8. Vulnerabilidades de Software

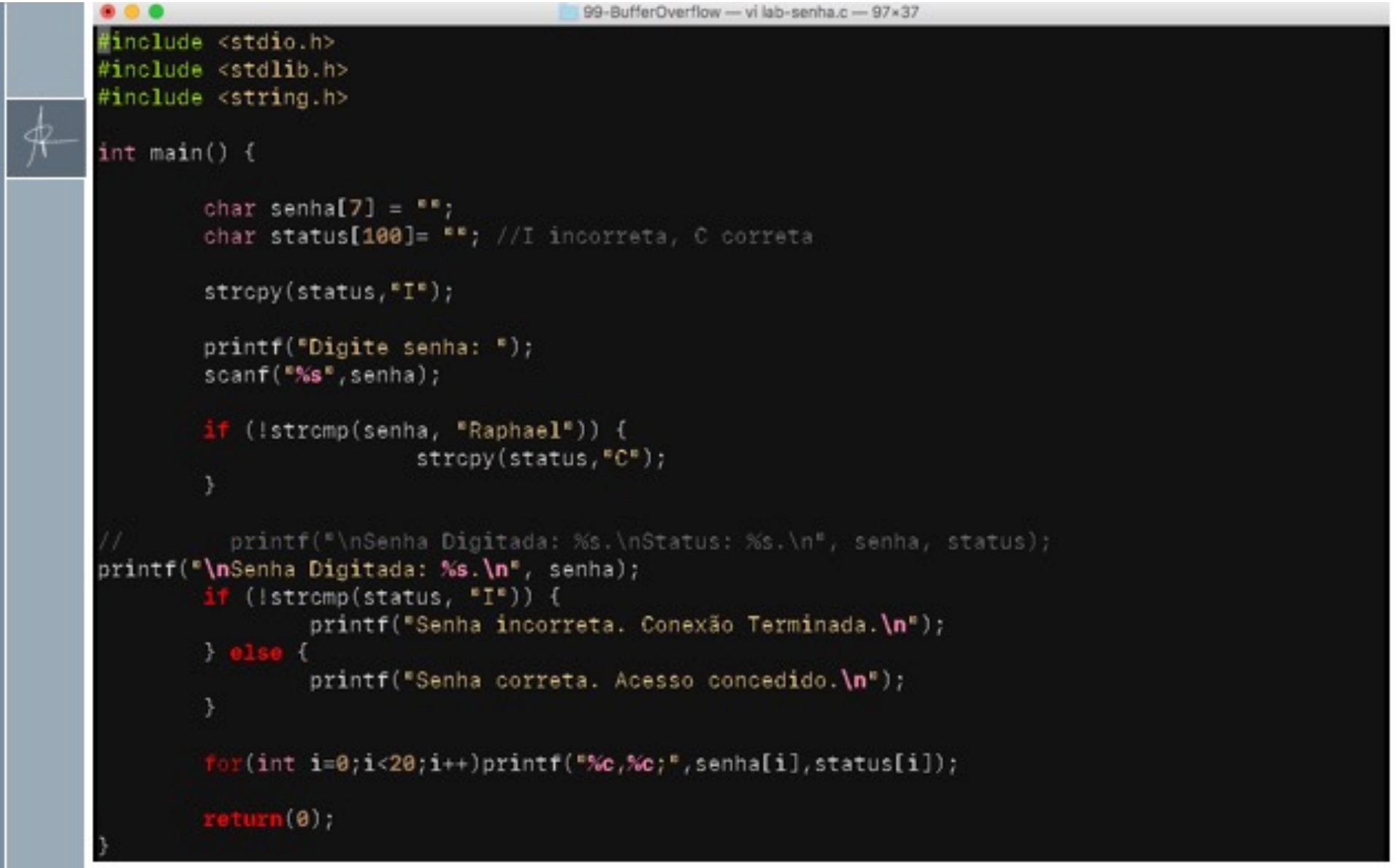
Vulnerabilidades/falhas de Software

- Falhas que concepção, projeto ou implementação que podem ser exploradas por atacantes
- Permitem que seu software passe por condições não previstas – e que podem ser exploradas por um atacante para comprometer a segurança
- Podem ter diferentes tipos de impacto e diferentes graus de dificuldade de descoberta e exploração
  - Impacto e explorabilidade são critérios típicos de classificação
- Vejamos alguns exemplos nos próximos slides

Exemplos de Falhas e Vulnerabilidades

Falha: Estouro de buffer

- Certamente, você já escutou muitas vezes
- Acontece quando um programa permite ao usuário escrever dados além dos limites de memória alocada
  - O programa pode travar – ou um atacante pode tomar controle da aplicação
  - Ainda é possível em linguagens como C# e Java (pois usam bibliotecas escritas em C/C++), mas é improvável.


