Attacks on Web Applications

Lecture on

Web Attacks

Attacking Web Applications

Walter Kriha

1

Goals

Show the steps needed in attacking a web site

Cover the immense problem of input validation

Cover useful tools

The final goal lies in making you sensitive for possible holes in your web sites security.

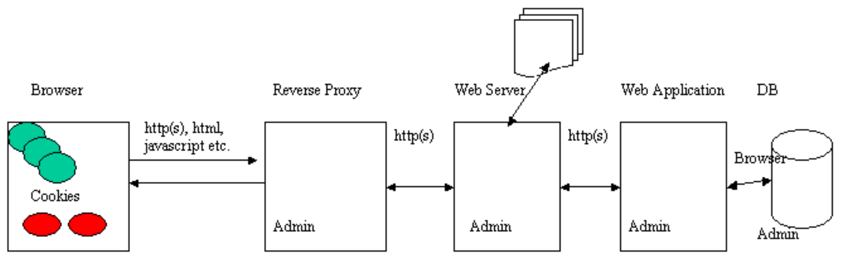
2

The steps of an attack

1. Gather intelligence (portscans, application errors, infrastructure leaks)
2. Match the information won against known defects (OS releases, Web Server versions and types etc.)
3. Start attacking the weak spots using input validation problems (from wrong directory path handling to SQL statements)

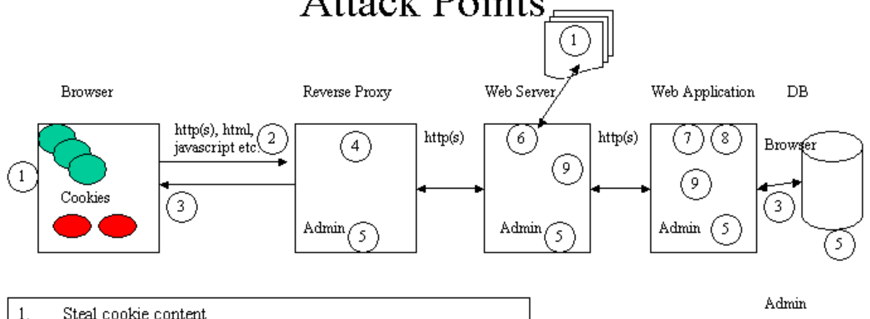
3

Canonical Web Architecture



Do we assume an Internet thread model for this web application? This means that we assume all components are uncompromised and the data are only vulnerable when transported. This is at least with respect to the browser a strong assumption. We will list the most common forms of attacks using this architecture.

Attack Points



1. Steal cookie content
2. Direct user to bogus servers (SSL redirect)
3. hijack user session (or DB connection)
4. crack proxy where SSL session ends
5. crack web components through admin interface
6. leverage known web server bugs with respect to file access or path interpretation
7. Downgrade authentication to low secrurity level
8. Use application bugs in authorization to gain illegal acces (privelege extension or upgrade)
9. Use fake input to work around controls

What makes attacks easy

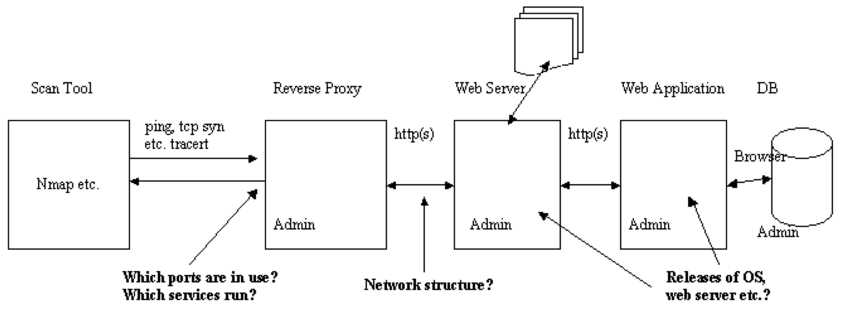
Lazy administration and configuration problems

- leavin admin interfaces unprotected
- forgetting to enforce security levels for authentication and privacy (e.g. SSL configuration)
- let users see application internal error messages
- bad authentication (basic authentication etc.)
- forgetting to do security updates of web components

Bad software

- buffer overlows in web components
- input validation errors in web components
- bad implementations of cryptographic algorithms
- directory path interpretation bugs (e.g. do a directory security check first and then convert from unicode characters to internals)
- store important data in cookies with weak encryption and without replay protection

Gathering Intelligence : scanning and probing



Scan tools like nmap allow an automated and precises investigation of the target infrastructure. This step needs to be differentiated from mounting the actual attack by probing for the proper buffer overflow string or input validation problem.

