— Tiny errors, big losses: stories of 0wnage —

Fyodor Y.

Guard-Info

OWASP

October 25, 2008

The OWASP Foundation

1. Introduction
2. Web from attacker's perspective
3. Presentation notes
4. Small screw-ups
5. Funny Design Screwups
6. Convinient stupidities
7. Conclusion

Introducing Presenter

My name is Fyodor Yarochkin (often simply Fyodor Y.). I am not the nmap guy (snort faq Q1.2). I did some stuff for snort, xprobe and some other obscure public projects. I like to code and experiment with fun stuff.

I also do some penetration testings and application testings as my day-job. So this presentation summarises my experience.

Why hacking web is fun?

- easy, time-efficient
- you often see simple/stupid errors that are easily exploitable
- these "stupid" errors can't be found with scanners, because they are "custom" type of bugs (no other web application would have exactly the same bug)
- application logic bugs are easy to find manually if you understand the application, but nearly impossible - with existing automated tools
- firewalls lock everything but they don't lock web!

Presentation Notes

- We will discuss bugs, which I've seen in the wild.
- The bugs I selected for discussion are - prevalent bugs - i.e. bugs that I have seen in one or another variation over the years.
- I will not tell you where I saw them.
- Screenshots that I will show - replicate actual bugs that I saw, but they are not screenshots of customer systems that I worked with.
- Still.. I will tell you what was the bug. what was the impact of the bug and how 0wnage happened.
- what is the "right" way to mitigate the type of bugs or make the exploitation harder.

Presentation Notes

I've organized this presentation as a collection of *SCREW-UPS* that usually lead to web application or system comrpomise. Technically, the web application is just as secure as secure its weakest component or set of components.

Presentation Notes

So if the web application code is secure, but the deployment is bad, the total "security" of the web application is bad. Likewise, if the deployment is good, and machines are expensive, and alot of money were spent on firewalls, but the code was produced by jsp-in-21-day, the application security is still "bad".

The 0wnage strategy

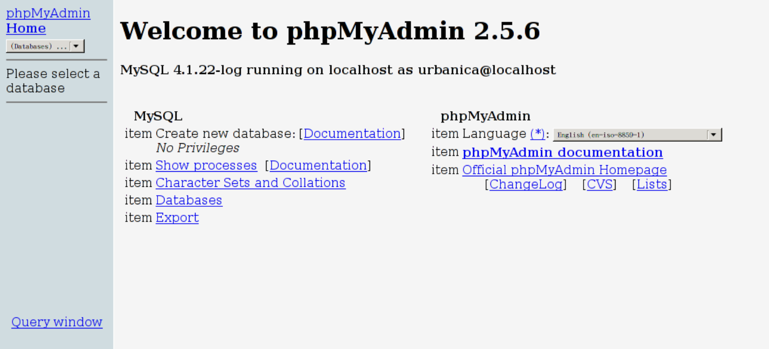
A web application 0wnage (or any 0wnage perse) doesn't usually happen because of one bug. Its more like solving a puzzle, using different sources of information.
These source of information could be anything, from web search posts, posts to the public forums to ldap data dumps, and application source code.

SCREWUP: Admin Forgetful

One of the things to search for on the web, is the stuff that admins forget about. And usually there are alot of juicy finds... and usually are easy ways in. Some examples..

Forgotten Admin interfaces

PHPMySQLAdmin, simple /admin/ interfaces to various stuff (cms systems, DB management.



phpMyAdmin
Home
(Databases) ...

Please select a database

Query window

Welcome to phpMyAdmin 2.5.6

MySQL 4.1.22-log running on localhost as urbanica@localhost

MySQL

item Create new database: [Documentation]
No Privileges
item Show processes [Documentation]
item Character Sets and Collations
item Databases
item Export

