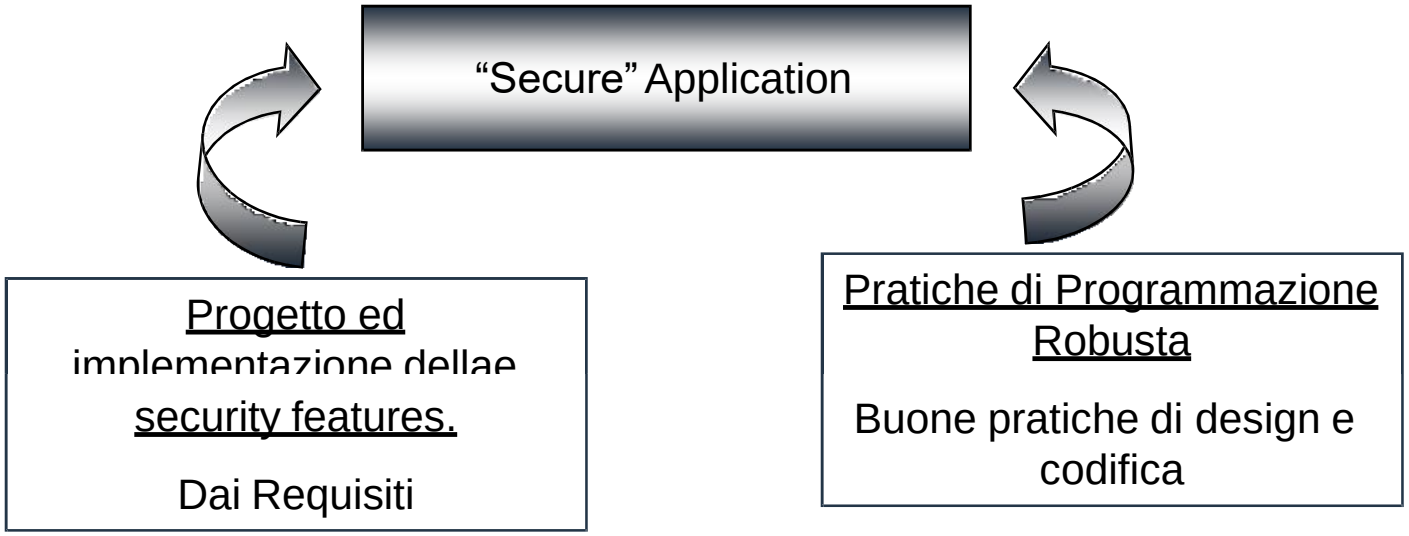
SECURE CODING

- Proprietà generali
- Common Weakness Enumeration: i 25 errori più comuni

Il Secure Coding

- Secure code deve avere le seguenti proprietà
  - **S**eamless (senza cuciture)
  - **E** asy to Understand
  - **C**ognizant of attacks
  - **U**nobtrusive
  - **R**esilient (elastico)
  - **E**rror Tolerant

Costruzione di una Applicazione sicura



Quando iniziare?

- Security dovrebbe essere vista come parte del sistema sin dall'inizio e non aggiunta alla fine

  Più tardi arriva

  - più il codice è insicuro
    (tricky patches instead of neat solutions)
  - può limitare le funzionalità
  - E costera molto di più

- Non si può aggiungere la sicurezza nella "versione 2.0"

Requisiti

Risultati di modello della minacce and valutazione del rischio:

- *Quali dati, qquali risorse protegger*
- *Contro cosa*
- *E da chi*

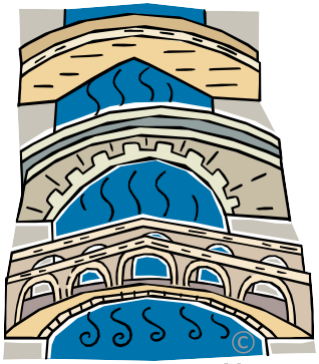
dovrebbe esser parte dei requisiti di sistema.

Minacce al Codice

- Noi, gli svluppatori
- Cattivi Inputs (e Outputs)
- API Abuso delle API
- Ambiente e Configurazione
- Tempo e Stato

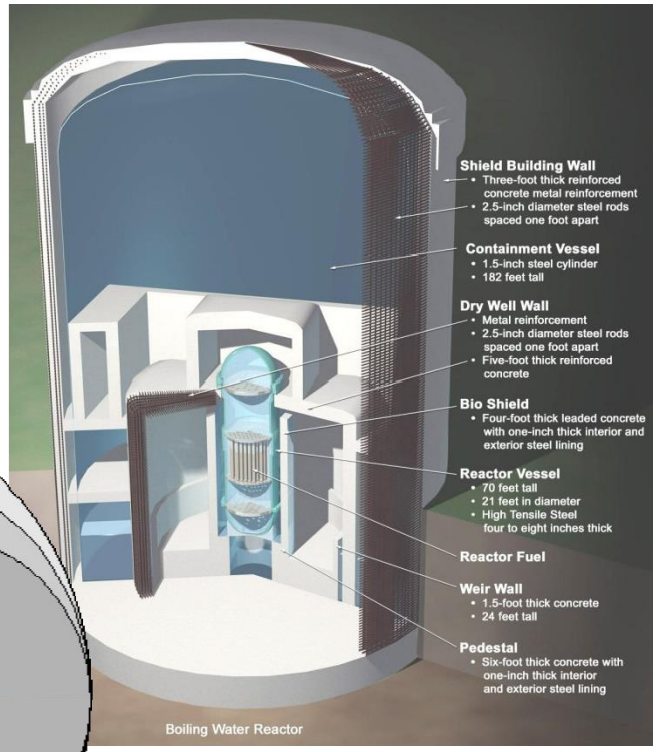
Architettura

- Modularità: dividere i programmi in parti semi-independenti
  - small, well-defined interfaces to each module/function
- Isolamento: ogni parte dovrebbe lavorare correttamente anche se le altre falliscono (ritornano risultati errati, inviano richieste con argomento errati)
- Difesa in profondità: costruire livelli di difesa multipli
- Semplicità (complesso => insicuro)
- Definire e rispettare catena di fiducia(trust)
- Pensare al sistema nella sua globalità

