infected, you might want to run virus checker or spyware remover to make sure that your systems are free of viruses and other spurious software.

We apologize for the inconvenience, and hope we'll see you again on Google.
Case Studies : References

http://www.secniche.org/advisory/YahooSearchPhishing_Vul.pdf
http://spamnews.com/Newsflashes/Newsflash/Two_Critical_Flaws_Found_in_Yahoo_200707116933.html
http://www.internetnews.com/security/article.php/3685131
http://article.wn.com/view/2007/06/23/Yahoo_Moves_Quickly_To_Plug_Phishing_Hole/
[ ] Digging Deeper : Web 2.0 Attacks
The Shift Towards Web 2.0

- **Classic Web Application Model**
  - browser client
  - user interface
  - HTTP request
  - http(s) transport
  - HTML+CSS data
  - web server
  - datastores, backend processing, legacy systems

- **Ajax Web Application Model**
  - browser client
  - user interface
  - JavaScript call
  - HTML+CSS data
  - Ajax engine
  - HTTP request
  - http(s) transport
  - XML data
  - web and/or XML server
  - datastores, backend processing, legacy systems

This diagram illustrates the shift towards Web 2.0, highlighting the changes in client-server interaction and data handling.
The Overall Structure: Web 2.0
The Shifting Points

1. Iframes Subtle Functioning and Implementation for Asynchronous Requests.
2. The XHR [XmlHttpRequest] on Run.
3. Extensibility Behavior of XHR over Iframe.
4. Cross Domain Calls are not processed Directly in XHR.
7. Asynchronous Implementation With an Ease through XHR.
8. The Endpoints Consideration is always Javascript Code [XHR+JSON]
Web 2.0 Most Favorable Attacks.

1. Incore XSS Attacks.
2. Cross Site Request Forging Attacks
   - 2.1 Direct Simulation.
   - 2.2 Indirect Simulation by Fusing Proxy.
3. The Serialization Stringent Attack Anatomy.
4. Denial of Service Attacks Through URL Concatenation.
[] XSS Attacks.

1. Cross Site Scripting Attacks are High.

2. Injected Parameter Processed by Server Renders the DOM on the Client Side to Cause an Injection.


4. Injections Use:
   - 4.1 `<script>alert(“XSS”);</script>`
   - 4.2 `document.cookie`, `document.domain` etc.
   - 4.3 `<img src=“malicious.jpg”></img>`
   - 4.4 `Eval` etc.

5. Information Disclosure at Full.
XSS Attacks: Example

http://www.icfai.org/icfe/current_students.asp?msg=%3Cscript%3Ealert(document.cookie);%3C/script%3E
[] Cross Site Request Forging: Direct

1. Cookie Hijacking with Session Undertaking.
2. Authenticating on the Behalf of User by an Attacker.
3. No Stealing of Cookies but Dynamic Manipulation.
4. Lets see:

```html
HTTP Request ①
GET / HTTP/1.1
Host: www.example.org

HTTP Response ②
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1234

<html>
  <a href="http://stocksexample.org/buy.php?symbol=GOOG&shares=1000">Buy</a>
</html>

CSRF Attack ③
GET /buy.php?symbol=GOOG&shares=1000 HTTP/1.1
Host: stocksexample.org
```
[] Cross Site Request Forging : Indirect

1. Fusing Proxy between End Points.

```
HTTP Request
GET / HTTP/1.1
Host: www.example.org

HTTP Response
HTTP/1.1 200 OK
Content-Length: 1234

<html>
  ...
</html>
```
[] Cross Site Request Forging : Indirect

1. Example : JSON Attack Point

```json
{
   "menu": {
      "value": "<img src="https://trading.example.com/xfer?from=MSFT&to=RHAT&confirm=Y">",
      "popup": "<script src="https://www.google.com/accounts/UpdateEmail?service=adsense&Email=mymail@newmail.net&Passwd=cool&save="/script>"
   }
}
```
[] Serialization: JSON/BI SON/ AJ AX

1. Object Interoperability.
2. Concept is Based on Serializing Data i.e Strings.
3. Web 2.0 Finest Edge Driven Attack Vector.

A JSON Layout
[] Serialization : JSON/ BISON/ AJAX

Example:

Send

```
{
    info: "[*] Array Infection Test!",
    InfectedArray:
    "<h3>Exploiting Serialization!</h3>",
    "<a href='http://www.google.com'>GOOGLE : Through Serialization</a>",
    "Array Infection Successful!"
}
```

Receive

```
Object
info | [*] Array Infection Test!
InfectedArray | Array
0
   Exploiting Serialization!
1 GOOGLE : Through Serialization
2 Array Infection Successful!
```
[]Dos : Web Denial of Service Attacks

2. Recursive Calling of URL through Concatenation.
3. Looping Iframe Tags against Entangled Web Entity.

Example:

Loop {
    Iframe Tagging.
}

// Load The Script
Conclusion

Integrity – Security Web Relation

Dual Side Data Transference [Client / Server]

Integrity and Confidentiality of Communication Channel

Integrity of Client Side Business Logic

Integrity of Client Side Data

[ © Sec Niche Security ]
Questions

Contradictory View Always Welcomed!
Thanks

Aditya K Sood, Security Researcher.

aditya_ks [at] secniche.org

Zeroknock [at] secniche.org