phpMyAdmin

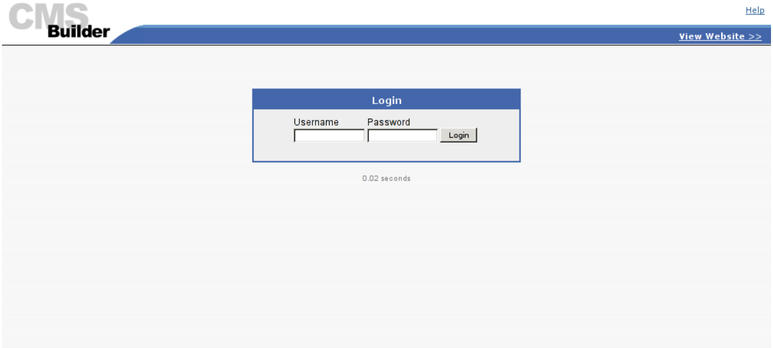
item Language (*): English (en-iso-8859-1)
item **phpMyAdmin documentation**
item Official phpMyAdmin Homepage
[ChangeLog] [CVS] [Lists]

Real Ownage Story

Unpassword'ed PHPMySQLAdmin interface provided access to mysql database that contained authentication credentials (user ids, passwords, and internal IP addresses) to a large number of internal systems.

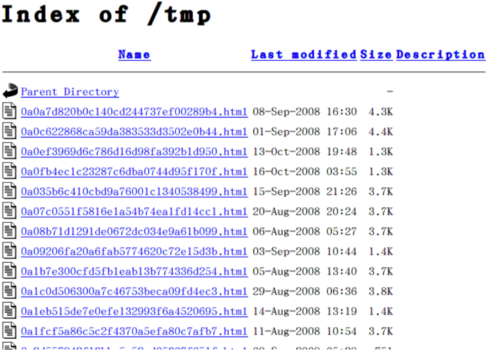
Forgotten Admin interfaces

You can find more admin interfaces by performing file name bruteforce and combining "admin" with variety of words related to web site. (i.e. cmsadmin). Some apache settings and IIS case insensitivity make the bruteforce tasks even easier.



and more...

While you're here, you can also check if admins keep other interesting folders. you can often find application source code, logs or other interesting stuff in it.



Index of /tmp

Name	Last modified	Size	Description
Parent Directory		-	
0a0a7d820b0c140cd244737ef00289b4.html	08-Sep-2008 16:30	4.3K	
0a0c622868ca59da383533d3502e0b44.html	01-Sep-2008 17:06	4.4K	
0a0ef3969d6c786d16d98fa392b1d950.html	13-Oct-2008 19:48	1.3K	
0a0fb4ec1c23287c6dba0744d95f170f.html	16-Oct-2008 03:55	1.3K	
0a035b6c410cbd9a76001c1340538499.html	15-Sep-2008 21:26	3.7K	
0a07c0551f5816e1a54b74ea1fd14cc1.html	20-Aug-2008 20:24	3.7K	
0a08b71d1291de0672dc034e9a61b099.html	06-Aug-2008 05:27	3.7K	
0a09206fa20a6fab5774620c72e15d3b.html	03-Sep-2008 10:44	1.4K	
0a1b7e300cfd5fb1eab13b774336d254.html	05-Aug-2008 13:40	3.7K	
0a1c0d506300a7c46753beca09fd4ec3.html	29-Aug-2008 06:36	3.8K	
0a1eb515de7e0efe132993f6a4520695.html	14-Aug-2008 13:19	1.4K	
0a1fcf5a86c5c2f4370a5efa80c7afb7.html	11-Aug-2008 10:54	3.7K	

Real Ownage Story

The tmp folder contained tars of various application components. ready to download. Consequential application vulnerabilities were discovered by application source code manual analysis. These were later exploited.

SCREWUP: passwords

Passwords to admin interface are easy to guess (*if any required at all!!*), if admins assume that the interfaces are not public. (admin/admin is a very common password)

SCREWUP: Forgotten source code

Application source code is usually a good find. Getting application source code is often easier than you think.

foo.com/myapp =¿ foo.com/myapp.tgz

You can also look for jars and other java classes. Java reverse engineering tools are reliable enough to get you readable application code. So, if you can find misconfigured WEB-INF folder and download classes from there, that's another good thing :)

Real Ownage Story

Application class files were downloaded from WEB-INF/classes folder. Decompiled and file upload vulnerabilities were found and exploited.