99-BufferOverflow — vi lab-senha.c — 97×37
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
char senha[7] = "";
char status[100]= ""; //I incorreta, C correta
strcpy(status,"I");
printf("Digite senha: ");
scanf("%s",senha);
if (!strcmp(senha, "Raphael")) {
strcpy(status,"C");
}
// printf("\nSenha Digitada: %s.\nStatus: %s.\n", senha, status);
printf("\nSenha Digitada: %s.\n", senha);
if (!strcmp(status, "I")) {
printf("Senha incorreta. Conexão Terminada.\n");
} else {
printf("Senha correta. Acesso concedido.\n");
}
for(int i=0;i<20;i++)printf("%c,%c;",senha[i],status[i]);
return(0);
}

```
MacBook-Air-de-Raphael:99-BufferOverflow raphaelmachado$ gcc lab-senha.c
MacBook-Air-de-Raphael:99-BufferOverflow raphaelmachado$ ./a.out
Digite senha: Joao

Senha Digitada: Joao.
Senha incorreta. Conexão Terminada.
J,I;o,;a,;o,;;;;;;;;;;;;;I,;;;;;;;;;;;;;;;;MacBook-Air-de-Raphael:99-BufferOverflow raphaelmac
hado$ ./a.out
Digite senha: Pedro

Senha Digitada: Pedro.
Senha incorreta. Conexão Terminada.
P,I;e,;d,;r,;o,;;;;;;;;;;;;I,;;;;;;;;;;;;;;;;MacBook-Air-de-Raphael:99-BufferOverflow raphaelma
chado$ ./a.out
Digite senha: Raphael

Senha Digitada: Raphael.
Senha correta. Acesso concedido.
R,C;a,;p,;h,;a,;e,;l,;;;;;;;;C,;;;;;;;;;;;;;;;;MacBook-Air-de-Raphael:99-BufferOverflow raphael
machado$ ./a.out
Digite senha: 12345678901234567890123456789012345678901234567890

Senha Digitada: 12345678901234567890123456789012345678901234567890.
Senha correta. Acesso concedido.
1,2;2,3;3,4;4,5;5,6;6,7;7,8;8,9;9,0;0,1;1,2;2,3;3,4;4,5;5,6;6,7;7,8;8,9;9,0;0,1;MacBook-Air-de-Ra
phael:99-BufferOverflow raphaelmachado$
```

Exemplos de Estouro de Buffer

- CVE-2008-0778
  - Multiple stack-based buffer overflows in an ActiveX control in QTPlugin.ocx for Apple QuickTime 7.4.1 and earlier allow remote attackers to cause a **denial of service (crash)** and **possibly execute arbitrary code** via long arguments to the (1) SetBgColor, (2) SetHREF, (3) SetMovieName, (4) SetTarget, and (5) SetMatrix methods.
- CVE-2003-0352
  - Buffer overflow in a DCOM interface for RPC in Windows ... allows remote attackers to **execute arbitrary code** via a malformed message, as exploited by Blaster/MSblast/LovSAN/Nachi/Welcia worms

Falha: Format String

- Um problema típico de C/C++
  - Mas o "conceito" se repete em outros ambientes
- String de entrada avaliada como código
  - Espécie de "injeção"de código"
- Exemplo

```
void main(int argc, char * argv[])
{
        printf(argv[1]);
}
```

- Simples – o que pode dar errado?

Falha: Format String

› E se o programa é executado como :

<executável> “%x %x”

› A saída é algo como...

12FFC0 4011E5

O %x lê a pilha 4 bytes por vez e imprime
Vaza informação importante para um atacante

Execução do exemplo anterior

```
MacBook-Air-de-Raphael:teste raphaelmachado$ gcc main.c
main.c:1:1: warning: return type of 'main' is not 'int' [-Wmain-return-type]
void main(int argc, char * argv[])
^
main.c:1:1:       change return type to 'int'
void main(int argc, char * argv[])
^~~~
int
main.c:3:2: warning: implicitly declaring library function 'printf' with type
      'int (const char *, ...)' [-Wimplicit-function-declaration]
        printf(argv[1]);
        ^
main.c:3:2:       include the header <stdio.h> or explicitly provide a
      declaration for 'printf'
main.c:3:9: warning: format string is not a string literal
      (potentially insecure) [-Wformat-security]
        printf(argv[1]);
               ^~~~~~~
main.c:3:9:       treat the string as an argument to avoid this
        printf(argv[1]);
               ^
               "%s",
3 warnings generated.
MacBook-Air-de-Raphael:teste raphaelmachado$
```

Execução do exemplo anterior

```
MacBook-Air-de-Raphael:teste raphaelmachado$ ./a.out
Segmentation fault: 11
MacBook-Air-de-Raphael:teste raphaelmachado$ ./a.out "%x %x"
51ba5ad8 51ba5af0MacBook-Air-de-Raphael:teste raphaelmachado$
```

Exemplos

- CVE-2000-0573
  - The lreply function in wu-ftpd 2.6.0 and earlier does not properly cleanse an untrusted format string, which **allows remote attackers to execute arbitrary commands**.
- CVE-2007-4708 TA07-352A
  - Format string vulnerability in Address Book in Apple Mac OS X 10.4.11 allows remote attackers to **execute arbitrary code** via the URL handler.