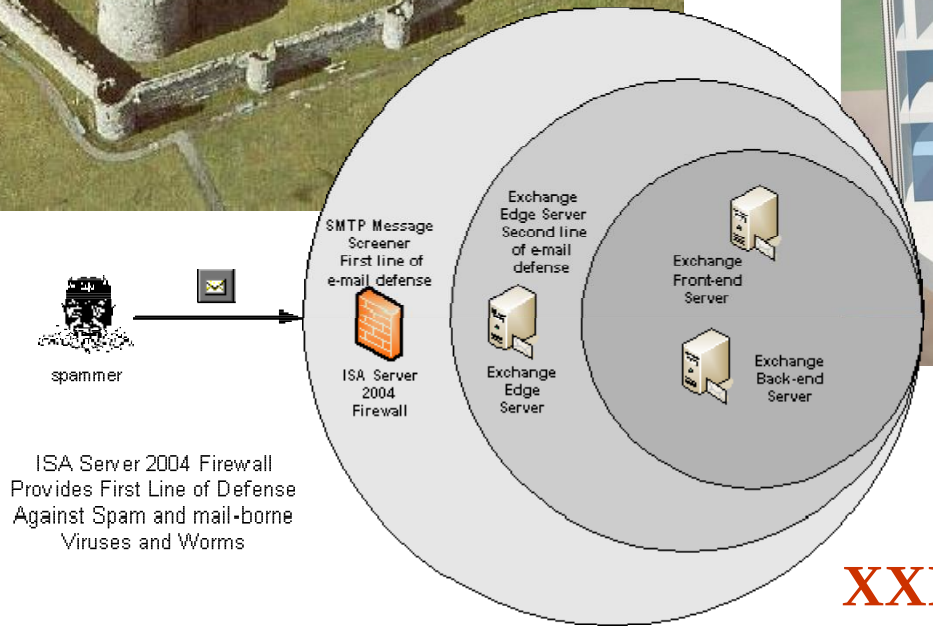


Livelli di difesa multipli

XIII century



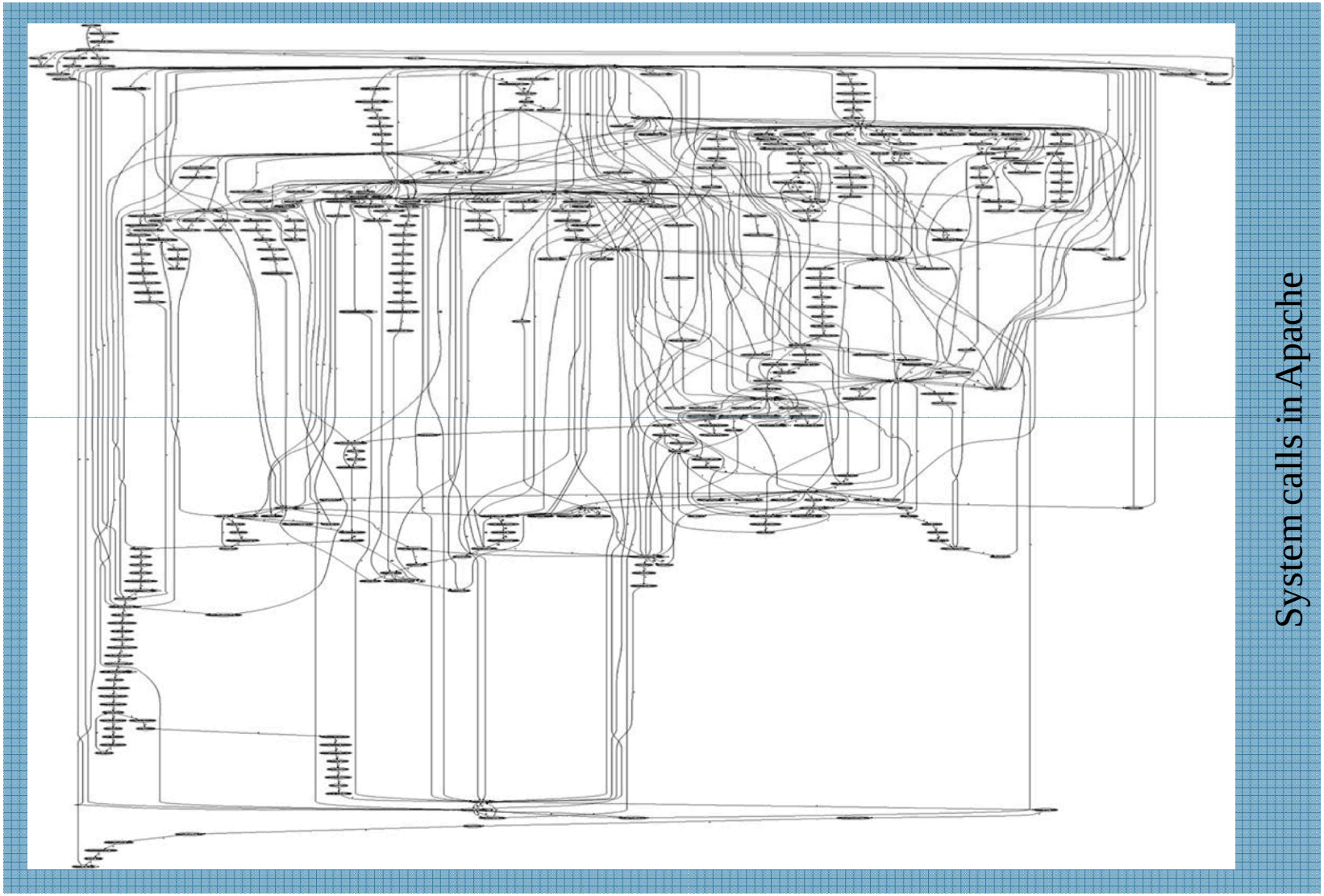
XX century



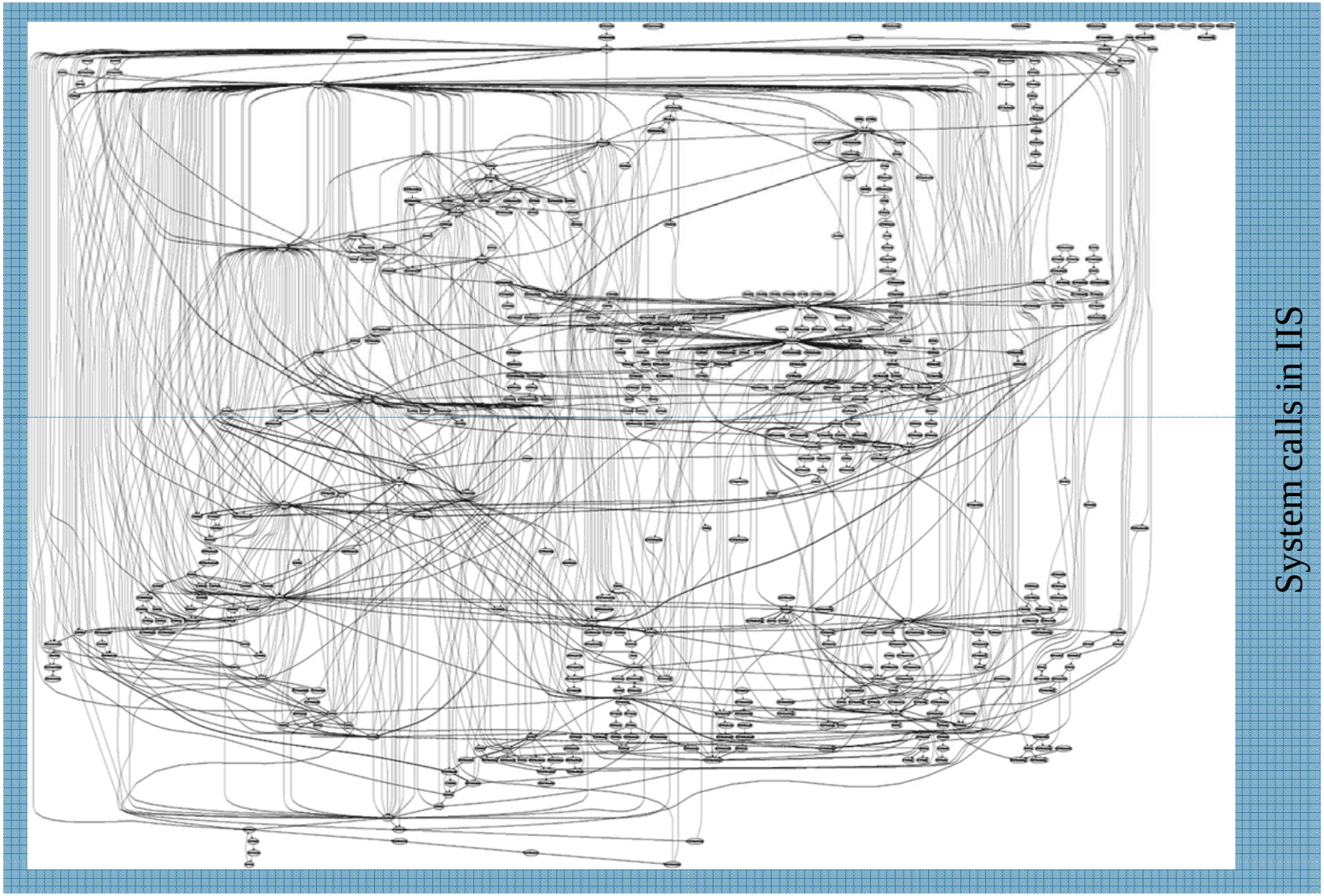
ISA Server 2004 Firewall Provides First Line of Defense Against Spam and mail-borne Viruses and Worms

