

2nd ISSNSM's Tutorial on

Hacking Web2

(Tutorial T1)

Speaker:

Radu State

June 2, 2008

Web Hacking

Radu State
Ph.D.

MADYNES

The MADYNES Research Team
LORIA – INRIA Lorraine
615, rue du Jardin Botanique
54602 Villers-lès-Nancy
France
Radu.State@loria.fr

Emanics Summer School, 2008 Zurich

- 1 -

What is Web Hacking ?

Penetrate the network using web applications
and servers

How is this done

1. Exploit vulnerable servers (SSL buffer overflows, directory traversal, etc)
2. Exploit weak configurations
3. Exploit web applications

Emanics Summer School, 2008 Zurich

- 2 -

Security threats and vulnerabilities

- **What is Security ?**
 - “Security is a process not a product”, Bruce Schneier,
 - “Maintaining an acceptable level of perceived risk”, Richard Bejtlich.
- **What is a threat ?**
 - A threat is an external security issue represented by a natural or man-made attack
- **What is a vulnerability ?**
 - a specific degree of weakness of an individual computer or network exposed to the influence of a threat
- **What is risk ?**
 - A risk is the degree of probability that a disaster will occur in light of the existing conditions, and the degree of vulnerability or weakness present in the system. The key difference between a threat and a risk is that a threat is related to the potential occurrence of a security issue, whereas a risk is the probability of an incident occurring based on the degree of exposure to a threat. Risk, for security purposes, is usually calculated in dollars and cents.

Threat Modeling

- **Closely related to a specific enterprise**
 - Takes into account users, roles, access, services, natural conditions etc..
- **Several models exists:**
 - The OCTAVE approach, Carnegie Mellon
 - STRIDE (Microsoft)
- **Objective**
 - Identify the threats and assess their impact
 - Produce a structural models of threats and countermeasures.

Vulnerabilities disclosure

- SANS (www.sans.org) keeps an updated view on the most 20 dangerous vulnerabilities /attack targets
- CERT (Computer Emergency Response)
 - Various regional/national sub groups
 - Historical source of information on vulnerabilities
- Web Sites/Mailing Lists
 - Milw0rm
 - Secunia
 - fulldisclosure

Security Assessment/Penetration Testing

- Security Assessment
 - identifies potential vulnerabilities, their impact and potential impact.
 - Provides a global view on the security of the overall network and services
- Penetration Testing
 - breaking into and exploiting vulnerabilities in order to replicate an real hacker
 - “Show” and very impressive
 - Limited, because maybe more ways to intrude might exist

What you need to know

- Network and application level knowledge
- A keen eye, open mind and curiosity to learn how things work
- A passion for generating and analyzing error messages.
- Master the toolsdo what You want to do, not what the tools can do.
- Ethics....
- Service continuity
 - Use off time business hours
 - Do not test DOS attacks
- You might go to jail if your actions affect third parties not included in the contract or national laws.
- Do not assess or perform penetration testing on networks that are not yours or for which you don't have a written permission

What do you search

1. A communication channel
2. A username
3. A password

Remember: If you know two of them, you can bruteforce the third.

Reconnaissance gathering

- **Objective :** Learn the most about a network
- **Who is doing it .**
 - Hackers going after your assets
 - Script kiddies running scanners
 - WORMS looking for new propagation and replication places
 - Automatised attack and installation software
- **What to learn about a network:**
 - Network topology (IP subnetworks, alive etc..)
 - Firewall ACL
 - Operating systems and the services/programs running
- **Approaches**
 - « Google hacking » - use google to search for vulnerabilities :<http://johnny.ihackstuff.com/>
 - DNS and internet databases
 - Scanning
 - Inverse mapping for network topology
 - Port scanning for OS fingerprinting and service identification
 - SNMP
 - Passive monitoring

Reconnaissance gathering

Objective : Learn domains and real network associated to an organisation.

Tool : Whois Databases

- European IP address allocation : www.ripe.net
- US army : whois.nic.mil
- France : whois.nic.fr

Example : Discover organisation information about Loria:
`whois « loria.fr » -h whois.nic.fr`

Information about :

- administrative contact (can be reused in social engineering)
- Network domains, name servers and allocated IP addresses

Reconnaissance gathering with DNS

Objective : Discover the network topology by DNS interrogation.

Tools : nslookup, dig, , zone transfer tools (SAM-SPADE, Smart-Whois, etc...)

What to discover !

- Name servers (ns entries)
- Mail servers (mx entries)
- Any IP and names visible
- HINFO records about systems
- Reverse DNS for more stealth

Exploiting web servers and configuration

Software :

- A server is just a piece of software, therefore it can be broken if software is not well written
- Famous examples
 - SSL buffer overflows against Apache
 - Directory traversal against ISS and Apache :
`www.vulnerable.com/../../../../../../../../etc/passwd`
- Configuration
 - Files with confidential information on the server (google hacking with ext:xls...)
 - Unprotected sensible zones
 - Security by Obscurity

Exploiting web applications

Major causes of threads:

- Programmers are busy, not well trained on security and sometimes lazy
- Security by obscurity
- Multiple programming languages and character formats
- Integration of multiple applications (web front, database servers, and programming environments)

What are the major 10 threats ? OWASP

- A1 – Unvalidated Input
- A2 – Broken Access Control
- A3 – Broken Authentication and Session Management
- A4 – Cross Site Scripting (XSS) Flaws
- A5 – Buffer Overflows
- A6 – Injection Flaws
- A7 – Improper Error Handling
- A8 – Insecure Storage
- A9 – Denial of Service (DoS)
- A10 – Insecure Configuration Management

