Secure coding training

Securing applications on a higher level

Gerard Frankowski, PSNC

Poznań, 22-23 June 2010

connect • communicate • collaborate

Introduction

- This presentation is intended to provide high-level advices in order to produce secure applications
- The following issues will be covered:
  - Responsibility for secure systems
  - Session management
  - Appropriate error handling
  - Information disclosure
  - Resource discovery
  - Coding conventions
  - And several others (very briefly!)

Responsibility for secure systems

connect • communicate • collaborate

Who is responsible for security?

- Manager / Project leader
  - The idea – what will our application do?
  - The budget
  - Usability and efficiency / security tradeoff
- The developers
  - Appropriate development of the project
  - Limited by the project scope…
- The administrators/security specialists
  - Configuration and access to services
  - Configuration of network, servers, access devices
  - Limited by their systems and security budget…

Problems of investing in security

- Investing in security does not bring immediate and noticeable gain!
  - We just spend some money, that we could save and maybe nothing bad would happen
  - The costs of security on all levels are usually relatively high
- The problem begins when we do not take care of security, and the bad things really happen
  - Usually there is no way back

Security – the problem of users?

- In general, we cannot expect specialized knowledge on IT security from users
  - True for mass e-services, but also e.g. for research institutions services users
- The user should have some basic awareness and applications should support that
  - Appropriate configuration facilities
  - Usability of security features
- Security vulnerabilities in the source code would affect even the best educated users!

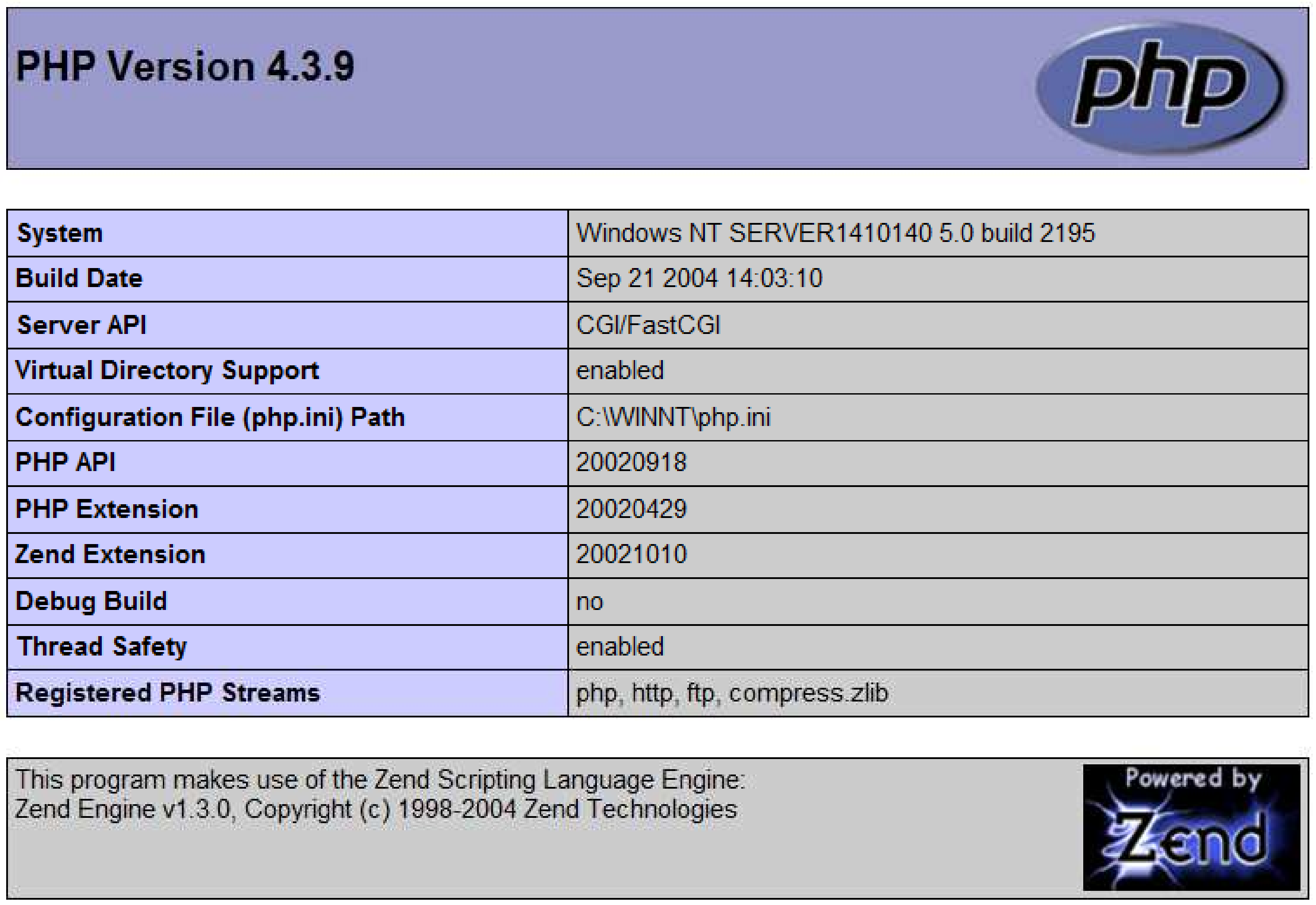
Information disclosure

connect • communicate • collaborate

Information disclosure

- Every user should have only information that he or she really needs for legitimate purposes
  - The above is directly derived from the minimum privileges principle
- All breaches of that rule may be called Information Disclosure
- An attacker tries to collect maximum amount of information before the attack
  - Information disclosure vulnerabilities are great help for that purpose

Information disclosure – example 1 phpinfo()