XXI century

Complessità



Complessità



Secure Coding – Principi Generali

- Validare inputs e outputs
  il principio FLTR – Format, Length, Type, Range
- Ridurre la superficie di attacco, Rendere pù piccole possibile le parti del codice security-sensitive
  Running Code
  Entry Points (UI, ports, files, database, API calls)
  Ridurre i Privilegi
- Limitare il consumo di risorse
- Applicare la difesa in profondità
  Usare il paradigma del gatekeeper
- Usare le APIs correttamente
  Funzioni stringa in C, Java, .NET
- Rispettare le prescrizioni dei fornitori

Secure Coding – Principi Generali

- Principio del Privilegio Minimo
  - programmi dovrebbero essere eseguiti con i minimi privilegi possibile
  - Idem per accesso a databases, file etc.
  - revocare un privilegio quando non è più necessario anymore
  - Open files/registry with required access rights
    Don't write data in protected portions of the Operating System
- Scegliere valori default sicuri
- Negare per default
- Rilevare gli attacchi, fallire bene ed in sicurezza

Implementazione

- Bugs compaiono nel codice, perchè *errare è umano*
- Alcuni bugs possono diventare vulnerabilità
- Gli Attaccanti potrebbero scoprire un exploit per una vulnerabilità

Cosa fare?

- Leggere e seguire le guide linee per il linguaggio di programmazione ed Il tipo di software
- Considerare implicazioni per la sicurezza
- Riusare codice fidato (librerie, moduli etc.)
- Scrivere codice di buona qualità, leggibile e manutenibile (codice cattivo: mai sicuro)

Implementazione

Cosa fa questo codice?

```
@P=split//,".URRUU\c8R";@d=split//,"\n
rekcah xinU / lreP rehtona tsuJ";sub
p{@p{"r$p","u$p"}=(P,P);pipe"r$p","u$p
";++$p;($q*=2)+=$f=!fork;map{$P=$P[$f|
ord($p{$_})&6];$p{$_}=/^$P/ix?$P:close
$_}keys%p}p;p;p;p;p;map{$p{$_}=~/^[P.]
/&& close$_}%p;wait until$?; map{
/^r/&&<$_>}%p;$_=$d[$q];sleep rand(2)
if/\S/;print
```

SECURE CODING

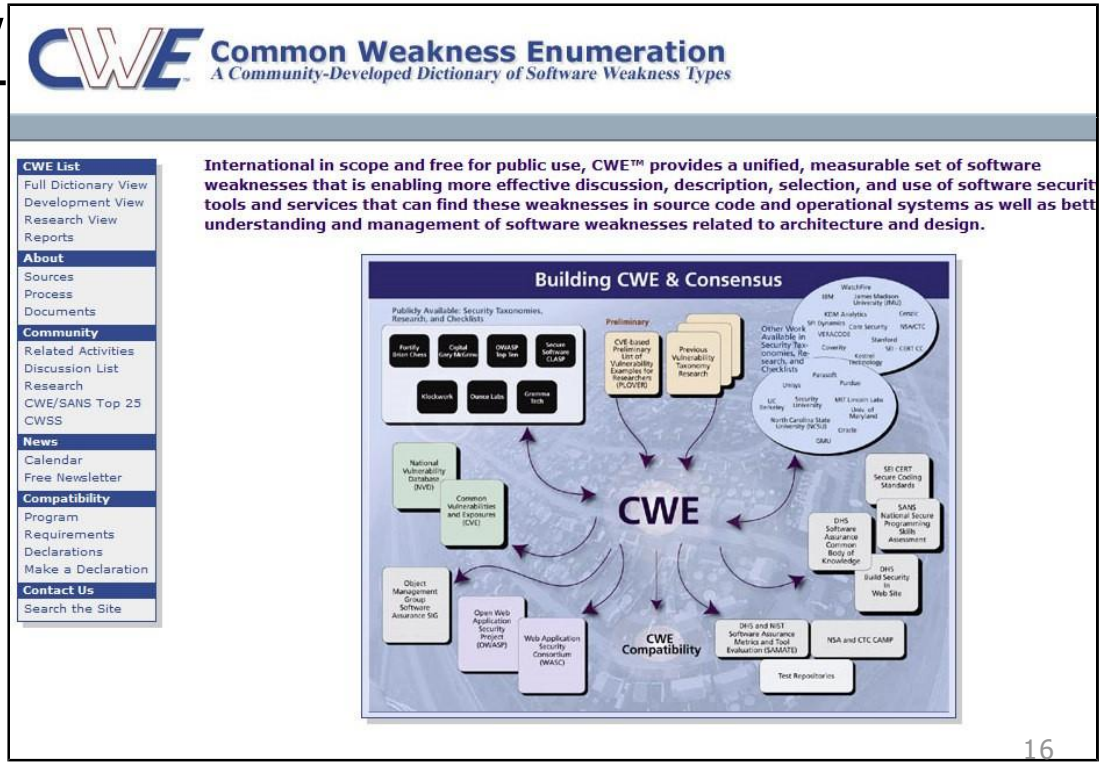
- Common Weakness Enumeration:
  i 25 errori software più comuni

I 25 errori software più comuni

- Prima Lista Pubblicata Marzo 2009 dalla
- *CWE Common Weakness Enumeration*

http://cwe.mitre.org/

- SANS e MITRE



- Interazione Insicura tra Componenti (9 casi)
- Gestione Rischiosa delle Risorse (9 casi)
- Falle nella Difesa (7 casi)

Interazione Insicura tra Componenti

Le debolezze di questa categoria sono dovute a modi non sicuri in cui vengono scambiati i dati tra componenti, moduli, programmi, processi, thread, o sistemi