Falha: Transbordamento de Inteiros

- Quando um inteiro fica muito grande para a quantidade alocada e "cicla"
  - Para unsigned, número grande vira zero
  - Para signed, número grande fica negativo
- Erros acontecem quando números são multiplicados, somados etc. e transbordam
  - Resultados podem ser muito ruins e imprevisíveis
  - Particularmente crítico se atacante conhece a lógica do programa e sabe como explorar
- Às vezes causado por casting

Examplos

- CVE-2008-0726
  - Integer overflow in Adobe Reader and Acrobat 8.1.1 and earlier allows remote attackers to **execute arbitrary code** via crafted arguments to the printSepsWithParams, which triggers memory corruption.
- CVE-2007-6261
  - Integer overflow in the load_threadstack function in the Mach-O loader (mach_loader.c) in the xnu kernel in Apple Mac OS X 10.4 through 10.5.1 allows local users to cause a **denial of service** (infinite loop) via a crafted Mach-O binary.

Falha: Injeção de SQL

- Sabe como atacantes conseguem descobrir números de cartão de crédito do seu site?
  - "Quebram" seu servidor explorando vulnerabilidades tais como estouro de buffer
  - Acessam uma porta de comunicação aberta com uma senha de admin
  - Engenharia social
  - Injeção de SQL

Exemplo de Injeção de SQL

› Código em PHP

```
$id = $_REQUEST[“id”];
$pass = $_REQUEST[“password”];
$qry = “SELECT ccnum FROM cust WHERE id = $id AND pass=$pass”;
```

Exemplo de Injeção de SQL

› Código em PHP

```
$id = $_REQUEST[“id”];
$pass = $_REQUEST[“password”];
$qry = “SELECT ccnum FROM cust WHERE id = ‘$id’ AND pass=‘$pass’”;
```

Atacante inclui id do usuário alvo

Para a senha, entrega: `' OR 1=1 –`

onde – ‘é o operador de comenário (ignora o que vem depois)

Examplos

- CAN-2004-0348
  - SQL injection vulnerability in viewCart.asp in SpiderSales shopping cart software allows remote attackers to execute arbitrary SQL via the userID parameter
- CVE-2008-0682
  - SQL injection vulnerability in wordspew-rss.php in the Wordspew plugin for Wordpress allows remote attackers to execute arbitrary SQL commands via the id parameter.

Falha: Injeção de comandos

› Em 1994, você poderia obter acesso de root num shel de IRIX enviando o seguinte comando para uma impressora:

```
FRED; xterm&
```

› Code:

```
char buf[1024];
snprintf(buf, “system lpr –P %s”, user_input, sizeof(buf) -1);
system(buf);
```

Falha: Cross Site Scripting

- De certa forma, nome inadequado – nem sempre precisa "cruzar sites"
  - Aliás, os exemplos cross-site são mais difíceis de entender=)
- Falha é bem simples
  - Aplicacão web recebe entrada de um usuário
  - Entrada é armazenada e enviada para outro usuário
  - Só isso=)

PHP Example

```
<?php

        if($_SERVER['REQUEST_METHOD'] != "POST")
        {
                header("Content-Type: text/html");
                print("<HTML><HEAD><TITLE>My Page</TITLE>");
                print("</HEAD>");
                print("<BODY>");
                print("<FORM method=post action='cssSin.php'>");
                print("Enter your comment.<p>");
                print("<INPUT type=text name='comment'>");
                print("<INPUT type=submit value='Submit'>");
                print("</FORM>");

                print("<HR>");
                print("<B>Here is the comment log:</B><p>");
```

PHP Example

```
                $f = fopen("c:\\comments.txt","r");
                print("<UL>");
                while (!feof($f))
                {
                        $line = fgets($f,2000);
                        print("<li>" . $line . "</LI>");
                }
                fclose($f);
                print("</UL>");
        }
        else {
                $comment = $_REQUEST['comment'];

                $f = fopen("c:\\comments.txt", "a");
                fprintf($f, "\n" . $comment);
                fclose($f);

                print("Thank you, your comment has been saved.");
        }
?>
```

O Problema

- Usuário malicioso pode injetar código script que depois será executado quando outro usuário vê a página
- O usuário malicioso pode...
  - Atrair uma vítima para sua página
  - Levar a vítima a clicar um link
  - Rodar o script sob o domínio do servidor para recuperar um cookie e obter informação sensível

Falha: Campos Ocultos em Forms

- Uso de campos ocultos para passar informações/variáveis sensíveis/críticas

```
<form action = " ..."
        <input type=text name="product">
        <input type=hidden name="price" value="300">
</form>
```

Falha: Uso inadequado de cripto (SSL/TLS)

- Se a autenticação não for feita de maneira adequada, atacante pode escutar/modificar conversas
- Impressão de que o site é impenetrável simplesmente porque usa SSL
  - Ainda pode ter overflow, SQLi etc...