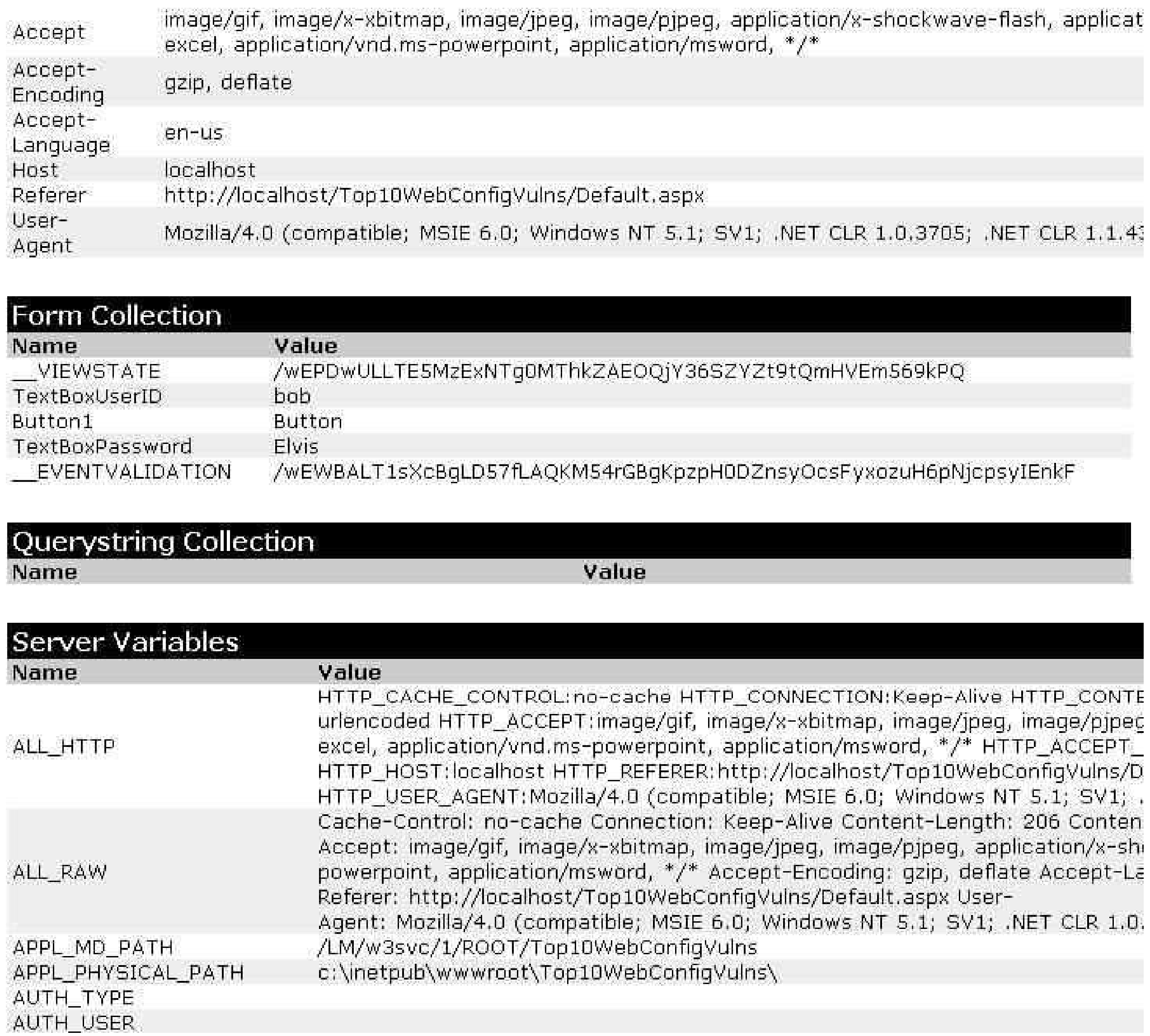


PHP Version 4.3.9

System	Windows NT SERVER1410140 5.0 build 2195
Build Date	Sep 21 2004 14:03:10
Server API	CGI/FastCGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINNT\php.ini
PHP API	20020918
PHP Extension	20020429
Zend Extension	20021010
Debug Build	no
Thread Safety	enabled
Registered PHP Streams	php, http, ftp, compress.zlib

This program makes use of the Zend Scripting Language Engine:
Zend Engine v1.3.0, Copyright (c) 1998-2004 Zend Technologies

Information disclosure – example 2 trace.axd



Accept	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, applicat excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Encoding	gzip, deflate
Accept-Language	en-us
Host	localhost
Referer	http://localhost/Top10WebConfigVulns/Default.aspx
User-Agent	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.0.3705; .NET CLR 1.1.4

Form Collection

Name	Value
__VIEWSTATE	/wEPDwULLTE5MzExNTg0MThkZAEOQjY36SZYZt9tQmHVEm569kPQ
TextBoxUserID	bob
Button1	Button
TextBoxPassword	Elvis
__EVENTVALIDATION	/wEWBALT1sXcBgLD57fLAQKM54rGBgKpzpH0DZnsyOcsFyxozuH6pNjcpsyIEnkF