- CWE-20: Validazione Impropria dell'Input
- CWE-116: Codifica impropria e "escaping" di Output
- CWE-89: Mancata protezione Struttura di Query SQL (simile 'SQL Injection')
- CWE-79: Mancata protezione Struttura di pagina Web (simile 'Cross-site Scripting')
- CWE-78: Mancata protezione Struttura di comando OS (simile 'OS Command Injection')
- CWE-319: Trasmissione di Informazioni Sensibili in Chiaro
- CWE-352: Cross-Site Request Forgery (CSRF)
- CWE-362: Race Condition
- CWE-209: Fuga di informazioni tramite Error Message

Gestione Rischiosa delle Risorse

Le debolezze in questa categoria sono correlate ai modi in cui il software non gestisce appropriatamente la creazione, uso, trasferimento o distruzione di importanti risorse del sistema

- CWE-119: Mancato vincolo delle operazioni nei Limiti del Memory Buffer
- CWE-642: Controllo esterno di Dati di Stato Critici
- CWE-73: Controllo esterno di Nomi di File o Path
- CWE-426: Path di ricerca non fidati
- CWE-94: Mancato controllo nella generazione del Codice (simile 'Code Injection')
- CWE-494: Download di codice senza Integrity Check
- CWE-404: Improprio rilascio o shutdown di risorse
- CWE-665: Inizializzazione impropria
- CWE-682: Calcolo Errato

Falle nella Difesa/Porous Defenses

Le debolezze in questa classe sono dovute a tecniche di difesa usate scorrettamente, impropriamente o ignorate

- CWE-285: Controllo di Accesso Improprio (Autorizzazione)
- CWE-327: Uso di un algoritmo crittografico errato o rischioso
- CWE-259: Hard-Coded Password
- CWE-732: Assegnamenti di permessi insicuri a risorse critiche
- CWE-330: Uso di valori non sufficientemente casuali/random
- CWE-250: Esecuzione con privilegi non necessari
- CWE-602: Rispetto Client-Side della sicurezza Server-Side Security

Interazione Insicura tra Componenti

- CWE-20: Validazione Impropria dell'Input

Summary			
Weakness Prevalence	High	Consequences	Code execution Denial of service Data loss
Remediation Cost	Low	Ease of Detection	Easy to Difficult
Attack Frequency	Often	Attacker Awareness	High

- When software fails to validate input properly, an attacker is able to craft the input in a form that is not expected by the rest of the application. This will lead to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution.

Interazione Insicura tra Componenti

- CWE-20: Validazione Impropria dell'Input
- When software fails to validate input properly, an attacker is able to craft the input in a form that is not expected by the rest of the application. This will lead to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution.

Esempio Java

```
...
public static final double price = 20.00;
int quantity = currentUser.getAttribute("quantity");
double total = price * quantity;
chargeUser(total);
...
```

Se l'utente è libero di specificare numeri negativi il suo conto viene incrementato anziché decrementato

Interazione Insicura tra Componenti

- CWE-116: Codifica impropria e "escaping" di Output

Summary			
Weakness Prevalence	High	Consequences	Code execution Data loss
Remediation Cost	Low	Ease of Detection	Easy to Moderate
Attack Frequency	Often	Attacker Awareness	High

Discussion

- I programmi fanno "quello che gli diciamo" e non "quello che vogliamo"

Quando un programma genera dati in output ad altri componenti nella forma di messaggi strutturati come richieste o query è necessario *separare le informazioni di controllo ed i metadati dai dati* veri e propri.*In molti paradigmi comndi e dati viaggiano assieme es.con caratteri speciali*

Interazione Insicura tra Componenti

- CWE-116: Codifica impropria e "escaping" di Output

Summary			
Weakness Prevalence	High	Consequences	Code execution Data loss
Remediation Cost	Low	Ease of Detection	Easy to Moderate
Attack Frequency	Often	Attacker Awareness	High

Discussion

- *Il software prepara un messaggio strutturato per comunicare con un'altro componente, ma la codifica o l'escaping dei dati è fatto scorrettamnete. Come risultato la struttura prevista del messaggio non è rispettata.*
- I programmi fanno "quello che gli diciamo" e non "quello che vogliamo".Quando un programma genera dati in output ad altri componenti nella forma di messaggi strutturati come richieste o query è necessario *separare le informazioni di controllo ed i metadati dai dati* veri e propri.*In molti paradigmi comndi e dati viaggiano assieme es.con caratteri speciali*

Interazione Insicura tra Componenti

- CWE-116: Codifica impropria e "escaping" di Output
- *Il software prepara un messaggio strutturato per comunicare con un'altro componente, ma la codifica o l'escaping dei dati è fatto scorrettamemte. Come risultato la struttura prevista del messaggio non è rispettata.*
- Output Sanitization, Output Encoding, Output Validation

This example takes user input, passes it through an encoding scheme and then creates a directory specified by the user.

Perl Example:

```
sub GetUntrustedInput {
  return($ARGV[0]);
}

sub encode {
  my($str) = @_;
  $str =~ s/\&/\&amp;/gs;
  $str =~ s/\"/\&quot;/gs;
  $str =~ s/\'/\&apos;/gs;
  $str =~ s/\</\&lt;/gs;
  $str =~ s/\>/\&gt;/gs;
  return($str);
}

sub doit {
  my $uname = encode(GetUntrustedInput("username"));
  print "<b>Welcome, $uname!</b><p>\n";
  system("cd /home/$uname; /bin/ls -l");
  }
```

The programmer attempts to encode dangerous characters, however the blacklist for encoding is incomplete (CWE-184) and an attacker can still pass a semicolon, resulting in a chain with command injection (CWE-77).

Interazione Insicura tra Componenti
CWE-116: Codifica impropria e "escaping" di Output

This example takes user input, passes it through an encoding scheme and then creates a directory specified by the user.

Perl Example:

```
sub GetUntrustedInput {
  return($ARGV[0]);
}

sub encode {
  my($str) = @_;
  $str =~ s/\&/\&amp;/gs;
  $str =~ s/\"/\&quot;/gs;
  $str =~ s/\'/\&apos;/gs;
  $str =~ s/\</\&lt;/gs;
  $str =~ s/\>/\&gt;/gs;
  return($str);
}

sub doit {
  my $uname = encode(GetUntrustedInput("username"));
  print "<b>Welcome, $uname!</b><p>\n";
  system("cd /home/$uname; /bin/ls -l");
  }
```

The programmer attempts to encode dangerous characters, however the blacklist for encoding is incomplete (CWE-184) and an attacker can still pass a semicolon, resulting in a chain with command injection (CWE-77).

- L'attaccante può inserire 'pwd cosa che provoca l'aggiunta di ";"

Interazione Insicura tra Componenti

- CWE-89: Mancata protezione Struttura di Query SQL (simile 'SQL Injection')
- The application dynamically generates an SQL query based on user input, but it does not sufficiently prevent that input from modifying the intended structure of the query.