Falha: senhas fracas / mau-uso de senhas

- Pessoas não gostam de senhas – é uma batalha convencê-las a usar senhas
- Armazenamento de senhas em claro
- Processo de redefinição de senhas
- Uso de ”perguntas” como alternativa a senhas
- Bloqueio de contas após erros (DoS)

Bases de riscos e vulnerabilidades

Bases de riscos e vulnerabilidades

- Grandes organizações dedicam-se a catalogar e organizar riscos vulnerabilidades de software
- Exemplos
  - CVE
  - NVDB
  - CWE
- Listas de riscos mais relevantes
  - SANS/CWE Top 25
  - OWASP Top 10

Common Vulnerabilities and Exposures (CVE)

- Fornece um método de referência para vulnerabilidades e exposições de segurança da informação conhecidas publicamente.
- A National Cybersecurity FFRDC, operada pela Mitre Corporation, mantém o sistema
  - financiamento da Divisão Nacional de Segurança Cibernética do Departamento de Segurança Interna dos Estados Unidos.
- O Security Content Automation Protocol usa o CVE, e os CVE IDs estão listados no sistema do MITRE, bem como no National National Vulnerability Database.

Exemplo de CVE

CVE-ID	
CVE-2019-0801	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

A remote code execution vulnerability exists when Microsoft Office fails to properly handle certain files.To exploit the vulnerability, an attacker would have to convince a user to open a specially crafted URL file that points to an Excel or PowerPoint file that was also downloaded.The update addresses the vulnerability by correcting how Office handles these files., aka 'Office Remote Code Execution Vulnerability'.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- MISC:https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2019-0801
- MISC:https://www.zerodayinitiative.com/advisories/ZDI-19-358/

Assigning CNA

Microsoft Corporation

Exemplo de CVE

Observe que o CVE é bastante específico (falha em particular do Excel ou PowerPoint)

CVE-ID

CVE-2019-0801 Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

A remote code execution vulnerability exists when Microsoft Office fails to properly handle certain files.To exploit the vulnerability, an attacker would have to convince a user to open a specially crafted URL file that points to an Excel or PowerPoint file that was also downloaded.The update addresses the vulnerability by correcting how Office handles these files., aka 'Office Remote Code Execution Vulnerability'.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- MISC:https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2019-0801
- MISC:https://www.zerodayinitiative.com/advisories/ZDI-19-358/

Assigning CNA

Microsoft Corporation

Exemplo de CVE

Exploitability Assessment

The following table provides an exploitability assessment for this vulnerability at the time of original publication.

Publicly Disclosed	Exploited	Latest Software Release	Older Software Release	Denial of Service
No	No	1 - Exploitation More Likely	1 - Exploitation More Likely	Not Applicable

Security Updates

To determine the support life cycle for your software version or edition, see the Microsoft Support Lifecycle.

Product ▲	Platform	Article	Download	Impact	Severity	Supersedence
Microsoft Office 2010 Service Pack 2 (32-bit editions)		4462223	Security Update	Remote Code Execution	Important	4462174
Microsoft Office 2010 Service Pack 2 (64-bit editions)		4462223	Security Update	Remote Code Execution	Important	4462174
Microsoft Office 2013 RT Service Pack 1		4464504	Security Update	Remote Code Execution	Important	4462138
Microsoft Office 2013 Service Pack 1 (32-bit editions)		4464504	Security Update	Remote Code Execution	Important	4462138
Microsoft Office 2013 Service Pack 1 (64-bit editions)		4464504	Security Update	Remote Code Execution	Important	4462138
Microsoft Office 2016 (32-bit edition)		4462242	Security Update	Remote Code Execution	Important	4462146
Microsoft Office 2016 (64-bit edition)		4462242	Security Update	Remote Code Execution	Important	4462146
Microsoft Office 2019 for 32-bit editions		Click to Run	Security Update	Remote Code Execution	Important	
Microsoft Office 2019 for 64-bit editions		Click to Run	Security Update	Remote Code Execution	Important	
Office 365 ProPlus for 32-bit Systems		Click to Run	Security Update	Remote Code Execution	Important	
Office 365 ProPlus for 64-bit Systems		Click to Run	Security Update	Remote Code Execution	Important	

National Vulnerability Database (NVD)

- O NVD é o repositório do governo dos EUA de dados de gerenciamento de vulnerabilidade baseados em padrões representados usando o SCAP (Security Content Automation Protocol).
- Esses dados permitem a automação do gerenciamento de vulnerabilidades, medidas de segurança e conformidade.
- O NVD inclui bancos de dados de listas de verificação de segurança, falhas de software relacionadas à segurança, configurações incorretas, nomes de produtos e métricas de impacto.
- NVD vs CVE: O NVD é o dicionário CVE, acrescido de análises adicionais, um banco de dados e um mecanismo de busca refinado.
  - O NVD é um superconjunto do CVE. O NVD é sincronizado com o CVE, de forma que qualquer atualização do CVE apareça imediatamente no NVD.

Common Weakness Enumeration (CWE)

- Sistema de categorização para fraquezas e vulnerabilidades de software.
- Também é patrocinado pela MITRE
- CWE foca nas fraquezas "genéricas", CVE lista as instâncias de falha
  - Exemplo: *buffer overflow* versus **o** *buffer overflow da versão X.Y.Z do software ABX*

Exemplo de CWE

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Weakness ID: 89
Abstraction: Base
Structure: Simple

Status: Draft

Presentation Filter: High Level

Description

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.