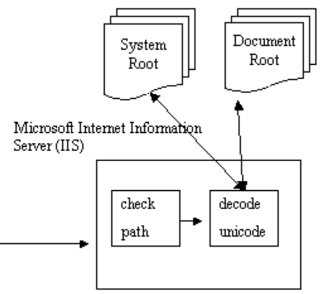
Input Validation Attacks

- Directory Traversal Attacks
- SQL errors
- size limits
- Cross site scripting
- hidden fields
- Unicode parse errors

Directory Traversal Attacks

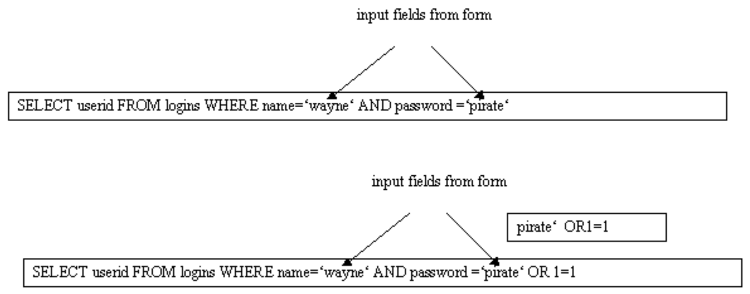
Get /scripts/..%c0%af../winnt/system32/cmd.exe?+/c+dir+‘c:\‘

HTTP /1.0



The whole trick behind directory traversal attacks is to find a server that does input validation in the wrong order. This means the web server first does a security check on the Get request to find illegal ..\.. sequences that would allow access outside of the document root. But after the check the original request gets transformed (decoded) and NOW the illegal ..\.. are created. The examples for IIS attacks are taken from Joel Scambray/Mike Shema, Hacking exposed, Web Applications (see resources)

SQL Injection



SELECT userid FROM logins WHERE name=‘wayne‘ AND password =‘pirate‘

SELECT userid FROM logins WHERE name=‘wayne‘ AND password =‘pirate‘ OR 1=1

The added OR 1=1 effectively disables the password check. It is a very simple SQL injection attack. Other tactics involve the use of SQL comments (--), built in names of variables or stored procedures etc. A detailed attack is described in Hacking exposed, Web Applications (see resources)

Server State on Client Attacks: hidden fields

<input name="masteraccess" type="hidden" value="N">

Client using e.g. Achilles proxy changes „N“ to „Y“ and becomes administrator. This is the classic case of privilege extension

<input name="masteraccess" type="hidden" value=„Y">

one solution is to encrypt the admin parameter in the form. Must make sure that no replay attacks are possible and that the user cannot create those encrypted tokens himself.

<input name="masteraccess" type="hidden" value="sa8ay23&^*&^7443&*(20f9dsf"

from: Mark Curphey, http://www.owasp.org/asac/parameter_manipulation/forms.

Session Takeover via Session Token

static part: date/time: counter:

http://www.somesite.com/view/AKT25050312451234

http://www.somesite.com/view/AKT25050312471241

SESSIONID=AKT25050312451234

<input name="masteraccess" type="hidden" value=" AKT25050312451234 "

The first example uses URLs to transfer the session token, the second stores it in a cookie and the third uses a hidden field to transport the token. In all three cases transfer via SSL would be a MUST but the token itself is extremely unsafe because the client or other people can easily guess how it is constructed. Remember: the session token stands in for YOUR session after authentication is done!!! Beautiful examples: David Endler, Brute Force Exploitation of Web Application Session IDs, www.idefense.com

Cross Site Scripting (Script Injection)

„Cross-site scripting (also called XSS) is somewhat different to the other attacks discussed thus far. Rather than attack the server or the application XSS attacks are aimed at the end-user's browser. XSS is not a problem of input sanitation, but rather a problem of output sanitation. It is typically used against sites that redisplay values that are input by the user. If the information presented by the user is not properly sanitized before it is displayed then an attacker may be able to use HTML tags to influence the way it is displayed by the site. This becomes particularly dangerous when the attacker inserts the <SCRIPT> tag, thereby forcing other users' browsers to execute the code the attacker specifies. An easy way to test for this is to insert HTML like this:

<SCRIPT>alert('Vulnerable!')</SCRIPT> If the Web site does not clean this up before displaying it will cause a pop-up message to be displayed by the visitor's browser. Obviously an attacker could do much worse than cause pop-ops, particularly when the site is trusted by other users. „

Taken from: http://www.securityfocus.com/infocus/1632 , Charl van der Walt, Assessing Internet Security Risk, Part Five: Custom Web Applications

Unicode

Code points for most characters in the languages of the world	Unicode code points (names and numbers of charcters) 9% of 4 Gigabyte
UTF8, UTF16 or UTH32 Encodings of code points (code units or blocks)	3 different ways to encode ALL code points (size vs. performance)
arbitrary glyphs (fonts)	Not defined by unicode.

Important points: no glyphs are standardized. One code point can map to SEVERAL code units. Only the shortest form (typically the ASCII representation) is now valid and processors are not allowed to interpret others (see next slide: UTF8 exploit)

Unicode Exploit

code point U+0000	Unicode code points (names and numbers of charcters) 9% of 4 Gigabyte
encoded as: 0, 110 00000 10 000000, etc.	

Processors are not allowed to interpret any encoding other than the shortest form, in this case 0. Otherwise the extended forms could escape filtering and become active during interpretation.

Unicode Visual Spoofing (homographs)

Two different code points

Algorithmically the code points are absolutely different

two different encodings

same goes for the encodings

two different fonts

Fonts can display unicode code points any way they want.

I,l,O

One visual „look“ (e.g. lowercase „l“ and uppercase „I“ or greek omicron vs latin o.

From: J. Meister (see Resources). This mapping of different values and meanings into one appearence confuses users and opens attack vectors.

Unicode homographs and DNS

Two different code points

ASCII DNS

Unicode Characters DNS

DNS names can now contain Unicode characters

two different fonts

Not defined by unicode.

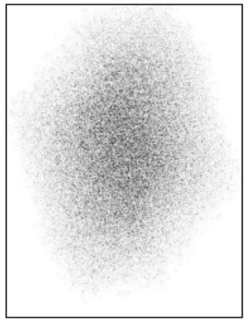
I,l,O

One visual „look“ (e.g. lowercase „l“ and uppercase „I“ or greek omicron vs latin o.

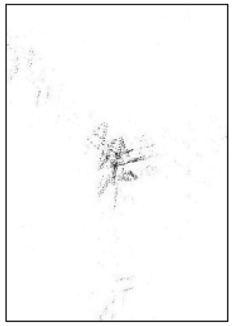
The firefox browser switched back to showing the unicode escape sequences in domain names to allow the user to differenciate e.g. a latin „a“ from a kyrillic „a“. Otherwise the user could be tricked into connecting to www.ebay.com with the „a“ being really the cyrillic version. In this case the user would connect to the wrong site. Expect many more security problems with unicode in the future, especially in the GUI area.

Quality of Random Numbers

Linux 2.2



Windows NT without patches



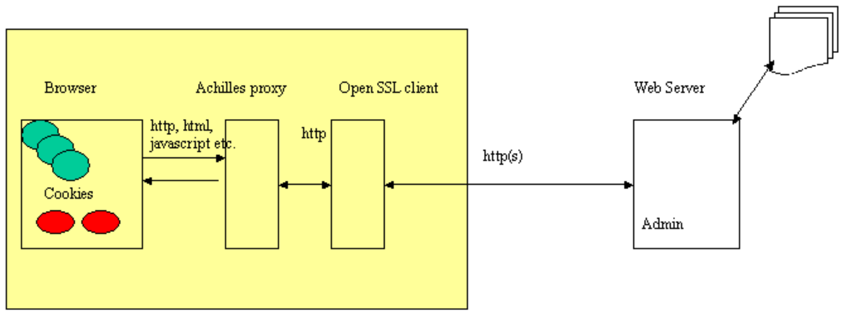
Michael Zielewski made the „randomness“ of pseude random number generators visible using a phase space analysis based on strange attractors. A sequence of PNRGs is converted into 3D space using a formula like: x[n] = s[n-2] - s[n-3] y[n] = s[n-1] - s[n-2] z[n] = s[n] - s [n-1] Notice the very regular structure on the right side which leads to a 97.5 % attack feasibility compare to 0.05% on the left side.