- Le “magic quotes”

Interazione Insicura tra Componenti

- CWE-89: Mancata protezione Struttura di Query SQL (simile 'SQL Injection')

```
...
string userName = ctx.getAuthenticatedUserName();
string query = "SELECT * FROM items WHERE owner = '" + userName + "' AND itemname = '" + ItemName.Text + "'";
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
...
```

name' OR 'a'='a

```
SELECT * FROM items WHERE owner = <userName> AND itemname = <itemName>;
```

Injection 1, si assuma che l'utente digiti rossi come Username e: **`name' OR 'a'='a`**

```
SELECT * FROM items WHERE owner = 'rossi' AND itemname = 'name' OR 'a'='a';
```

Injection 2 sia username che item name: **`' OR ''='`** equivale logicamente a

```
SELECT * FROM items
```

Interazione Insicura tra Componenti

- CWE-79: Mancata protezione Struttura di pagina Web (simile 'Cross-site Scripting')

Summary			
Weakness Prevalence	High	Consequences	Data loss Security bypass
Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

1. Untrusted data enters a web application, typically from a web request.
2. The web application dynamically generates a web page that contains this untrusted data.
3.During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc.
4.A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data.
5.Since the script comes from a web page that was sent by the web server, the web browser executes the malicious script in the context of the web server's domain.
6.This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.

Interazione Insicura tra Componenti

CWE-79: Mancata protezione Struttura di pagina Web (simile 'Cross-site Scripting')

- Type 1: Reflected XSS (or Non-Persistent) *Il server legge I dati direttamente dalla HTTP request e li riflette nella HTTP response. Es. In chat.* Il codice dell'attaccante è eseguito sul browser dell'utente accedendo a cookies etc.
- Type 2: Stored XSS (or Persistent) L'applicazione memorizza codice pericoloso in un *database, message forum, visitor lo* , o in un altro data store fidato. I dati pericolosi vengono successivamente riletti per generare contenuti dinamici. Se gli utenti che eseguono il doice sono utenti con privilegi elevati: esecuzione di azioni privilegiate.
- Type 0: DOM-Based XSS. Nel DOM-based XSS, il client o il server effettua l'injection del XSS nella pagina. Spesso usato Javascript che effettua sanity checks sulle form dei dati inseriti dall'utente (dynamic HTML).

Interazione Insicura tra Componenti

CWE-78: Mancata protezione Struttura di comando OS (simile 'OS Command Iniection')

Summary			
Weakness Prevalence	High	Consequences	Data loss Security bypass
Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

SHELL INJECTION o INJECTION di Metacaratteri

Il software usa un input esterno per costruire dinamicamente tutto o parte di un comando, che è poi passato per l'esecuzione al sistema operativo, che non controlla e forza sufficientemente quali comandi e parametri vengono specificati.

Viola anche la non repudiation

Interazione Insicura tra Componenti

CWE-319: Trasmissione di Informazioni Sensibili in Chiaro

Summary			
Weakness Prevalence	Medium	Consequences	Data loss
Remediation Cost	Medium	Ease of Detection	Easy
Attack Frequency	Sometimes	Attacker Awareness	High

Il software trasmette *in chiaro* informazioni sensibili o dati critici per la
in un canale di comunicazione che può essere *sniffato* da attori autorizzati.

- Criptare i dati con schema di crittografia affidabile prima di trasmetterli
- Quando si usano applicazioni Web con SSL, usare SSL l'intera sessione da *login* a *logout*, e non solo per la pagina iniziale di login

Interazione Insicura tra Componenti

CWE-352: Cross-Site Request Forgery (CSRF)

Summary			
Weakness Prevalence	High	Consequences	Data loss Code execution
Remediation Cost	High	Ease of Detection	Moderate
Attack Frequency	Often	Attacker Awareness	Medium

Quando un web server è progettato per ricevere richieste da un client senza meccanismi per verificare se la richiesta è stata spedita intenzionalmente, allora potrebbe essere possibile per un attaccante indurre il client a fare una richiesta non voluta al web server che sarò trattata come richiesta autentica. Ciò può essere realizzato attraverso URL, caricamento di immagini, XMLHttpRequest, etc. e può tradursi in data disclosure, esecuzione di codice indesiderato.

Es. Add user accounts via a URL in an img tag

Delete a victim's information via a URL or an img tag

Perform actions as administrator via a URL or an img tag

Interazione Insicura tra Componenti

- CWE-362: Race Condition

Summary			
Weakness Prevalence	High	Consequences	Data loss Code execution
Remediation Cost	High	Ease of Detection	Moderate
Attack Frequency	Often	Attacker Awareness	Medium

Il codice richiede che un certo stato non dovrebbe essere modificato tra due operazioni, ma esiste una finestra temporale in cui lo stato può essere modificato da un attore o un processo inatteso. Mancato controllo della sincronizzazione nell'accesso a risorse condivise.

Utilizzare se disponibili primitive di sincronizzazione, che incapsulano solo il codice critico.

Interazione Insicura tra Componenti

- CWE-362: Race Condition

```
$transfer_amount = GetTransferAmount();
$balance = GetBalanceFromDatabase();

if ($transfer_amount < 0) {
FatalError("Bad Transfer Amount");
}
$newbalance = $balance -
$transfer_amount; if (($balance -
$transfer_amount) < 0) {
FatalError("Insufficient Funds");
SendNewBalanceToDatabase($newbalance);
NotifyUser("Transfer of $transfer_amount
succeeded."); NotifyUser("New balance: $newbalance");
```

Si suppone all'inizio 100 euro, una serie di richieste di prelievo di 1 euro
Aggiornamento non sincronizzato.

Interazione Insicura tra Componenti

- CWE-209: Fuga di informazioni tramite Error Message

Summary			
Weakness Prevalence	High	Consequences	Data loss
Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

Il software genera messaggi di errori che rivelano informazioni sensibili su utenti, ambiente di esecuzione o dati associati. (es. Pathname path traversal weakness, query injection malformate etc.) (comune in PHP)

Esempio imprudente in Java

```
try {
/.../
}
catch (Exception e) {
System.out.println(e);
}
```

Interazione Insicura tra Componenti

- CWE-209: Fuga di informazioni tramite Error Message

Esempio PERL

```
$ConfigDir = "/home/myprog/config";
$uname = GetUserInput("username");
# avoid CWE-22, CWE-78, others.
ExitError("Bad hacker!") if ($uname !~ /^\w+$/);
$file = "$ConfigDir/$uname.txt";
if (! (-e $file)) {
ExitError("Error: $file does not exist");
}
...
```

Messaggi di errore disabilitati/ridotti

Messaggi di errori: informazioni minime per l'audience utenti vs. programmatori (informazioni di debugging diversa da release di produzione)

Non passare le SQL exception agli utenti senza filtraggio

Creare pagine di errore di default, disabilitare modi "verbose" (es.

