Web Security Threats

Giuseppe Paternò

February 2004

Agenda

- Objective of the presentation
- Why firewalls are ineffective
- Application security
- Known attacks
- Some suggestions

What I am here for

- I don't want to create security guys
- I don't want to teach you how to program or configure a product
- I want to make sure you're aware of what's “out there” about cracking

Firewalls are death

- "Have you got any security issue? Then you need a firewall" (sales men in 2k)
- The web is going to be the only available service to the Internet (along mail and DNS)
  - Newer attacks will be propagated through the web
  - If HTTP is the way to exploit a remote network, what is IP filtering for ?
- Goodbye to firewalls

Security in applications

- Security should be set at application layer
- If attacks are now at the application layer, the only way to protect is:
  - To configure the application properly (webserver, directory server, application server)
  - To "code" (i.e. program) in such a way that is difficult for an intruder to break in or to gain more privileges

Known threats

- Current techniques to exploiting a web application (it also applies to webservices)
  - Default configuration
  - Buffer overflow
  - RootKits
  - HTML code injection
  - Error handling
  - SQL/LDAP/XML injection
  - Social engineering (not really technical, but...)

Default configuration

- Configuring an application with defaults is the worst enemy
  - For example using user "root" password "root" on an Unix system
- Configure the application properly, eg:
  - Change default passwords
  - Give least access to the server (not let them running as root)
  - Using ACLs in LDAP to prevent information evasdropping
  - Restrict list of allowable IP addresses

Buffer overflow

- It has been discovered in mid 1990 and it's a common technique today
- It exploits an unchecked buffer size in programs and overwrites the program code, so that it points to a different memory address and execute a shell on the remote OS.
  - www.linuxjournal.com/article.php?sid=6701

Buffer overflow

- The buffer overflow theory

/* note that the size of the buffer is 256 bytes, but the loop inserts 512 bytes of data */

```
void func(void) {
   int i;
   char buffer[256];
   for (i=0; i<512 ; i++)
       buffer[i] = 'x';
   return;
}
```

Buffer overflow

- Buffer overflows are often used by malicious programs named as exploits
- Java usually do not suffers of buffer overflows
  - Better, it has not been proven yet
  - Theoretically it can be done where buffers has been used inside a program (socket listening)
  - The attack can be addressed to the JVM and not to the program itself

RootKits

- A collection of tools that allows a cracker to provide a backdoor into a system, collect information on other systems on the network, mask the fact that the system is compromised, and much more. Rootkit is a classic example of Trojan Horse software.
- Eg: Adore, T0rn, etc... (also avalable for Windows)

HTML Code Injection

- Attackers are often able to embed malicious HTML-based content within client web requests. Attackers can exploit these flaws by embedding scripting elements (JavaScript) within the returned content without the knowledge of the sites visitor, for example to inject a trojan horse.
  - www.technicalinfo.net/papers/CSS.html

URL poisoning / File inclusion

- Sometimes applications loads external files or external URLs, for example:

link.jsp?url=http://www.mysite.com

include.jsp?page=mypage.jsp

- Check always the input: this can be modified by an intruder as

link.jsp?url=file:///etc/passwd

include.jsp?page=../../../etc/passwd

Error handling

- Displaying errors are used mostly for debugging/troubleshooting
- Error handling are "false friends"
  - Used by attackers to reveal how the web site is structured: most of the times are used to understand database tables and how queries are structured (and do SQL injection)
- Suggestion: give minimal error to user and log to a file (log4j).

Error handling

- Example:

http://www.mycompany.com/product.php?id=182924983739 4

Error at line 125: Unable to perform query: select Date,Object from where Date > NOW() - INTERVAL 1 YEAR order by Date :You have an error in your SQL syntax near 'where Date > NOW() - INTERVAL 1 YEAR order by Date' at line 1

- It reveals:
  - Database tables (two queries)
  - Which line of the code is in error (line 125)

SQL injection

- It is a trick to inject SQL query/command as form input.
- Many web pages take parameters from user, and make SQL query to the database.
  - Eg: web login page with user name and password and make SQL query to the database.
  - With SQL Injection, it is possible for us to send crafted user name and/or password

SQL injection

- Example FORM posting to:

https://www.mycompany.com/servlet/login?userid=shmoe&password=dumb

- I could write it as:

https://www.mycompany.com/servlet/login?userid=shmoe&password=letmein'%20OR%20'a'='a

- The last "OR 'a'='a" makes SQL statement true, bypassing security.

"SELECT * from passwords where user='" + username + "' and password='" + password +"';"

SQL injection

- SQL injection can be done on any queries
  - Search forms, web forms in general and even XML files (web services)
- You can load/modify system files
  - Password files, configuration files, etc..
- You can spawn external processes
  - Opening backdoors

LDAP injection

- Similar to SQL injection
- It is quite uncommon, but possible
- Example:

  https://www.mycompany.com/login.jsp&user=gipp
  a&password=letmein)(|(cn=*))

Some suggestions

- If you are going to install software or maintain a test site:
- Avoid default configurations
- Change passwords
  - Please do not use easy-to-guess password
- Do not let applications run as root
- Keep patches up-to-date
- Use "in-deep security" philosophy

Some suggestions

- If you are going to write code:
- Never trust user input (it also applies to XML in web services)
  - Always check fields for invalid characters such as & ; ` ' \ " | * ? ~ < > ^ ( ) [ ] { } $ \n \r
  - ... and escape them
  - Check input length
- Encrypt data and use safe connections
- Give the user the least privileges to access data

But remember: there is no 100% security!

(Gippa is watching you!)

Thank you!

Giuseppe Paternò
gpaterno@gpaterno.com