Extended Description

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

SQL injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

Exemplo de CWE

Observe que o CWE é genérico (falha em comando SQL)

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Weakness ID: 89
Abstraction: Base
Structure: Simple

Status: Draft

Presentation Filter: High Level

Description

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.

Extended Description

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

SQL injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

Exemplo de CWE

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOf and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that the user may want to explore.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	B	943	Improper Neutralization of Special Elements in Data Query Logic
ParentOf	V	564	SQL Injection: Hibernate
CanFollow	B	456	Missing Initialization of a Variable

Relevant to the view "Weaknesses for Simplified Mapping of Published Vulnerabilities" (CWE-1003)

Nature	Type	ID	Name
ChildOf	B	943	Improper Neutralization of Special Elements in Data Query Logic

Relevant to the view "Architectural Concepts" (CWE-1008)

Nature	Type	ID	Name
MemberOf	C	1019	Validate Inputs

Relevant to the view "Development Concepts" (CWE-699)

Nature	Type	ID	Name
ChildOf	B	943	Improper Neutralization of Special Elements in Data Query Logic
ParentOf	V	564	SQL Injection: Hibernate

Relevant to the view "Weaknesses in OWASP Top Ten (2013)" (CWE-928)

Nature	Type	ID	Name
ParentOf	V	564	SQL Injection: Hibernate

Exemplo de CWE

Modes Of Introduction

The different Modes of Introduction provide information about how and when this weakness may be introduced. The Phase identifies a point in the software life cycle at which introduction may occur, while the Note provides a typical scenario related to introduction during the given phase.

Phase	Note
Architecture and Design	This weakness typically appears in data-rich applications that save user inputs in a database.
Implementation	REALIZATION: This weakness is caused during implementation of an architectural security tactic.
Operation	

Common Consequences

The table below specifies different individual consequences associated with the weakness. The Scope identifies the application security area that is violated, while the Impact describes the negative technical impact that arises if an adversary succeeds in exploiting this weakness. The Likelihood provides information about how likely the specific consequence is expected to be seen relative to the other consequences in the list. For example, there may be high likelihood that a weakness will be exploited to achieve a certain impact, but a low likelihood that it will be exploited to achieve a different impact.

Scope	Impact	Likelihood
Confidentiality	**Technical Impact:** *Read Application Data* Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL injection vulnerabilities.	
Access Control	**Technical Impact:** *Bypass Protection Mechanism* If poor SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password.	
Access Control	**Technical Impact:** *Bypass Protection Mechanism* If authorization information is held in a SQL database, it may be possible to change this information through the successful exploitation of a SQL injection vulnerability.	
Integrity	**Technical Impact:** *Modify Application Data* Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL injection attack.	

Likelihood Of Exploit

High

Exemplo de CWE

Likelihood Of Exploit

High

Memberships

This MemberOf Relationships table shows additional CWE Categories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name
MemberOf	V	635	Weaknesses Originally Used by NVD from 2008 to 2016
MemberOf	C	713	OWASP Top Ten 2007 Category A2 - Injection Flaws
MemberOf	C	722	OWASP Top Ten 2004 Category A1 - Unvalidated Input
MemberOf	C	727	OWASP Top Ten 2004 Category A6 - Injection Flaws
MemberOf	C	751	2009 Top 25 - Insecure Interaction Between Components
MemberOf	C	801	2010 Top 25 - Insecure Interaction Between Components
MemberOf	C	810	OWASP Top Ten 2010 Category A1 - Injection
MemberOf	C	864	2011 Top 25 - Insecure Interaction Between Components
MemberOf	V	884	CWE Cross-section
MemberOf	C	929	OWASP Top Ten 2013 Category A1 - Injection
MemberOf	C	990	SFP Secondary Cluster: Tainted Input to Command
MemberOf	C	1005	7PK - Input Validation and Representation
MemberOf	C	1027	OWASP Top Ten 2017 Category A1 - Injection
MemberOf	C	1131	CISQ Quality Measures - Security

Notes

Relationship

SQL injection can be resultant from special character mismanagement, MAID, or blacklist/whitelist problems. It can be primary to authentication errors.

More information is available — Please select a different filter.

Exemplo (do CWE)

The following code dynamically constructs and executes a SQL query that searches for items matching a specified name. The query restricts the items displayed to those where owner matches the user name of the currently-authenticated user.

Example Language: **C#**

```
...
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);
...
```

The query that this code intends to execute follows:

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

However, because the query is constructed dynamically by concatenating a constant base query string and a user input string, the query only behaves correctly if itemName does not contain a single-quote character. If an attacker with the user name wiley enters the string:

```
name' OR 'a'='a
```

for itemName, then the query becomes the following:

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

The addition of the:

```
OR 'a'='a
```

condition causes the WHERE clause to always evaluate to true, so the query becomes logically equivalent to the much simpler query:

```
SELECT * FROM items;
```

SANS/CWE Top 25

- CWE/SANS Top 25 Most Dangerous Software Errors (2011)
  - Erros de software mais comuns/críticos
- Geralmente são fáceis de encontrar e fáceis de explorar.
- Frequentemente permitem que invasores assumam completamente o software, roubem dados ou impeçam o software de funcionar.
- A lista é uma ferramenta de educação e conscientização
  - Ajuda programadores a evitar as vulnerabilidades, identificando e evitando erros comuns
- A lista é resultado da colaboração entre o SANS Institute, o MITRE e especialistas nos EUA e Europa.