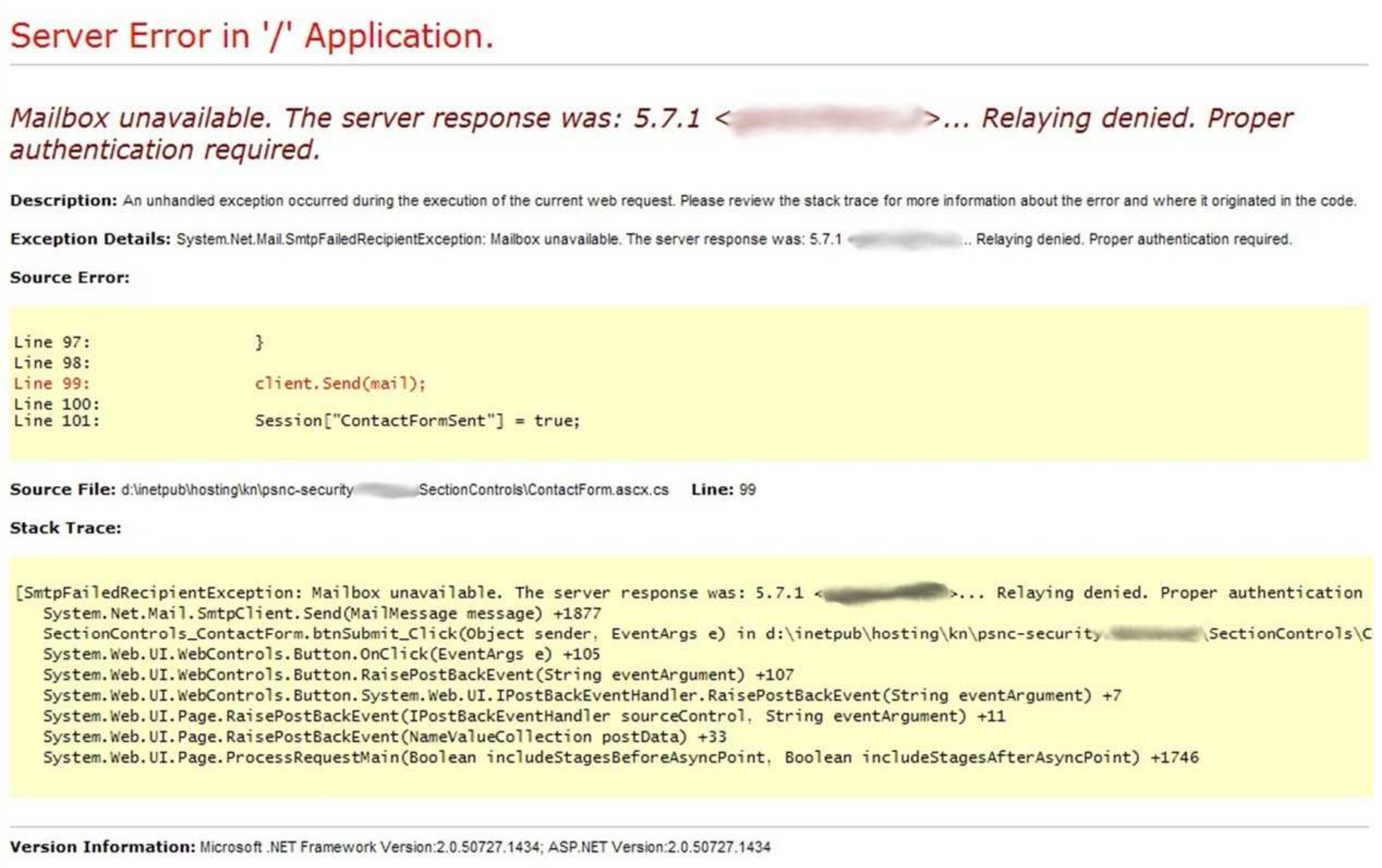
Querystring Collection

Name	Value

Server Variables

Name	Value
ALL_HTTP	HTTP_CACHE_CONTROL:no-cache HTTP_CONNECTION:Keep-Alive HTTP_CONTE urlencoded HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg excel, application/vnd.ms-powerpoint, application/msword, */* HTTP_ACCEPT_ HTTP_HOST:localhost HTTP_REFERER:http://localhost/Top10WebConfigVulns/D HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .
ALL_RAW	Cache-Control: no-cache Connection: Keep-Alive Content-Length: 206 Conten Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-sh powerpoint, application/msword, */* Accept-Encoding: gzip, deflate Accept-La Referer: http://localhost/Top10WebConfigVulns/Default.aspx User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.0.
APPL_MD_PATH	/LM/w3svc/1/ROOT/Top10WebConfigVulns
APPL_PHYSICAL_PATH	c:\inetpub\wwwroot\Top10WebConfigVulns\
AUTH_TYPE	
AUTH_USER	

Information disclosure – example 3
ASP.net improperly configured



Server Error in '/' Application.

Mailbox unavailable. The server response was: 5.7.1 <[redacted]>... Relaying denied. Proper authentication required.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Net.Mail.SmtpFailedRecipientException: Mailbox unavailable. The server response was: 5.7.1 [redacted].. Relaying denied. Proper authentication required.

Source Error:

```
Line 97:                    }
Line 98:
Line 99:                    client.Send(mail);
Line 100:
Line 101:                   Session["ContactFormSent"] = true;
```

Source File: d:\inetpub\hosting\kn\psnc-security[redacted]SectionControls\ContactForm.ascx.cs   **Line:** 99

Stack Trace:

```
[SmtpFailedRecipientException: Mailbox unavailable. The server response was: 5.7.1 <[redacted]>... Relaying denied. Proper authentication
   System.Net.Mail.SmtpClient.Send(MailMessage message) +1877
   SectionControls_ContactForm.btnSubmit_Click(Object sender, EventArgs e) in d:\inetpub\hosting\kn\psnc-security[redacted]\SectionControls\C
   System.Web.UI.WebControls.Button.OnClick(EventArgs e) +105
   System.Web.UI.WebControls.Button.RaisePostBackEvent(String eventArgument) +107
   System.Web.UI.WebControls.Button.System.Web.UI.IPostBackEventHandler.RaisePostBackEvent(String eventArgument) +7
   System.Web.UI.Page.RaisePostBackEvent(IPostBackEventHandler sourceControl, String eventArgument) +11
   System.Web.UI.Page.RaisePostBackEvent(NameValueCollection postData) +33
   System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint) +1746
```

Version Information: Microsoft .NET Framework Version:2.0.50727.1434; ASP.NET Version:2.0.50727.1434

What may be disclosed?