Gestione Rischiosa delle Risorse

Le debolezze in questa categoria sono correlate ai modi in cui il software non gestisce appropriatamente la creazione, uso, trasferimento o distruzione di importanti risorse del sistema

- CWE-119: Mancato vincolo delle operazioni nei Limiti del Memory Buffer
- CWE-642: Controllo esterno di Dati di Stato Critici
- CWE-73: Controllo esterno di Nomi di File o Path
- CWE-426: Path di ricerca non fidati
- CWE-94: Mancato controllo nella generazione del Codice (simile 'Code Injection')
- CWE-494: Download di codice senza Integrity Check
- CWE-404: Improprio rilascio o shutdown di risorse
- CWE-665: Inizializzazione impropria
- CWE-682: Calcolo Errato

Gestione Rischiosa delle Risorse

- CWE-119: Mancato vincolo delle operazioni nei Limiti del Memory Buffer

Summary			
Weakness Prevalence	High	Consequences	Code execution Denial of service Data loss
Remediation Cost	Low	Ease of Detection	Easy to Moderate
Attack Frequency	Often	Attacker Awareness	High

Il software consente operazioni di read o write operations su memoria collocata all'esterno dell'intervallo di allocazione. Un attaccante potrebbe accedere/modificare informazioni sensibili, causare crash di sistema, alterare il flusso atteso del controllo, o esegurie codice arbitrario.

Gestione Rischiosa delle Risorse

- CWE-119: Mancato vincolo delle operazioni nei Limiti del Memory Buffer

Esempio in C

```
int main (int argc, char **argv) {
char *items[] = {"boat", "car",
"truck", "train"};
int index = GetUntrustedOffset();
printf("You selected %s\n",
items[index-1]);
}
```

Usare linguaggi che controllano l'overflow in modo nativo (Java, Pearl) e non disabilitano(come ad Es.C#), librerie (es. Safe C String Library, Strsafe.h di Microsoft), o framework

Altri Esempi:

CWE-120 Classic Buffer Overflow CWE-129 Unchecked Array Indexing CWE-130 Failure to Handle Length Parameter Inconsistency CWE-131 Incorrect Calculation of Buffer Size CWE-415 Double Free CWE-416 Use After Free

Gestione Rischiosa delle Risorse

- CWE-642: Controllo esterno di Dati di Stato Critici

- Il software memorizza informazioni di stato critiche per la sicurezza o lo stesso software in locazioni accessibili ad attori non autorizzati
- Se l'attaccante può modificare tali informazioni il software potrebbe accedere a risorse non previste.
- Le informazioni di stato possono essere memorizzate in locazioni quali *cookie*, campi hidden di un web form, parametri o argomenti di input, variabili di ambiente, record di database, file di settings etc. Sono possibili vulnerabilità.

  Ad esempio, se una applicazione effettua autenticazione e salva lo stato in un cookie "authenticated=true" Un attaccante potrebbe semplicemente creare questo cookie per bypassare l'autenticazione.

Gestione Rischiosa delle Risorse

CWE-642: Controllo esterno di Dati di Stato Critici

- Analizzare attentamente le locazioni "pericolose" per I dati critici
- Usare un framework HTTP per mantenere lo stato
- Mantenere informazioni di stato sul client tramite encryption and integrity checking, con un meccanismo di controllo server side usare un MAC algorithm tipo Hash Message Authentication Code (HMAC)

Observed Examples

Reference	Description
CVE-1999-0073	Telnet daemon allows remote clients to specify critical environment variables for the server, leading to code execution.
CVE-2000-0102	Shopping cart allows price modification via hidden form field.
CVE-2000-0253	Shopping cart allows price modification via hidden form field.
CVE-2005-2428	Mail client stores password hashes for unrelated accounts in a hidden form field.
CVE-2006-7191	Untrusted search path vulnerability through modified LD_LIBRARY_PATH environment variable.
CVE-2007-4432	Untrusted search path vulnerability through modified LD_LIBRARY_PATH environment variable.
CVE-2008-0306	Privileged program trusts user-specified environment variable to modify critical configuration settings.
CVE-2008-1319	Server allows client to specify the search path, which can be modified to point to a program that the client has uploaded.
CVE-2008-4752	Application allows admin privileges by setting a cookie value to "admin."
CVE-2008-5065	Application allows admin privileges by setting a cookie value to "admin."
CVE-2008-5125	Application allows admin privileges by setting a cookie value to "admin."
CVE-2008-5642	Setting of a language preference in a cookie enables path traversal attack.
CVE-2008-5738	Calendar application allows bypass of authentication by setting a certain cookie value to 1.

Gestione Rischiosa delle Risorse

- CWE-73: Controllo esterno di Nomi di File o Path
- Il software consente allo user imput di controllare o influenzare dei path che sono usati nel filesystem. L'attaccante potrebbe modificare I path di sistema, o ad esempio forzare l'uso di una sua configurazione.

Example 1:

The following code uses input from an HTTP request to create a file name. The programmer has not considered the possibility that an attacker could provide a file name such as "../../tomcat /conf/server.xml", which causes the application to delete one of its own configuration files (CWE-22).

```
String rName = request.getParameter("reportName");
File rFile = new File("/usr/local/apfr/reports/" + rName);
...
rFile.delete();
```

- Quando il nome dei filename è limitato o conosciuto, creare un mapping da un insieme fissato di valori di input es.1,2, etc. ai nomi di file reali rifiutando tutti gli altri input.

Gestione Rischiosa delle Risorse

- CWE-426: Path di ricerca non fidati
- *La applicazione cerca risorse critiche usando un path fornito esternamente che punta a risorse che non sono sotto il controllo diretto dell'applicazione.* L'attaccante potrebbe eseguire i propri programmi, accedere dati o modificare configurazioni. Se l'applicazione utilizza un path locale per accedere a risorse critiche, l'attaccante potrebbe modificare tale path di ricerca, per puntare ad un programma malicious. Il problema si estende ad ogni tipo di risorsa critica fidata.

- Esecuzione con privilegi non dovuti
- Ridirezione del programma su file errati

Gestione Rischiosa delle Risorse

- CWE-426: Path di ricerca non fidati
- *Es.programma eseguito con privilegi setuid per accedere al una directory "restricted" il programma è in /bin*. L'attaccante non ha accesso a /bin

C Example:

```
#define DIR "/restricted/directory"

char cmd[500];
sprintf(cmd, "ls -l %480s", DIR);
/* Raise privileges to those needed for accessing DIR. */
RaisePrivileges(...);
system(cmd);
DropPrivileges(...);
...
```

- Apparentemente senza pericoli: il codice non ha input il e nome della directory da leggere è fisso. L'attaccante sembra poter osservere soltanto il contenuto di DIR. Il codice eseguito con privilegi è limitato ad un solo comando.