Useful Tools

- Automatic discovery (scan and probe)
- Generating and reading web traffic without a browser
- Tracking and modifiying web traffic transparently
- Tool to support client side SSL (and still be able to track everything)
- Tools to replicate complete web sites

On the next slide a generic setup is given to track all traffic between a browser and a web application. This allows investigation of temporary cookies etc. For tools see resources.

Tracking Setup



This setup allows the inspection of the complete traffic between user agent and web server. http header information can be manipulated. The use of the SSL client is of course only needed when the web server runs SSL.

Semantic Attacks

Check out breaking news at CNN:
<http://www.cnn.com&story=breaking_news@18.69.0.44/evarady/www/top_story.htm>

(Unfortunately, the URL no longer works. But stick with me.) At first glance, this looks like a CNN URL. But the URL does not lead to, or does not redirect from, cnn.com. The page is not CNN's. The URL is a clever hack that plays with people's assumptions about what a URL is supposed to look like.

Here's how it works. An MIT student created a fake Web page and put it up on his Web site at:

<http://salticus-peckhamae.mit.edu/evarady/www/top_story.htm>

He then sent out the first URL above. If you examine that URL carefully, you can see that the host name is not "www.cnn.com" but "18.69.0.44," which is the same as salticus-peckhamae.mit.edu. (For extra obfuscation, he could have converted that host name to decimal.) That entire bit before the @-sign -- "www.cnn.com&story=breaking_news" -- is a "username," something allowed by the HTTP specification but rarely used in actual URLs.

Taken from Bruce Schneiers Cryptogram newsletter. http://www.counterpane.com/crypto-gram-0010.html#1 on semantic attacks. Certificates used in SSL also allow for semantic attacks because the name part of the certificate is both not really checked by most users AND does not come from one standardized namespace.

E-Mail based semantic attacks

The typical e-mail fraud goes like this:

1) Mail seems to come from a big public site (e.g. ebay or paypal). HTML elements are used to make it look real (better disable this html feature in your browser NOW – that is if you CAN)
2) The mail claims that due to technical problems the account holder should login to his or her accout to check if things are still OK.
3) If the user follows the provided link he or she will send her account data with password not to ebay or paypal but to some attacker which will turn around and take over the account.

This example is from a paypal attack:

The URL listed was https://www.paypal.com/cgi-bin/webscr/?cmd=_login-run.

when clicked it directed the user to a seemingly secure site, but with a URL like: http://www.paypalsys.com/. Users were then asked to log in with their e-mail addresses and their passwords.

(Example taken from: http://www.internetnews.com/ec-news/print.php/1470291

Resources (1)

- A good article on tcp sequence number generation on different platforms http://razor.bindview.com/publish/papers/tcpseq.html
- a good article on brute forcing Session IDs: www.idefense.com/idpapers/SessionIDs.pdf
- Joel Scambray, Mike Shema, Hacking exposed. An easy to read guide for web site security (does not cover application servers). Mostly correct with some errors e.g. in offering multiple authentication methods leading to a downgrade in security. Also the explanation of MSpassport seems to be incorrect with respect to cookie handling. Shows the problem of input validation very well. homepage: www.webhackingexposed.com with all the tools mentioned.

Resources (2)

- www.mavensecurity.com A web site with intentional bugs for you to find them. Also home of Achilles, the web attack proxy.
- List of security tools at: http://www.insecure.org/tools.html They also have a good newsletter.
- Web Site Test Tools and Site Management Tools at: http://www.softwareqatest.com/qatweb1.html#SECURITY
- www.securityfocus.com Look for articles by Charl van der Walt on Web Application Assessment (5 parts).
- David Endler, Brute Force Exploitation of Web Application Session IDs, www.idefense.com Excellent paper on session id hacking.
- Strange Attractors and TCP/IP Sequence Number Analysis, http://razor.bindview.com/publish/papers/tcpseq/print.html Michael Zalewski shows how strange attractors display the quality of random number generators – which are e.g. used for encrypting session tokens. Very nice pictures.

Resources (3)

- M. Davis, Security Considerations for the Implementation of Unicode and Related Technology (http://www.unicode.org(/reports/tr36/tr36-2.html) (from J.Meisters thesis on internationalization of applications). Explains attacks through unicode non-minimal codes and visual spoofing (homographs)