- System paths (and thus OS family/type)
- Configuration details
- Software detailed versions
  - Known exploits for that versions may be then tested
- Application structure and behavior
- Source code details
  - File, line number, function
  - Sometimes even code snippets
- User data
- And more…

Responsibility distribution for information disclosure

- This is an area where system and server administrators have relatively much to do
  - Web server configuration (e.g. Apache, IIS)
  - Scripting engine configuration (e.g. PHP, ASP.NET)
  - OS & database configuration
  - Local firewall
- However, the mistakes of developers may contribute to information disclosure as well
  - Application mechanisms that reveal too much information
  - Too much trust that the server administrators will not cause information disclosure

How developers may disclose too much information

- Indirectly – allowing drawing conclusions
  - Inappropriate functionality – e.g. different error messages for bad username and bad password
  - Timing issues – e.g. different computation times for certain data sets may cause disclosing the crypto key
- Directly
  - Application versions/banners
  - System paths
  - Error messages in cryptographic protocols
  - Too detailed error explanations

Too detailed error explanations

- Sensitive data in displayed messages, e.g.

  Password *blah1234* provided for user *blah* incorrect

  - If this is a client-server application (incl. Web), the message may be sniffed – better would be:

  Your password is incorrect. Please try again

  - Even the user name is not necessary – might cause enumerating users (and the user knows it anyway)
- The user should not exactly know your technical problem – but just:
  - That *something* has just happened
  - What he or she has got to do

Countermeasures

- The detailed information should be saved to logs only
- Give the user the minimum amount of information
  - E.g. only a filename instead the full path
  - A good idea would be assigning an ID to problems where the user might need some interaction

    An unexpected problem occurred. Please contact our support at [address] and refer to problem ID [id]
- Timing issues – in sensitive applications consider adding short random *sleep()* to computations
- In case of cryptographic protocols, consider terminating the transmission instead an error message
- Don't always trust administrators;)

Resource discovery

connect • communicate • collaborate

Resource discovery

- Leaving any functionality (usually not available directly) in easy-to-guess places
  - Usually applied to Web applications
- May lead to information disclosure and is closely related with (but not equivalent to) it
- Often performed by attackers in the initial phase of the attack
- Especially easy in applications based on known frameworks/CMSs

Resource discovery – phpinfo

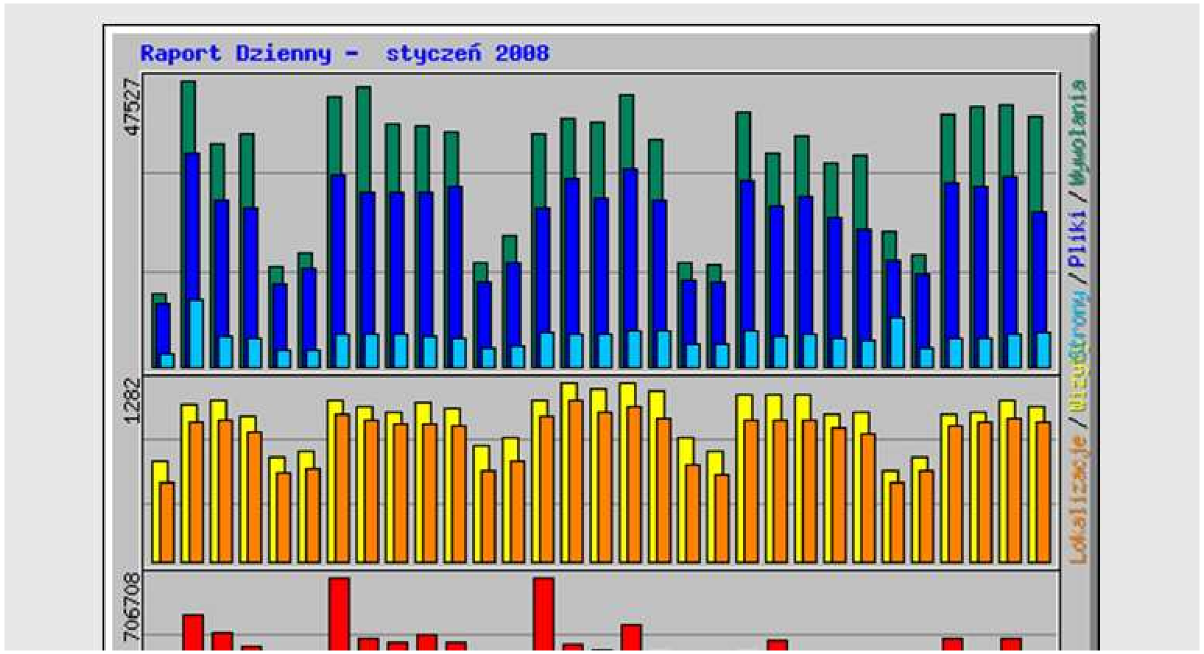
- Many PHP developers use a short file with phpinfo() call to check whether their fresh application works
  - Yet many of them forget to remove it later
- Where the above may be usually found?
  - phpinfo.php
  - test.php
  - info.php
  - php.php
  - a.php
  - p.php
- A good example of resource discovery – results in information disclosure

Resource discovery – other examples

- Administration interface
  - Often found under /admin or /administrator directories
  - phpMyAdmin and alike common directories
  - May ease attacks on the administrator account
- Statistics pages
  - Usually /stats or /statistics
- Old or test functionality
  - Check /test, /old, /backup
- Documents
  - May be often found under /doc(s), /document(s), /upload(s) and alike
  - Real gain for attackers usually requires directories indexing switched on

Resource discovery – example Statistics page

- A bank website, Webalizer pages under /stats
  - Often clicked links (including some to “hidden” administration interface and internal documents)
  - Several internal IP addresses



Countermeasures

- Besides avoiding the most apparent errors (like leaving phpinfo()), the situation is not so easy
  - You will not rewrite Content Management Systems to change directory names
  - Remember that the majority of the Web application structure may be read from the HTML code
- But what you can do
  - If you have modules that are not linked to anywhere else, do not place them under default / intuitive names
  - Never leave test or unnecessary functionality, especially under directories like *test*, *old*, etc.