Gestione Rischiosa delle Risorse

- CWE-426: Path di ricerca non fidati
- *Es.programma eseguito con privilegi setuid per accedere ad una directory "restricted"*

C Example:

```
#define DIR "/restricted/directory"

char cmd[500];
sprintf(cmd, "ls -l %480s", DIR);
/* Raise privileges to those needed for accessing DIR. */
RaisePrivileges(...);
system(cmd);
DropPrivileges(...);
...
```

- Ma il codice non setta la variabile PATH, viene quindi usato il PATH dell'utente che potrebbe: 1)togliere /bin dal proprio PATH;2)inserire /my/bin nel PATH 3)inserire un proprio "ls" in /my/bin 4) eseguire il programma che chiamando system() 5) usa PATH 6) e dunque esegue "ls" modificato con privilegi raised

Gestione Rischiosa delle Risorse

- CWE-94: Mancato controllo nella generazione del Codice (simile 'Code Injection')

Esempio inserimento di codice PHP, in una "messaggeria"

PHP Example:

```
$MessageFile = "cwe-94/messages.out";

if ($_GET["action"] == "NewMessage") {
  $name = $_GET["name"];
  $message = $_GET["message"];
  $handle = fopen($MessageFile, "a+");
  fwrite($handle, "<b>$name</b> says '$message'<hr>\n");
  fclose($handle);
  echo "Message Saved!<p>\n";
}
else if ($_GET["action"] == "ViewMessages") {
  include($MessageFile);
}
```

Se un attaccante inserisce

```
name=h4x0r
message=%3C?php%20system(%22/bin/ls%20-l%22);?%3E
```

Il suo messaggio sarà deco

```
<?php system("/bin/ls -l");?>
```

Filtri: usare white list per accettare e black list per rilevare attacchi

Usare specifici parametri di controllo di ambiente Es. parametro"-T" nel Perl

Gestione Rischiosa delle Risorse

- CWE-494: Download di codice senza Integrity Check

Downloads di codice sorgente o codice eseguibile da una locazione remota senza aver verificato integrità ed origine del codice.

Consente ad un attaccante di eseguire malicious code compromettendo *host server*, *DNS spoofing*, o *modificando il codice in transito*

Effettuare forward e reverse DNS per rilevare il DNS spoofing. (non rileva modifiche in transito o nel server)

Crittare il codice con uno schema affidabile prima di trasmetterlo(non rileva lo sppofing o le modifiche nel server)

Usare firma digitale del codice (es.schema authenticode)

Esempio Java

```
URL[] classURLs= new URL[]{
new URL("file:subdir/")
};
URLClassLoader loader = new URLClassLoader(classURLs);
Class loadedClass = Class.forName("loadMe", true,
loader);
```

Gestione Rischiosa delle Risorse

- CWE-404 Improprio rilascio o shutdown di risorse
- Il programma non rilascia o rilascia scorrettamente una risorsa di sitema. Quando una risorsa viene creata o allocata lo sviluppatore è responsabile per un rilascio corretto e deve tern conto dei possibili percorsi di scadenza, o invalidità come un fissato periodo di tempo o di revoca
- Esempio: Metodo che non chiude il file che legge, ache se alla fine il Finalize() dello StreamReader chiamerà Close() non c'e' garanzia che il sistema potrebbe esaurire le handle a file

```
private void processFile(string fName) {
  StreamWriter sw = new
  StreamWriter(fName);
  string line;
  while ((line = sr.ReadLine()) != null)
    processLine(line);
}
```

Gestione Rischiosa delle Risorse

- CWE-404 Improprio rilascio o shutdown di risorse
- Esempio: Se un eccezione avviene dopo aver stabilito una connessione al database e prima di chiudere la connessione si può esaurire il pool di connessioni al database. Usare un "pattern" di connessione come il seguente:

```
try {
  Connection con = DriverManager.getConnection(some_connection_string)
}
catch ( Exception e ) {
  log( e )
}
finally {

  con.close()
}
```

- Controllare il rilascio di tutti i componenti di strutture complesse
- Controllare I punti di uscita
- Usare linguaggi con garbage collection automatici (Java, Ruby,Lisp)
- Non mescolare coppie di funzioni acquisizioni/rilascio come malloc/free, new/delete, and new[]/delete[].

Gestione Rischiosa delle Risorse

- CWE-665: Inizializzazione impropria

- Il software non segue le procedure appropriate per inizializzazione di una risorsa,la risorsa può trovarsi in uno stato improprio quando viene usata/acceduta
- Esempio il codice di inizializzazione non è mai eseguito

Java Example:

```
private boolean initialized = true;
public void someMethod() {
  if (!initialized) {
    // perform initialization tasks
    ...

    initialized = true;
  }
```

Gestione Rischiosa delle Risorse

- CWE-665: Inizializzazione impropria

Es. Esegue operazioni se è l'amministratore uid=0, un timeout nell'accesso al database può generare un assegnamento di "0" a $uid

Perl Example:

```
$username = GetCurrentUser();
$state = GetStateData($username);
if (defined($state)) {
  $uid = ExtractUserID($state);
}
# do stuff
if ($uid == 0) {
  DoAdminThings();
}
```

Es. Concatenazione di stringa non inizializzata, comportamento che dipende dallo stato della memoria se c'e' o non c'è il teminatore null in str[0] o dopo.

C Example:

```
char str[20];
strcat(str, "hello world");
printf("%s", str);
```

Gestione Rischiosa delle Risorse

- CWE-665: Inizializzazione impropria

Es. Esegue operazioni se è l'amministratore uid=0, un timeout nell'accesso al database può generare un assegnamento di "0" a $uid

Perl Example:

```
$username = GetCurrentUser();
$state = GetStateData($username);
if (defined($state)) {
  $uid = ExtractUserID($state);
}
# do stuff
if ($uid == 0) {
  DoAdminThings();
}
```

Linguaggi che controllano le inizializzazione. Es, Java per variabili locali

Attenzione ai condizionali

Attenzione a variabili inizializzate dall'esterno

Attivare i warning nei compilatori

Gestione Rischiosa delle Risorse

- CWE-682: Calcolo Errato

Summary			
Weakness Prevalence	High	Consequences	Denial of service Data loss Code execution
Remediation Cost	Low	Ease of Detection	Easy to Difficult
Attack Frequency	Often	Attacker Awareness	Medium

Il software effettua calcoli che generano risultati scorretti o indesiderati che sono sccessivamente utilizzati in decisioni security-critical o per la gestione di risorse.(possibilità di crash, privilegi non dovuti

Es. Second_char "dovrebbe" puntare al secondo byte di x

```
int *p = x;
char * second_char = (char *)(p + 1);
```

Ma è fuori di 3 byte in una piattaforma a 32-bit, se *p* è usato in scrittura (write su memoria non autorizzata) se in lettura fuga di info.

Gestione Rischiosa delle Risorse

- CWE-682: Calcolo Errato

Es. Il codice dovrebbe allocare il numero di item per una lista di inventario,il cui totale è letto da get_num_items se gli item sono maggiori del massimo int verrà allocata una lista più breve.

```
inv_item_t table_ptr; /*10kb struct containing item info */
int num_items;
...
num_items = get_num_items();
table_ptr = (inv_item_t*)malloc(sizeof(inv_item_t)*num_items);
...
```

- Comprendere la rappresentazione sottostante al linguaggio
- Usare I tipi di dati corretti es unsigned, validazione degli input nei range prescritti, Librerie di gestione interi es. SafeInt (C++) or IntegerLib (C or C++).

Falle nella Difesa/Porous Defenses

Le debolezze in questa classe sono dovute a tecniche di difesa usate scorrettamente, impropriamente o ignorate

- CWE-285: Controllo di Accesso Improprio (Autorizzazione)
- CWE-327: Uso di un algoritmo crittografico errato o rischioso
- CWE-259: Hard-Coded Password/Password "inchiodate"
- CWE-732: Assegnamenti di permessi insicuri a risorse critiche
- CWE-330: Uso di valori non sufficientemente casuali/random
- CWE-250: Esecuzione con privilegi non necessari
- CWE-602: Rispetto Client-Side della sicurezza Server-Side Security

Falle nella Difesa/Porous Defenses

- CWE-285: Controllo di Accesso Improprio (Autorizzazione)

Il Software none ffettua gli access control checks in modo consistente attraverso tutti i path di esecuzione.