SANS/CWE Top 25

- O Common Weakness Enumeration (CWE) é uma lista formal de tipos de fraquezas de software e é patrocinado pela Divisão Nacional de Segurança Cibernética do Departamento de Segurança Interna dos EUA.
  - O Instituto SANS (SysAdmin, Audit, Network, Security) foi estabelecido em 1989 como uma organização de pesquisa e educação.
- Fonte: http://www.sans.org/top25errors/

Participantes

›Robert C. Seacord, CERT
›Pascal Meunier, CERIAS, Purdue University
›Matt Bishop, University of California, Davis
›Kenneth van Wyk, KRvW Associates
›Masato Terada, Information-Technology Promotion Agency (IPA), (Japan)
›Sean Barnum, Cigital, Inc.
›Mahesh Saptarshi and Cassio Goldschmidt, Symantec Corporation
›Adam Hahn, MITRE
›Jeff Williams, Aspect Security
›Carsten Eiram, Secunia
›Josh Drake, iDefense Labs at VeriSign, Inc.
›Chuck Willis, MANDIANT
›Michael Howard, Microsoft
›Bruce Lowenthal, Oracle Corporation
›Mark J. Cox, Red Hat Inc.
›Jacob West, Fortify Software
›Djenana Campara, Hatha Systems
›James Walden, Northern Kentucky University
›Frank Kim, ThinkSec
›Chris Eng and Chris Wysopal, Veracode, Inc.
›Ryan Barnett, Breach Security
›Antonio Fontes, New Access SA, (Switzerland)
›Mark Fioravanti II, Missing Link Security Inc.
›Ketan Vyas, Tata Consultancy Services (TCS)
›Lindsey Cheng, Ian Peters and Tom Burgess, Secured Sciences Group, LLC
›Hardik Parekh and Matthew Coles, RSA - Security Division of EMC Corporation
Ivan Ristic Apple Product Security
›Software Assurance Forum for Excellence in Code (SAFECode)
›Core Security Technologies Inc.
›Depository Trust & Clearing Corporation (DTCC)

Reconhecimento

- National Security Agency's Information Assurance Directorate
  - "The publication of a list of programming errors that enable cyber espionage and cyber crime is an important first step in managing the vulnerability of our networks and technology. There needs to be a move away from reacting to thousands of individual vulnerabilities, and to focus instead on a relatively small number of software flaws that allow vulnerabilities to occur, each with a general root cause. Such a list allows the targeting of improvements in software development practices, tools, and requirements to manage these problems earlier in the life cycle, where they can be solved on a large scale and cost-effectively."
  - Tony Sager, National Security Agency's Information Assurance Directorate

Reconhecimento

- Microsoft:
  - "The 2009 CWE/SANS Top 25 Programming Errors project is a great resource to help software developers identify which security vulnerabilities are the most important to understand, prevent and fix."
  - Michael Howard, Principal Security Program Manager, Security Development Lifecycle Team, Microsoft Corp.
- Symantec:
  - "The 2009 CWE/SANS Top 25 Programming Errors reflects the kinds of issues we've seen in application software and helps provide us with actionable direction to continuously improve the security of our software."
  - - Wesley H. Higaki, Director, Software Assurance, Office of the CTO, Symantec Corporation

A lista Top 25

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

Organização em Categorias:

- **Insecure Interaction Between Components**
  - Essas fraquezas estão relacionadas a formas inseguras nas quais os dados são enviados e recebidos entre componentes, módulos, programas, processos, threads ou sistemas separados.
- **Risky Resource Management**
  - Os pontos fracos dessa categoria estão relacionados às maneiras pelas quais o software gerencia (inadequadamente) a criação, o uso, a transferência ou a destruição de recursos importantes do sistema.
- **Porous Defenses**
  - As fraquezas nessa categoria estão relacionadas a técnicas defensivas que são frequentemente mal utilizadas, abusadas ou simplesmente ignoradas.

Insecure Interaction Among Components

- CWE-20: Improper Input Validation
- CWE-116: Improper Encoding or Escaping of Output
- CWE-89: Failure to Preserve SQL Query Structure (aka 'SQL Injection')

Insecure Interaction Among Components

- CWE-79: Failure to Preserve Web Page Structure (aka 'Cross-site Scripting')
- CWE-78: Failure to Preserve OS Command Structure (aka 'OS Command Injection')
- CWE-319: Cleartext Transmission of Sensitive Information

Insecure Interaction Among Components

- CWE-352: Cross-Site Request Forgery (CSRF)
- CWE-362: Race Condition
- CWE-209: Error Message Information Leak

Risky Resource Management

- CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer
- CWE-642: External Control of Critical State Data
- CWE-73: External Control of File Name or Path