Coding conventions

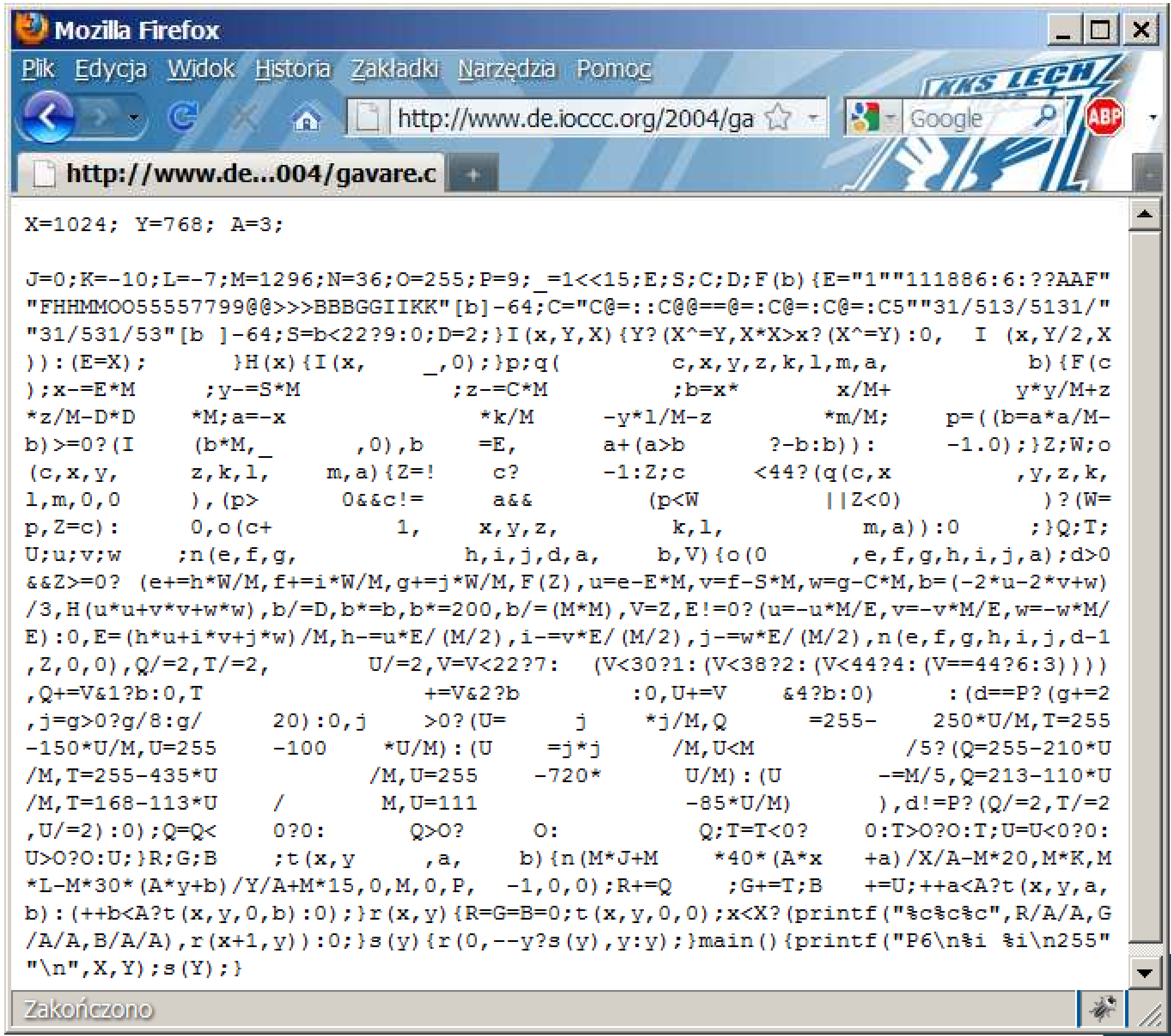
Introduction to coding conventions

- Any convention that produces solid code is good ;)
  - Although there are ones that make the code easier to read (we personally prefer such ones)
- Significant facets for understandable source code
  - Naming
  - Function specifications
    - *What does it do?*
    - *Input and output parameters*
    - *Caveats*
  - Comments through the code are welcome as well
- Especially Java has got strongly accepted conventions

Why the code clarity does matter?

- Remember that someone else may have to read and/or modify the code
  - If the code is easier to read, a security specialist will prepare a faster and more accurate opinion
  - It the code is harder to understand, another developer, who will be working on it, will produce more bugs
- Debian OpenSSL PRNG vulnerability (published in May 2008) was caused by commenting a line that ”seemed” unnecessary by Debian developers
  - The line filling a PRNG buffer with random data was not provided with any comment about its significance
  - The degree of data randomness decreased drastically, which produced weak crypto keys

International Obfuscated C Code Contest ;)