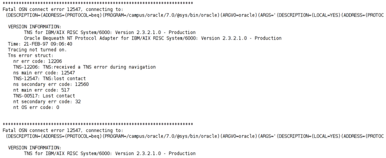
Test components and remote shell (left by developers?) was first found inside downloaded .tgz archive, and then used on actual system.

more source code

Another way to hunt for application source pieces is to search for .inc/.bak files.. - some editors create .bak automagically. And web admins often wouldn't remove these...

SCREWUP: log files

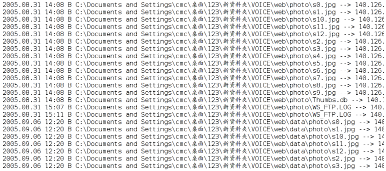
Web app admin uses oracle client to debug application
Client drops sqlnet.log file into current directory.
Current directory is ofen the web app directory.. accessible from outside



```
***********************************************************************
Fatal OSN connect error 12547, connecting to:
 (DESCRIPTION=(ADDRESS=(PROTOCOL=beq)(PROGRAM=/campus/oracle/7.0/@sys/bin/oracle)(ARGVO=oracle)(ARGS='(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOC

  VERSION INFORMATION:
        TNS for IBM/AIX RISC System/6000: Version 2.3.2.1.0 - Production
        Oracle Bequeath NT Protocol Adapter for IBM/AIX RISC System/6000: Version 2.3.2.1.0 - Production
  Time: 21-FEB-97 09:06:40
  Tracing not turned on.
  Tns error struct:
    nr err code: 12206
    TNS-12206: TNS:received a TNS error during navigation
    ns main err code: 12547
    TNS-12547: TNS:lost contact
    ns secondary err code: 12560
    nt main err code: 517
    TNS-00517: Lost contact
    nt secondary err code: 32
    nt OS err code: 0


***********************************************************************
Fatal OSN connect error 12547, connecting to:
 (DESCRIPTION=(ADDRESS=(PROTOCOL=beq)(PROGRAM=/campus/oracle/7.0/@sys/bin/oracle)(ARGVO=oracle)(ARGS='(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOC

  VERSION INFORMATION:
        TNS for IBM/AIX RISC System/6000: Version 2.3.2.1.0 - Production
```

SCREWUP: log files

admin uses WSFTP to upload web content..



```
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s0.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s1.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s10.jpg --> 140.126
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s11.jpg --> 140.126
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s12.jpg --> 140.126
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s2.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s3.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s4.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s5.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s6.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s7.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s8.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\s9.jpg --> 140.126.
2005.08.31 14:08 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\Thumbs.db --> 140.
2005.08.31 15:07 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\WS_FTP.LOG --> 140.
2005.08.31 15:11 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\photo\WS_FTP.LOG --> 140.
2005.09.06 12:20 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\data\photo\s0.jpg --> 140
2005.09.06 12:20 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\data\photo\s1.jpg --> 140
2005.09.06 12:20 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\data\photo\s10.jpg --> 14
2005.09.06 12:20 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\data\photo\s11.jpg --> 14
2005.09.06 12:20 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\data\photo\s12.jpg --> 14
2005.09.06 12:20 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\data\photo\s2.jpg --> 140
2005.09.06 12:20 B C:\Documents and Settings\cmc\桌面\123\新資料夾\VOICE\web\data\photo\s3.jpg --> 140
```

This may give some details on internal network, IPs etc. Not very useful by itself, but often helpful in combination with other bugs

Design Bugs

The other set of interesting bugs to go after is design bugs. These are usually hard, if not ipossible to find with application scanners. Therefore they are a good hunt :)

Embedded Data

Some developers think it is a very neat idea to allow your ActiveX component to authenticate users by connecting to DB directly. But implementations may be wrong..

Embedded Data

ActiveX connects to database. Executes SQL query to validate user and ether logs user in or displays error message.
The caveat of such design is that database authentication credentials are embedded within the ActiveX binary and can be easily extracted by ”curious” user.

Broken Authentication Schemes

An application uses two components that run under two incompatible (thank you, industry) application servers. One application provides authentication service. The other application needs to verify that user was authenticated.
How people do it...

Broken Authentication Schemes

The second application simply verifies that parameter COOKIE=blah is passed to the application. The application has no way to know the actual cookie thu, so it "assumes" that the cookie is good.

