PHP Security Workshop

Bruno Mairlot

PHP Security Workshop

- Authenticating Variable : Never trust the web
- Using the underlying Web Server as a security measure
- Safe Mode review and analysis
- PHP Streams : Making the SSL/TLS connection easy
- PHP SSL Module : Automating Key generation and Certificate Signing
- Security consideration with Dreamweaver's PHP code : How your firewall is bypassed without you knowing it

Never Trust the Web

- HTTP and HTML have been designed as for public data.
- It is the things you don’t control that are at the source of attack
- Bots and Agent exists and are easy to use
- You can’t hide form parameter, cookies,…
- Do not trust JavaScript check test, these data can be easily faked

Never Trust the Web

1. Stick with **register_globals = Off**
2. Always consider $_REQUEST is untrustworthy, this is a state of mind
3. Check the type of your variable (is_numeric(), is_integer(), casting,…) (1.php)
4. Be careful with conversion function like settype() (2.php)

Never Trust The Web
What is Trustworthy

- $_SERVER, $_ENV and $_SESSION

Never Trust the Web
Make $_REQUEST trustworthy

- Using one-way hash signature (md5, sha1, sha256 or sha512)
- Create a string concatenating the different variables you want to trust, add a private key (whatever) and hash it :

```
<a href="dest.php?var1=value1&auth=<?php echo sha1($value1.$key)?>">
```

- (3.php)
- Then, check your signature with the different variables. These must be equal.

```
If(sha1($_REQUEST["var1"].$key)==$_REQUEST["auth"]{
        // $_REQUEST["var1"] has not been tampered
}
```

- This technique ensure that the value of var1 is authentic. (mail,)

Never Trust the Web

- When handling filesystem functions, be sure to hardcode the root path and to check variables
- Your friends are :
  - $_SERVER[“DOCUMENT_ROOT”]
  - __FILE__
  - dirname() , basename() , realpath()

Never Trust the Web

- Check your data … and check again.
- Example of tampering with a query :

```
$query  = "SELECT * FROM products LIMIT 20,$_REQUEST[offset]";
$res = mysql_query($query,$dbh);
```

- What if $_REQUEST[“offset”] is actually “0 ; SELECT user,password FROM user”

Note : This trick doesn’t work with MySQL

Web Server

- ini_set() can override your carefully crafted php.ini file.
- Use the php_admin_flag and php_admin_value Apache directive to enforce some security feature and/or to make sure you don’t expose security code.

```
<VirtualHost *:80>
        DocumentRoot "C:/Program Files/Apache Group/Apache/vhost7"
        ServerName arpenteur7.maehdros.local
        php_admin_value mysql.default_host localhost
        php_admin_value mysql.default_user bruno
        php_admin_value mysql.default_password password
</VirtualHost>
```

- (5.php) and (5.php)
- This doesn’t work with SQL_SAFE_MODE
- Be careful, phpinfo() will expose your configuration value (http://arpenteur7.maehdros.local/phpinfo.php) (you should use *disable_functions* ini directive)

Web Server

- Good php_admin_value practice :
  - max_execution_time
  - safe_mode
  - safe_mode_allowed_env_vars (be careful it is a prefix list => if empty all environment are writeable)
  - safe_mode_protected_env_vars
  - safe_mode_include_dir
  - safe_mode_exec_dir
  - open_basedir (is not safe_mode dependant)

Safe Mode

- What is Safe Mode
  - Safe Mode alters the behavior of a lot of system-related function (fopen(), exec(), move_uploaded_file(),…) and some are completely disabled (dl(), ``,
  - As there are many UID check (and GID) against the system, it is a good practice to enable suexec on Apache and dedicate one user for each VirtualHost, though this might bec complicated to set up.
  - Safe Mode is a sort of attempt to chroot but at the logical level
  - Safe Mode is a step toward security but…

PHP Streams

- PHP Streams are fantastic way to abstract file location and connection.

```
/* Read local file from /home/bar */
$localfile = file_get_contents("/home/bar/foo.txt");
$localfile = file_get_contents("file:///home/bar/foo.txt");

/* Read remote file from www.example.com using HTTP */
$httpfile  = file_get_contents("http://www.example.com/foo.txt");
$httpsfile = file_get_contents("https://www.example.com/foo.txt");

/* Read remote file from ftp.example.com using FTP */
$ftpfile  = file_get_contents("ftp://user:pass@ftp.example.com/foo.txt");
$ftpsfile  = file_get_contents("ftps://user:pass@ftp.example.com/foo.txt");
```

PHP Streams

- Creating an SSL connection is as easy as opening a standard one :

```
$sock = fsockopen("ssl://secure.example.com", 443, $errno, $errstr, 30);
if (!$sock){
        die("$errstr ($errno)\n");
}
```

- Different secure streams are :
  - https://
  - ftps://
  - ssl:// - sslv2:// - sslv3:// - tls://
  - ssh2.shell:// - ssh2.exec:// - ssh2.tunnel:// - ssh2.sftp://

PHP Streams

- To use Certificate and Private Key, you may use the function stream_context_set_option() to specify the following context options :
  - verify_peer : Require verification of peer
  - cafile (or capath) : Certificate of Authority to use with verify_peer
  - allow_self_signed
  - local_cert : You PKCS12 certificated
  - passphrase : the passphrase of your private key
  - CN_match : Common Name expected