What are the major threats ? WASC

1. **Authentication**
 - Brute Force
 - Insufficient Authentication
 - Weak Password Recovery Validation
2. **Authorization**
 - Credential/Session Prediction
 - Insufficient Authorization
 - Insufficient Session Expiration
 - Session Fixation
3. **Client-Side Attacks**
 - Content Spoofing
 - Cross-site Scripting
4. **Command Execution**
 - Buffer Overflow
 - Format String Attack
 - LDAP Injection
 - OS Commanding
 - SQL Injection 4.6 SSI Injection 4.7 XPath Injection
5. **Information Disclosures**
 - Directory Indexing
 - Information Leakage
 - Path Traversal
 - Predictable Resource Location
6. **Logical Attacks**
 - Abuse of Functionality
 - Denial of Service
 - Insufficient Anti-automation Insufficient Process Validation

Input Validation

- Can you find any limitations in the defined/used variables and protocol payload, that is, accepted data length, accepted data types, data formats, and so on?
- Use exceptionally long character-strings to find buffer overflow vulnerability in the application code base or the web server itself.
- Use concatenation techniques in the input strings to try to get the target application to behave incorrectly.
- Inject specially crafted SQL statements in the input strings
- Force Cross-Site Scripting (XSS) functionality
- Look for unauthorized directory or file access with path or directory traversal in the input strings of the target application.
- Try using specific URL-encoded strings and Unicode-encoded strings to bypass input validation mechanisms used within the target application.
- Use of server-side includes, try executing remote commands.
- Manipulate the session management techniques to fool Try to manipulate (hidden) field variables in HTML forms to fool server-side logic.
- Manipulate the "Referrer" value in the HTTP "Host" header in order to fool or modify server-side logic.
- Try to force illogical or illegal input so as to test the target's error-handling routines.

Input Validation pentesting

Inject server side script :

[http://example.com/index.php=<?passthru\("/path/to/prog"\);?>](http://example.com/index.php=<?passthru().

Execute other commands:

<http://example.com/foo.pl?page=../../../../bin/lis%20-las%20/home>.

Bypass filtering mechanisms when Perl and C use other conventions:

<http://example.com/foo.pl?page=../../../../etc/passwd%00html>

Path traversal

<http://example.com/index.php?file=../../../../etc/passwd>

Use alternate character sets

- `..%u2215` : Unicode encoded backward slash character
- `..%c0%af` : UTF-8 encoded forward slash character

Breaking Access Control

- How is the app administrated? By how many people? And what gives them that right above regular app users?
- How are changes made to content? How are these changes published to production?
- How many people have publishing rights? How are those rights determined, established, and enforced?
- Is there a QA testing and verification process for content?
- How are changes made to the app? How are these changes published to production?
- How many people can touch the app to publish new or updated code? Are they developers? How are those rights determined, established, and enforced?
- Is there a QA testing and verification process for app modifications?
- Is any of the publishing or deploying done remotely? If so, how?
- How is the DB maintained and administrated? By how many people? Do the DBAs have remote access to the DB server(s)?
- Is the app segmented by access control or is there one blanket group with publishing rights?

Breaking Authentication

Attempt to concretely ascertain the authentication mechanism that is in place

Verify that said mechanism is being used uniformly across all sensitive resources

Verify how this mechanism is being applied to all the resources within the Web application

Web Authentication

Types of authentication

1. Basic Authentication with username send almost in clear -base64 encoded)
2. HTTP digest using M5 cryptographic hashes
3. HTML forms (using maybe an additional databa)
4. Windows specific (NTLM kind of)

Breaking authentication

Brute force (using brutus)

Database SQL injection

Hacking the session management

Hacking the sessions

How are sessions maintained ?

1. Using a mixture of headers (referer, url, IP source) and cookies (most cases an encrypted and time stamp based system)
2. Sometimes with hidden HTML field ☺

Breaking sessions

Detecting the predictability of session generation mechanism

Examples: Easy to break;

<http://example.com/<filename>/191-4039737-1105>

<http://example.com/<filename>/162-4039740-1105>

Not so easy

<https://example.com/login.jsp?token=E7F8C189-728F-46EA-A3FE-FABA5B9384D0>

<https://example.com/login.jsp?token=A5BD2BBA-311D-4625-A218-8AC51C7AB688>

Hacking the sessions

Session reuse where an old session ID can be replayed.

Session fixation where an attacker initiates a session and somehow convinces the victim to connect using this session

By email/roque server

```
<a href="http://example.org/index.php?PHPSESSID=987654321">  
  Don't Click here!! </a>
```

By Javascript injection: Jikto

XSS Cross site scripting

Hacker injects scripts in vulnerable applications (forums, online shared virtual spaces, logs)

```
<ahref="http://example.com/viewdata.cgi?comment=<script>MALICIOUS%20SCRIPT</script>">My link!</a>
```

Victim executes the script on his machine when visiting vulnerable system (efficiency MySpace worm Sammy infected 1000000 machines)

```
<div class="comment"> <p>Hello, user!</p> <script>MALICIOUS CLIENT-SIDE CODE</script> <p>Anyone up for a party?</p> </div>
```

Dangers:

Theft of identity/cookies

Abuse of client machine (interception with invisible frames, penetration of internal networks)

User tracking

Injecting commands

Perl based cgi :

Valid URL

:

<http://example/cgi-bin/showInfo.pl?name=John&template=tmp1.txt>.

Attacking :

<http://example/cgi-bin/showInfo.pl?name=John&template=/bin/lsl>.

Executing `open(FILE, "/bin/lsl")`

A PHP script using `exec("ls -la $dir", $lines, $rc)`

`;%3B`

Attacking

<http://example.com/directory.php?dir=%3Bcat%20/etc/passwd>.