Mozilla Firefox
Plik Edycja Widok Historia Zakładki Narzędzia Pomoc
http://www.de.ioccc.org/2004/ga
Google
http://www.de...004/gavare.c
Zakończono

```
X=1024; Y=768; A=3;

J=0;K=-10;L=-7;M=1296;N=36;O=255;P=9;_=1<<15;E;S;C;D;F(b){E="1""111886:6:??AAF"
"FHHMMOO55557799@@>>>BBBGGIIKK"[b]-64;C="C@=::C@@==@=:C@=:C@=:C5""31/513/5131/"
"31/531/53"[b ]-64;S=b<22?9:0;D=2;}I(x,Y,X){Y?(X^=Y,X*X>x?(X^=Y):0,  I (x,Y/2,X
)):(E=X);      }H(x){I(x,    _,0);}p;q(        c,x,y,z,k,l,m,a,          b){F(c
);x-=E*M     ;y-=S*M           ;z-=C*M         ;b=x*        x/M+         y*y/M+z
*z/M-D*D    *M;a=-x             *k/M     -y*l/M-z        *m/M;   p=((b=a*a/M-
b)>=0?(I    (b*M,_     ,0),b    =E,      a+(a>b      ?-b:b)):    -1.0);}Z;W;o
(c,x,y,     z,k,l,   m,a){Z=!    c?      -1:Z;c     <44?(q(c,x        ,y,z,k,
l,m,0,0     ),(p>     0&&c!=     a&&        (p<W         ||Z<0)         )?(W=
p,Z=c):     0,o(c+        1,    x,y,z,        k,l,         m,a)):0     ;}Q;T;
U;u;v;w    ;n(e,f,g,           h,i,j,d,a,    b,V){o(0     ,e,f,g,h,i,j,a);d>0
&&Z>=0? (e+=h*W/M,f+=i*W/M,g+=j*W/M,F(Z),u=e-E*M,v=f-S*M,w=g-C*M,b=(-2*u-2*v+w)
/3,H(u*u+v*v+w*w),b/=D,b*=b,b*=200,b/=(M*M),V=Z,E!=0?(u=-u*M/E,v=-v*M/E,w=-w*M/
E):0,E=(h*u+i*v+j*w)/M,h-=u*E/(M/2),i-=v*E/(M/2),j-=w*E/(M/2),n(e,f,g,h,i,j,d-1
,Z,0,0),Q/=2,T/=2,      U/=2,V=V<22?7:   (V<30?1:(V<38?2:(V<44?4:(V==44?6:3))))
,Q+=V&1?b:0,T               +=V&2?b        :0,U+=V    &4?b:0)    :(d==P?(g+=2
,j=g>0?g/8:g/    20):0,j    >0?(U=     j    *j/M,Q     =255-    250*U/M,T=255
-150*U/M,U=255   -100    *U/M):(U    =j*j     /M,U<M          /5?(Q=255-210*U
/M,T=255-435*U          /M,U=255    -720*      U/M):(U      -=M/5,Q=213-110*U
/M,T=168-113*U   /       M,U=111               -85*U/M)      ),d!=P?(Q/=2,T/=2
,U/=2):0);Q=Q<   0?0:      Q>O?     O:          Q;T=T<0?   0:T>O?O:T;U=U<0?0:
U>O?O:U;}R;G;B   ;t(x,y     ,a,    b){n(M*J+M    *40*(A*x  +a)/X/A-M*20,M*K,M
*L-M*30*(A*y+b)/Y/A+M*15,0,M,0,P,  -1,0,0);R+=Q   ;G+=T;B  +=U;++a<A?t(x,y,a,
b):(++b<A?t(x,y,0,b):0);}r(x,y){R=G=B=0;t(x,y,0,0);x<X?(printf("%c%c%c",R/A/A,G
/A/A,B/A/A),r(x+1,y)):0;}s(y){r(0,--y?s(y),y:y);}main(){printf("P6\n%i %i\n255"
"\n",X,Y);s(Y);}
```

A commentary convention we like

```
/*!
 *  Check a string for the occurrence of certain characters. This is specifically for
 *  the checking of environment variables that make it to a log file. The newline
 *  character '\n' is not allowed to appear in it as it allows reformatting the
 *  intended layout of the log file and may cause a potential exploitation.
 *
 *  \param variable  Variable to be checked
 *
 *  \return true, if the variable is found to be sane, false otherwise.
 */
int glexec_sane_variable(const char *variable)
```

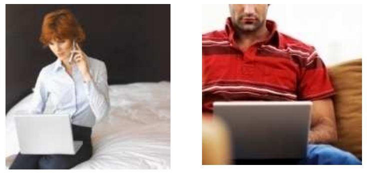
When to avoid too much comments?

- Static pages in Web applications!
  - Static pages and JavaScript files may be easily downloaded and studied by attackers
- Consider using:
  - Tools that will strip out all comments from the release version of the code
  - (For more sensitive applications) some techniques of code obfuscation – even if it contradicts, what we have already said

Session management

connect • communicate • collaborate

HTTP session



- HTTP is stateless
  - Simplicity is great, but not always
  - HTTP cannot differentiate between user A and user B
- Extra solutions are required for e-services
  - Connection state sent in URL
  - Connection state sent in hidden fields
  - Session
- HTTP session
  - Set of information about a connection, differentiated by session ID
  - Server side – session file or database entry
  - Client side – *cookie*

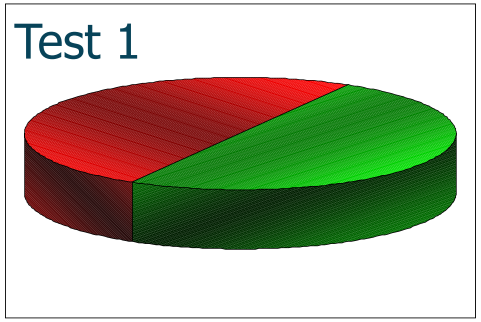
Threats

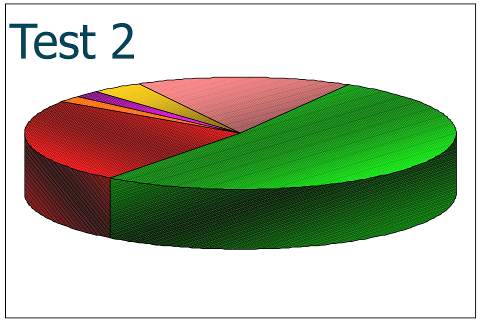
- Stealing identities of other users
  - Impersonating the victim
  - Stealing data or money, gaining private information (blackmail opportunities), making damages on behalf of the victim
  - If the administrator identity is stolen, arbitrary activity in the portal may be performed
- Information disclosure
  - Improperly configured error handling mechanism
  - Different OSs/Web servers may produce specific errors

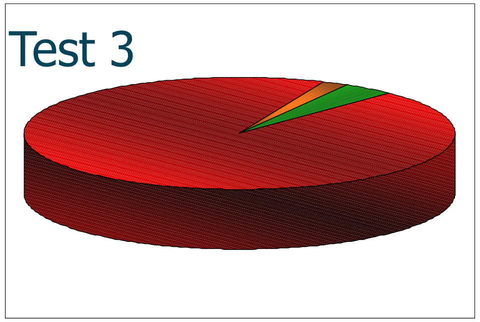
Is it real?