Risky Resource Management

- CWE-426: Untrusted Search Path
- CWE-94: Failure to Control Generation of Code (aka 'Code Injection')
- CWE-494: Download of Code Without Integrity Check

Risky Resource Management

- CWE-404: Improper Resource Shutdown or Release
- CWE-665: Improper Initialization
- CWE-682: Incorrect Calculation

Porous Defenses

- CWE-285: Improper Access Control (Authorization)
- CWE-327: Use of a Broken or Risky Cryptographic Algorithm
- CWE-259: Hard-Coded Password

Porous Defenses

- CWE-732: Insecure Permission Assignment for Critical Resource
- CWE-330: Use of Insufficiently Random Values

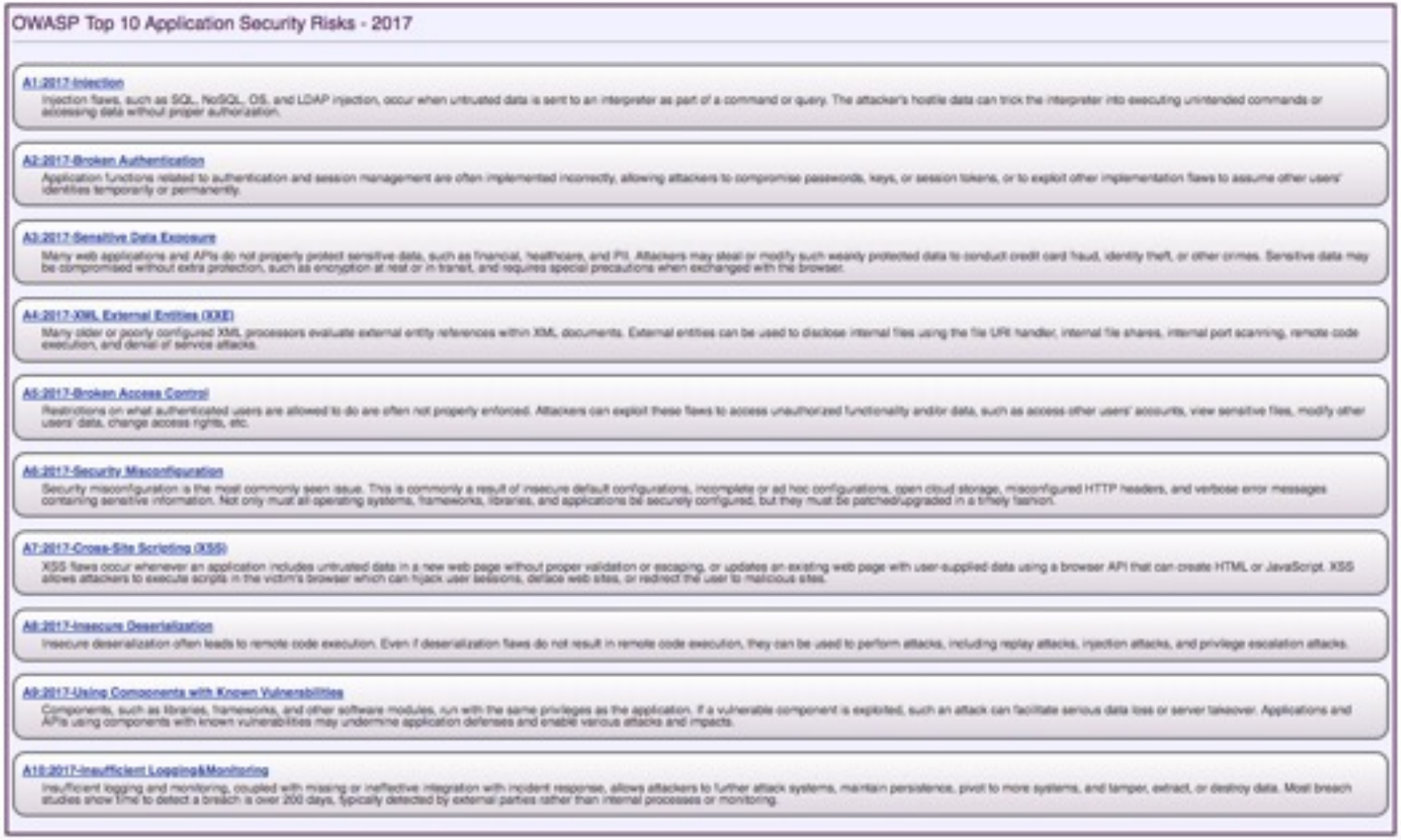
Porous Defenses

- CWE-250: Execution with Unnecessary Privileges
- CWE-602: Client-Side Enforcement of Server-Side Security

OWASP Top 10

- O Open Web Application Security Project (OWASP) é uma organização internacional dedicada a melhorar a segurança de aplicações web.
- O projeto OWASP Top 10 publica uma lista dos considerados 10 principais riscos de segurança de aplicações da web em todo o mundo.
- A lista descreve cada vulnerabilidade, fornece exemplos e oferece sugestões sobre como evitá-la.
- A versão mais recente da lista dos 10 melhores, publicada oficialmente em 2017.
- O OWASP priorizou o top 10 de acordo com sua prevalência e sua relativa capacidade de exploração, detectabilidade e impacto.