Real Ownage Story

The application had to use some other host for file uploading functions. The original application used WebLogic, while file upload server was an IIS system. Not only you could specify where to upload the files, but also you did not have to log into the application to do it.

Another Broken Authentication Schema

PageA asks for password and submits to PageB
PageB takes password, validates, and submits to PageC
PageC sets session to authenticated and redirects to Main..
How do you bypass this auth. thing? :)

Broken Session Tracking Mechanisms

Privileged access. How does application know if that is admin session?

Broken Session Tracking Mechanisms

Set-Cookie: Admin=1

heh..

More funny stuff (real-life examples)

URL Embedded passwords - what could be more convinient.. ;-)

```
setCookie('OTPlogin', '***censored****', expires);
var URLList =
'https://**censored**/user.userlogin?username=**censored*
&passwd=decc**passcut**&auth=radius
&clientip=**huhu**&custom=free';
```

This type of URLs cries for some manipulation ;-)

Funnys

Access control - the way not to do it:

Don'ts in implementing access control

"I only show you menus, which you are allowed to see". (and you can guess the rest, .. especially when application component names are sequential).

Even worse method to implement the same thing

"I only show you menus, which you are allowed to see". (and you see the rest inside HTML commented code).

Real Ownage Story

Hidden application components were guessed because application folder names were sequential (numbers). No access control check was performed anywhere within the application except for the main menu. Once "admin"-privileged application components were found, we had full control of the application (including ability to create/remove users, alter data and so on)

File Upload is usually a huge hole

File uploads get worse when file upload path is within web directory.
This is usually done for file linking convinience (you can simply include ¡a href=/uploads/blargh¿ .. links to the file.
But there are just too many things that can go wrong with this file upload mechanism. (depends not only on application coding practice, but also on proper system hardening and web server secure configuration to function in secure way).

File Upload is usually a huge hole

Frequent exploitation scenarios: file extension path manipulation, playing with difference of multi file extrension handling by web server and application upload component, access to the files uploaded by other users and so on.

Real Ownage Story

The file upload function was implemented within 3 steps. intermediate page kept relative path of uploaded file. It was discovered that it was possible to simply modify path within the "intermediate" page, to upload files into web server webroot folder.

Worse than SQL injection

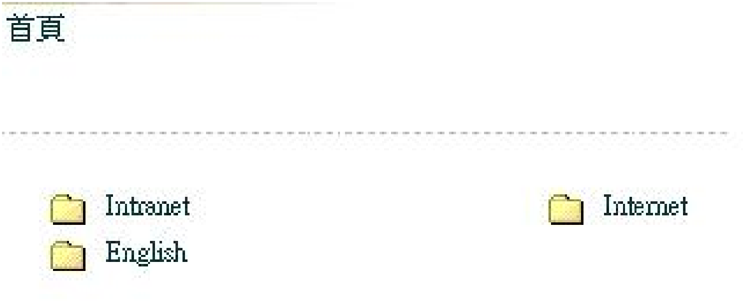
SQL code as part of HTML "hidden" parameter

Real Ownage Story

The developers thought it was very convinient to have a single jsp script to display nicely the application data. The SQL code to select data was passed to teh script as "hidden" parameter. With simple parameter changes it was possible to completely compromise the application, not only querying, but also modifying and inserting new data (including application users) into the backend database.

Saving Cash on hardware

Intranet and Internet webs on the same box, sharing the same Content Management System(s)



首頁

Intranet

English

Internet

Other amusing web configurations

FTP and WEB roots map to the same root folder

Real Ownage Story

ftp passwords were reused from other compromised system. Files uploaded into ftp folder, and executed through web request.

Remote Desktop Applications

Remote Desktop Applications, such as Tarantella/Sun Secure Global Desktop,Citrix are usually good way in, if you can find user ids/passwords. You are usually restricted in what you can use (or even bound to a single application), but finding local shell execution possibilities usually is not an issue.

Real Ownage Story

Userids were extracted from directory service. Passwords were automagically guessed against ftp servers. these user ids and passowrds were reused to access remote desktop applications, compromise underlying systems and internal segments.
good thing - you're already inside intranet usually, once you're on remote desktop application.