OpenSSL Module

- On Win32 system, make sure the OPENSSL_CONF environment variable is set and points to a valid openssl.cnf (there is one included with PHP)
- When using Certificate, be sure to check the mandatory fields…

OpenSSL Module

- Creating a Private Key :

```
$privkey = openssl_pkey_new();
openssl_pkey_export($privkey, $privatekey);
echo $privatekey;
```

- 6.php

OpenSSL Module

- Create a Certificate Signing Request

```
$dn = array(
            "countryName" => 'LU',
            "stateOrProvinceName" => 'Luxembourg',
            "localityName" => 'Luxembourg',
            "organizationName" => 'Hack.lu',
            "organizationalUnitName" => 'Hack.lu',
            "commonName" => 'Hack.lu',
            "emailAddress" => 'demo@hack.lu'
            );
$csr = openssl_csr_new($dn, $privkey,$configarg);
openssl_csr_export($csr,$csrStr,true);
```

- 7.php

OpenSSL Module

- Signing a Certificate Request
  - Have your CA Certificate and Key ready
  - Load the CSR data (it is a string)

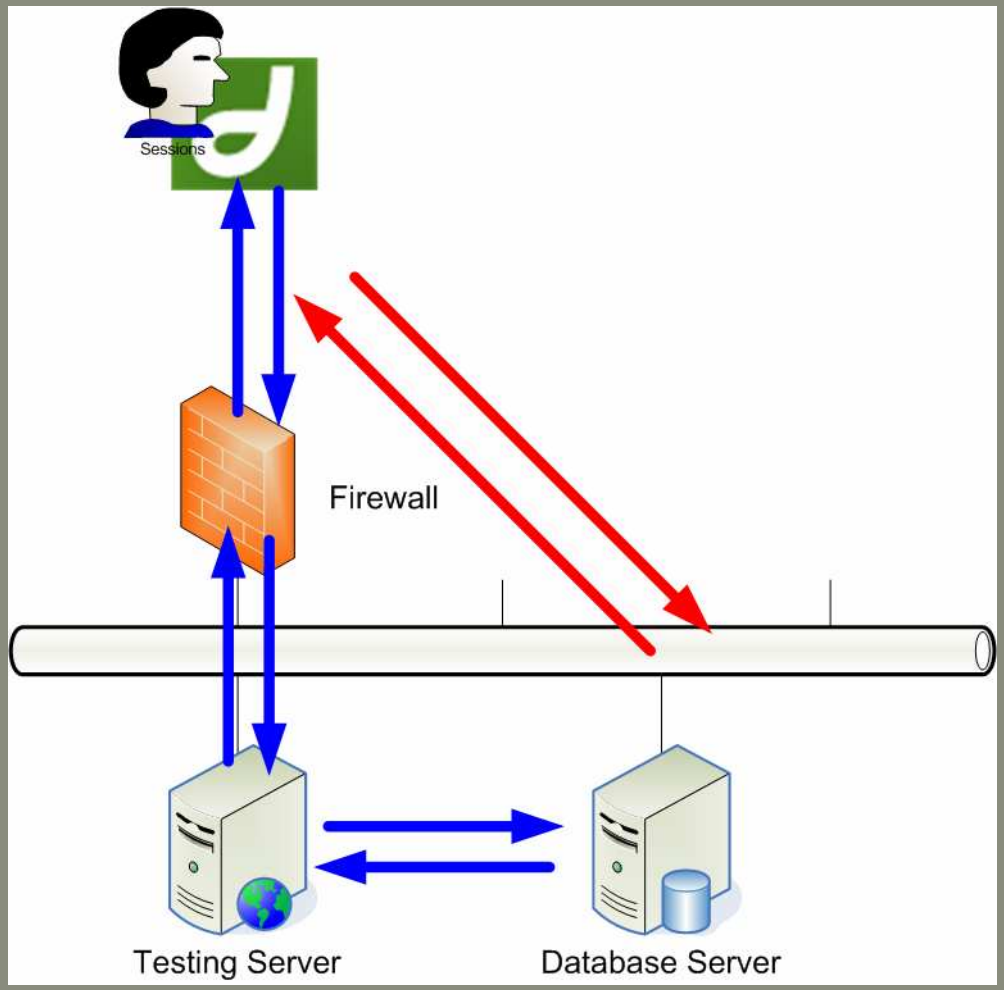
```
$cacert=file_get_contents("brunomairlot.crt");
$cakey = file_get_contents("brunomairlot.key");

// Loading CSR data
$csrdata = file_get_contents("hack.csr");
$certificate = openssl_csr_sign($csrdata, $cacert, $cakey, 365);
// Exporting Certificate
openssl_x509_export($certificate, $certout);
```

  - (8.php)

Dreamweaver

- Dreamweaver (since UltraDev) use a HTTP mechanism to interact with the database through request to the ‘Testing Server’
- This mechanism is stored in the ‘_mmServerScripts’ directory that DW doesn’t show. This directory contains the script called “MMHTTPDB.php” that is the main request URL

Dreamweaver



Dreamweaver

- See MMHTTPDB.php
- Remove Connection Scripts

If we have time…

- Other topics : defeating bots
- Authenticating User