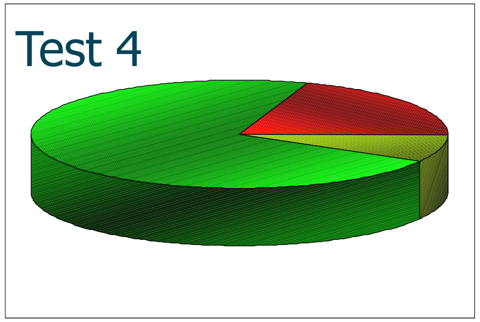
- In 2008 PSNC Security Team investigated 50 Polish e-commerce websites
  - Test 1 – unexpected characters in cookie name
  - Test 2 – forcing errors during writing cookie file
  - Test 3 – cookie expiration time
  - Test 4 – session expiration time
  - Test 5 – possibility of enforcing session ID
  - Test 6 – associating session ID with IP address
  - Test 7 – using *httponly* attribute
- More (in Polish): http://security.psnc.pl/reports/sklepy_internetowe_cookies.pdf

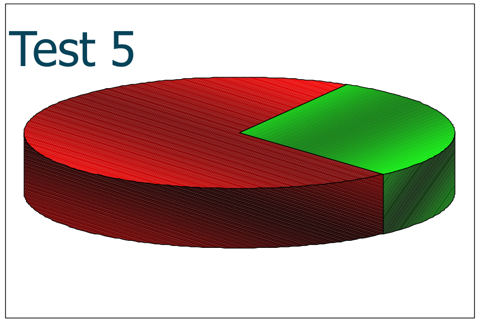
The results

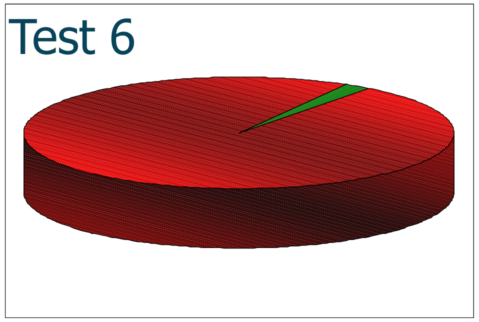
Test 1


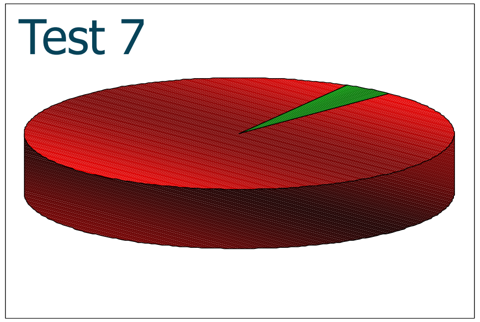
Test 2


Test 3


Test 4


Test 5


Test 6


Test 7


Red and alike = dangerous

Green and alike = secure

The most recent research (10 selected e-commerce sites)

- Credits to Jakub Tomaszewski (csc@bluerose.pl)
  - Only 1 column associated with sessions, but still shows problems

	Logging via SSL	Sending sensitive data via SSL	Password	Confirmation email	Association Session / IP address	Secured from XSS
Website 1	0	0	1	1	0	0
Website 2	0	0	1	0	0	0
Website 3	1	1	1	0	0	1
Website 4	1	1	0	1	0	1
Website 5	0	0	1	1	1	0
Website 6	1	1	1	0	0	0
Website 7	0	0	1	0	0	0
Website 8	1	0	1	0	1	1
Website 9	0	0	1	0.5	0	0
Website 10	1	1	1	0.5	0	0

Cookie in PHP

- Simple definition

```
<?
setcookie(“counter”, “1”, time()+3600,
“/my_dir”, “www.example.com”, true, true);
?>
```

- Significant properties
  - Name and value
  - Expiration time
  - Path (cookie visibility on the server)
  - Domain (where the browser may send the cookie to)
  - Additional attributes

Proper session configuration and implementation

- Session security may be assured both by proper configuration and implementation
  - Even if PHP is configured badly, a developer may implement secure session

    ```
    ini_set(‘session.[parameter]’, ‘[value]’);
    ```
- Significant topics to consider
  - Session ID threats
    - *Session Hijacking*
    - *Session Fixation*
  - Session and cookie expiration times
  - Applying security attributes
  - Session handling at the server side

Session ID threats (1)

- Session Hijacking
  - A legitimate user establishes a session with the given ID
  - An attacker steals the cookie and establishes an own session with this ID
- Countermeasures
  - Use cookies for exchanging session IDs
  - Beware of XSS
    - *Sanitize input data (in general)*
    - *Use security attributes (for cookies – will be shown later)*
  - Consider custom cookies handling on the server side

Session ID threats (2)

- Session Fixation
  - An attacker establishes a session with the given ID
  - An attacker forces the victim to use this ID
- Countermeasures
  - Regenerate session ID as often as possible
    - *session_regenerate_id()*
    - *Should be used at least just after the privileges have been changed (e.g. after authentication)*
  - Avoid session adoption
    - *The applications should never accept session ID that was enforced by the client*

Expiration times

- Session and cookie expiration time
  - The shorter, the more secure
  - However, the user must feel comfortable
- Examples / suggestions
  - e-banking services: 10-20 minutes
  - SMS gateway: an hour
  - OWASP standard: 5 minutes!
- Avoid persistent cookies whenever possible
- The user should always be able to log out

Additional attributes

- Domain and path
  - Should be consistent with minimum privileges principle
- *secure* attribute
  - Requires HTTPS in place to send cookie
- *httponly* attribute
  - Disabless access to cookies from client side scripting languages
  - Good support for protection against XSS attacks
  - Not all browsers support the attribute (especially older versions)