- Usare le ACL
- Dividere i livelli di privilegio in almeno: anonymous, normale, privilegiato, e ammnistrativo

Falle nella Difesa/Porous Defenses

- CWE-327: Uso di un algoritmo crittografico errato o rischioso
- L'uso di un algoritmo non-standard è pericoloso, Esistono tecniche ben note e documentate per "crakkare" l'algoritmo.

- Usare librerie, ESAPI Encryption feature. Non reimplementare: errori di dettaglio

Falle nella Difesa/Porous Defenses

- CWE-259: Hard-Coded Password/Password "inchiodate"
- Il codice contiene password codificate dentro il codice:

In ingresso: per controllare una password fornita dall'esterno

In uscita: per utilizzare la password per accesso ad altre risorse

Causa In input: spesso backdoor per sviluppatori che resta nelle patch se pubblicata gravi problemi

Causa in output: sistemi front-end che autenticano verso un back-end es. db chiunque ha accesso al software può estrarla (rev.eng.) pericolo maggiore per client side sw

Falle nella Difesa/Porous Defenses

- CWE-259: Hard-Coded Password/Password "inchiodate"
- Codice disassemblato

```
...
DriverManager.getConnection(url, "scott", "tiger");
...
```

```
javap -c ConnMngr.class
  22: ldc #36; //String jdbc:mysql://ixne.com/rxsql
  24: ldc #38; //String scott
  26: ldc #17; //String tiger
```

- Backdoor(distribuito con il codice binario)

C/C++ Example:

```
int VerifyAdmin(char *password) {
  if (strcmp(password, "Mew!")) {
    printf("Incorrect Password!\n");
    return(0)
  }
  printf("Entering Diagnostic Mode...\n");
  return(1);
}
```

Falle nella Difesa/Porous Defenses

- CWE-259: Hard-Coded Password/Password "inchiodate"

Rimedio possibile in input: usare un login mode speciale, abilitare solo da console di sistema e non da network connection

Rimedio possibile in output: codificare le password in un file di configurazione criptato e/o inaccessibile con appropriati diritti (attenzione ambiente)

Falle nella Difesa/Porous Defenses

- CWE-732: Assegnamenti di permessi insicuri a risorse critiche
- Il software specifica permessi per una risorsa critica che la rendono modificabile o accessibile da utenti non autorizzati

- Nell'accesso ad una risorsa critica (es. file di configurazione) controllare che abbia le protezioni previste, eventualmente generare un errore/eccezione o interrompere ed uscire dal programma per pericolo intrusione
- Settare i permessi default allo startup con privilegio minimo
- Non assumere che l'amministratore di sistema imposterà i settaggi raccomandati

Falle nella Difesa/Porous Defenses

- CWE-330: Uso di valori non sufficientemente casuali/random
- Il software utilizza numeri non sufficientemente casuali in un contesto di sicurezza che dipende da numeri non predicibili

- NON reimplementare
- Attenzione ai seed (almeno 256 bit) dei generatori random.
- Attenzione ai seed di origine statistica: es data/ora

Consigliato "reseed" da dispositivi hardware

```
String GenerateReceiptURL(String baseUrl) {
  Random ranGen = new Random();
  ranGen.setSeed((new Date()).getTime());
  return(baseUrl + Gen.nextInt(400000000) + ".html");
}
```

Falle nella Difesa/Porous Defenses

- CWE-250: Esecuzione con privilegi non necessari
- Il software viene eseguito a livelli di privilegio più alti del necessario cosa che crea punti deboli o amplifica quelli già esistenti

Observed Examples

Reference	Description
CVE-2007-3931	Installation script installs some programs as setuid when they shouldn't be.
CVE-2007-4217	FTP client program on a certain OS runs with setuid privileges and has a buffer overflow. Most clients do not need extra privileges, so an overflow is not a vulnerability for those clients.
CVE-2007-5159	OS incorrectly installs a program with setuid privileges, allowing users to gain privileges.
CVE-2008-0162	Program does not drop privileges before calling another program, allowing code execution.
CVE-2008-0368	setuid root program allows creation of arbitrary files through command line argument.
CVE-2008-1877	Program runs with privileges and calls another program with the same privileges, which allows read of arbitrary files.
CVE-2008-4638	Composite: application running with high privileges allows user to specify a restricted file to process, which generates a parsing error that leaks the contents of the file.

- Validazione input codice con privilegi
- Rilascio privilegi più possibile anticipato, controllo del rilascio

Falle nella Difesa/Porous Defenses

- CWE-602: Rispetto Client-Side della sicurezza Server-Side Security

"*Client ricchi possono rendere gli attaccanti ricchi ed i clienti più poveri* "

- Il software ha un server che si basa sul fatto che il client implementi meccanismo con lo scopo di proteggere il server

Falle nella Difesa/Porous Defenses

- Il server accetta 2 comandi, "AUTH" che autentica lo user, e "CHANGE-ADDRESS" che aggiorna

This example contains client-side code that checks if the user authenticated successfully before sending a command. The server-side code performs the authentication in one step, and executes the command in a separate step.

CLIENT-SIDE (client.pl)

```
$server = "server.example.com";
$username = AskForUserName();
$password = AskForPassword();
$address = AskForAddress();
$sock = OpenSocket($server, 1234);
writeSocket($sock, "AUTH $username $password\n");
$resp = readSocket($sock);
if ($resp eq "success") {
  # username/pass is valid, go ahead and update the info!
  writeSocket($sock, "CHANGE-ADDRESS $username $address\n";
}
else {
  print "ERROR: Invalid Authentication!\n";
}
```

SERVER-SIDE (server.pl):

```
$sock = acceptSocket(1234);
($cmd, $args) = ParseClientRequest($sock);
if ($cmd eq "AUTH") {
  ($username, $pass) = split(/\s+/, $args, 2);
  $result = AuthenticateUser($username, $pass);
  writeSocket($sock, "$result\n");
  # does not close the socket on failure; assumes the
  # user will try again
}
elsif ($cmd eq "CHANGE-ADDRESS") {
  if (validateAddress($args)) {
    $res = UpdateDatabaseRecord($username, "address", $args);
    writeSocket($sock, "SUCCESS\n");
  }
  else {
    writeSocket($sock, "FAILURE -- address is malformed\n");
  }
}
```

a un CHANGE-
ià effettuato la
esso dell'utente
he invia il

Falle nella Difesa/Porous Defenses

- CWE-602: Rispetto Client-Side della sicurezza Server-Side Security

"*Client ricchi possono rendere gli attaccanti ricchi ed i clienti più poveri* "

- Il software ha un server che si basa sul fatto che il client implementi meccanismo con lo scopo di proteggere il server

- Per ogni controllo di sicurezza lato client, duplicarlo lato server
- I controlli lato client sono comunque utili per:
  - Feedback utente
  - Carico sul server per errori in input
  - Controlli di intrusion detection, se il server riceve un input che non avrebbe dovuto passare il filtro lato client