Broken User input validation

It is very convinient to validate user input and filter "wrong" characters by using javascript code that executes when submit button is pressed...
but this is badly wrong (and useless as well ;-))

Real Ownage Story

The file upload function was performing validation of uploading file extensions by checking (via javascript) whether these files were any of .exe/.com/.php/.jsp files and would only "submit" form in case if the validation test passed.

Checks were bypassed by using perl script to upload files ;-)

Sum-up on discussed bugs

Web Applications are complex systems. A large number of factors affect web application security. There's no "silver bullet" solution for all the problems. All the discussed earlier bugs can be classified into following groups: Design flaws, Implementation flaws, Coding bugs, Configuration and Deployment Flaws, Maintenance issues. Proper process that utilizes automated tools, and manual analysis (performed by human brain) is the key attribute in creating and maintaining properly secured web applications.

Classification

- Design flaws - the wrong or erronous decisions, which were made during application design phase.
- Implementation flaws - even if the original design ideas were security-wise correct, decision on component implementation methods may still be erroneous (packages to use, required system configurations)
- Coding bugs - bugs that appear as coding errors. Even if the intention was correct, the way the intention was converted into code, might be wrong.
- Configuration and Deployment flaws - even if the web application was designed, and developed securely, the actual deployment of web application may lead to security problems. Missed configuration options, forgotten source code.
- Maitenance issues - web application is usually a live system. Constant system changes and modifications may leave security-relevant ”traces”, which could be exploited by attacker

Design flaws

It is nearly impossible to automate process of analysing and reviewing design flaws. The most effective ways to identify design flaws are peer-reviews by domain and security experts, application architecture reviews and so on.
Application manual analysis and testing by application security testing teams may also be helpful to identify and mitigate (at higher cost, of already developed application) certain design issues.
Knowledge of basic security principles (trusted vs. untrusted data), analaysis of data flows from security viewpoint are also helpful to avoid mistakes at design level.

Implementation flaws

Some of the implementation flaws might be picked by automated code analysis and testing tools. Others have to manually evaluated by system security experts. Detailed understanding of functionality of used application components is usually required to avoid mistakes at this stage.

Coding bugs

The majority of automated code analysis software and blackbox application testing software are most effective at this level, as there are relatively robust technologies to identify and often patch coding bugs automagically :)

Configuration and Deployment flaws

Use of automated blackbox application testing software along with manual application is usually helpful approach to identify and mitigate configuration and deployment problems.

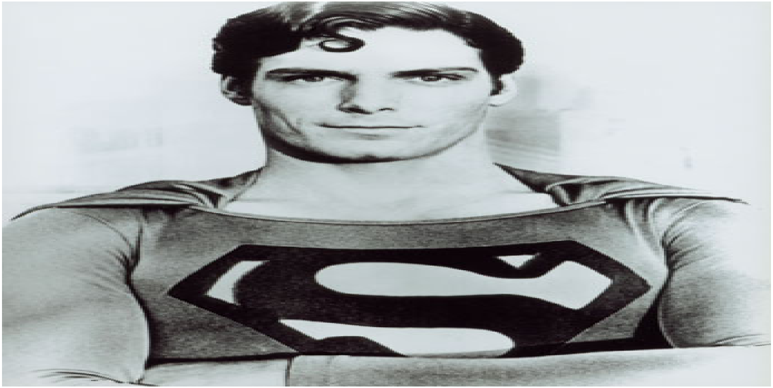
Often, existence of proper configuration and deployment security policies, system baseline security policies is a great factor in addressing possible security problems at this level.

Automated tools exist to validate system compliance to base security policies. Penetration Tests and Application Security Tests may also be helpful to identify configuration errors, which can't be detected automatically.

Maintenance issues

I believe the existence and application of security policies is the key factor to avoid problems during application maintenance phases. Productional systems should be periodically validated for existence of possible misconfigurations, or other security threats that were introduced by application changes.
Some of these "compliance" checks may actually be automated (for example it is very easy to automade checks for .bak files, log files or application source code within the application web root folder).
Other checks could be performed using automated tools or periodical manual review.

Questions?



Or answers.. comments... :-)
fygrave at gmail dot com
And thank you very much ;-)