Association with the IP address

- Sending a cookie from other IP address than the one stored by the server causes invalidating the session
- Improves security, but significantly limits functionality
  - The clients may use proxies
  - Mobile clients
- Must be carefully considered during the design stage
- Very rarely applied
- Consider as a part of expert system
  - Do not invalidate the session but take some extra care

Storing session data at the server side

- By default PHP stores session data in files under /tmp path
  - Not secure especially in shared hosting scenario
- Minimum: set a save path
  - In the PHP configuration file, or
  - Using ini_set() function:

```
ini_set('session.save_path', '/safe/dir');
```

- Consider using custom database mechanism
  - Additional protection layer
  - Enables e.g. enhanced logging
  - Based on session_set_save_handler() function

Session management – summary

- Sessions are extremely useful, because we may differentiate users
- This introduces danger of stealing identities
- PHP (and alike) offer easy session management by default
  - However its security level should be increased
- Many settings may be defined both by administrators and developers
- There are many security/functionality tradeoffs
  - Additional measures are often a subject to consider during the design stage
  - Significant to have sufficient awareness on them

Other issues (short but relevant)

Test and production environment

- Usual mistakes (especially in non-purely corporate environments)
  - No dedicated test environment
  - Not used or pre-release code/functionality left
  - Revealing diagnostic functionality
- Countermeasures
  - Separate test and production environments
  - Test environment should work on same data (may be statistically obfuscated)
  - Mechanism for assuring consistency between the two environments necessary
  - Careful configuration of the release version

Leaving test functionality

- In the binary code
  - More functionality = more code = more bugs
  - Does not fit to minimum privileges principle
  - Test functionality may contain more bugs or backdoors
- In the source code
  - An attacker may try to draw conclusions on the process of development
  - Sometimes sensitive data are disclosed in such comments
    - *e.g. “Mike will repair that later” – so there is something worth taking a look, let’s analyze that!*

Configuration issues

- The configuration facilities should assure that it is possible to prepare a secure configuration
  - Problem for you: you do not design configuration facilities, but just implement the project
- Moreover, the behavior of the application should:
  - Warn the user against setting insecure options and explain the threads
  - Intuitively lead the user towards a more secure configuration
  - Assure the user about the security level of the current configuration

The code is good, but just a bit inefficient

- Not optimal code may be also dangerous
  - An attacker might search for data, for which the application responds very slowly
  - E.g. loops with counters dependent on the user data may be dangerous

```
int year = argv[1];
for (int i=1996; i<year; i++)
  calculate_very_detailed_stats_for_year(i);
```

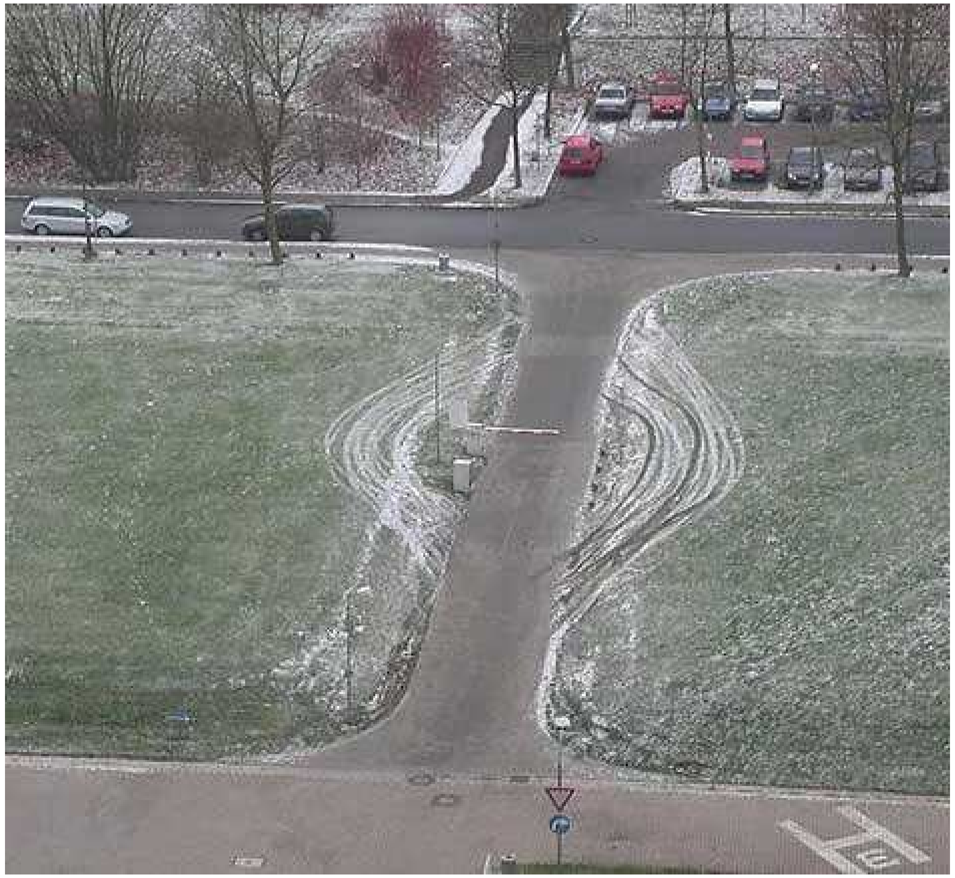
  - What will happen if the attacker may pass 3000 as year?
  - DoS/DDoS attacks on the application and/or server
- So it is always worth saving your resources

Whether to defend on the function/module level

- Sometimes we write an internal function, accepting some data that already “should have been” filtered
  - Our internal function may be copied to another module that does not assure sufficient sanitization
  - The function may work on another OS/hardware under different conditions
    - *It was one of the causes of the Therac 25 incident*
  - Someone might reuse only a part of the function, not aware about sanitizing issues
- Indeed, it makes no sense that every single function has its own sanitization mechanism – but at least remember to consider the problem
  - NULL pointers may cause the most trouble

And remember…

- It is often easier to bypass security mechanisms than to defeat it…



Source:
https://www.securecoding.org