A Lista OWASP Top 10



OWASP Top 10 Application Security Risks - 2017

A1:2017-Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

A2:2017-Broken Authentication

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

A3:2017-Sensitive Data Exposure

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

A4:2017-XML External Entities (XXE)

Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

A5:2017-Broken Access Control

Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

A6:2017-Security Misconfiguration

Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

A7:2017-Cross-Site Scripting (XSS)

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

A8:2017-Insecure Deserialization

Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

A9:2017-Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

A10:2017-Insufficient Logging&Monitoring

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

A Lista OWASP Top 10

- A1:2017-Injection
- A2:2017-Broken Authentication
- A3:2017-Sensitive Data Exposure
- A4:2017-XML External Entities (XXE)
- A5:2017-Broken Access Control
- A6:2017-Security Misconfiguration
- A7:2017-Cross-Site Scripting (XSS)
- A8:2017-Insecure Deserialization
- A9:2017-Using Components with Known Vulnerabilities
- A10:2017-Insufficient Logging&Monitoring

OWASP Top 10 versus CWE Top 25

- OWASP originalmente focado em aplicações web
  - Atualmente, inclui vulnerabilidades mais "genéricas"
- OWASP é mais geral do que o CWE
  - Exemplo: vulnerabilidade OWASP A1 (Injection) está associada a seis vulnerabilidades CWE
- CWE é uma lista de vulnerabilidades enquanto OWASP é uma lista de riscos

OWASP A1: Injection

- CWE-78: Improper Neutralization of Special Elements Used in an OS Command ('OS Command Injection')
- CWE-89: SQL Injection
- CWE-94: Code Injection
- CWE-434: Unrestricted Upload of File with Dangerous Type
- CWE-494: Download of Code Without Integrity Check
- CWE-829: Inclusion of Functionality from Untrusted Control Sphere

OWASP A2: Broken Authentication

- CWE-306: Missing Authentication for Critical Function
- CWE-307: Improper Restriction of Excessive Authentication Attempts
- CWE-798: Use of Hard-coded Credentials
- CWE-807: Reliance on Untrusted Inputs in a Security Decision
- CWE-862: Missing Authorization
- CWE-863: Incorrect Authorization

OWASP A3: Sensitive Data Exposure

- CWE-311: Missing Encryption of Sensitive Data
- CWE-319: Cleartext Transmission of Sensitive Information

OWASP A4: XML External Entities

› Nenhum

OWASP A5: Broken Access Control

- CWE-73: External Control of File Name or Path
- CWE-285: Improper Authorization

OWASP A6: Security Misconfiguration

- CWE-250: Execution with Unnecessary Privileges
- CWE-676: Use of Potentially Dangerous Function
- CWE-732: Incorrect Permission Assignment for Critical Resource

OWASP A7: Cross-Site Scripting (XSS)

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

OWASP A8: Insecure Deserialization

› CWE-134: Use of Externally-Controlled Format String

OWASP A9: Using Components with Known Vulnerabilities

- CWE-190: Integer Overflow or Wraparound
- CWE-327: Use of a Broken or Risky Cryptographic Algorithm
- CWE-759: Use of a One-way Hash Without a Salt

OWASP A10: Insufficient Logging and Monitoring

› Nenhum

As vulnerabilidades da OWASP TOP 10

- Desenvolvido para descrever as 10 vulnerabilidades mais comuns em Aplicações Web e seus mecanismos de prevenção
  - Define de forma clara quais vulnerabilidades ocorrem com maior frequência nas Aplicações Web.
  - Apresenta e direciona propostas para medidas corretivas e preventivas, com suporte constante dos projetos disponibilizados pelo OWASP.
  - O objetivo primário é educar programadores, designers, arquitetos e organizações acerca das consequências das 10 vulnerabilidades mais comuns em Aplicações Web.

As vulnerabilidades da OWASP TOP 10

- O OWASP TOP 10 é uma lista de vulnerabilidades mais comuns e para garantir a eficácia no uso de suas recomendações, deve ser entendido exatamente dentro deste conceito
  - Não é e nem tem a intenção de ser uma lista exaustiva das vulnerabilidades que podem ocorrer em uma Aplicação Web
  - Deve ser utilizada como uma referência básica para adequação do nível de segurança em Aplicações Web e posterior desenvolvimento de um programa de proteção amplo

OWASP: Conceito de Risco em Aplicações

As Vulnerabilidades da OWASP TOP 10

- A1 - Injeção
- A2 - Quebra de Autenticação
- A3 – Exposição de Dados Sensíveis
- A4 – Entidades Externas de XML
- A5 – Quebra de Controle de Acesso
- A6 – Configuração de Segurança Incorretas
- A7 – Cross-Site Scripting (XSS)
- A8 – Desserialização Insegura
- A9 - Utilização de Componentes Vulneráveis
- A10 – Registro e Monitoramento Insuficientes

A OWASP TOP 10 – 2013 *vs* 2017

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Resumo

- O OWASP TOP 10 é uma lista de vulnerabilidades mais comuns e para garantir a eficácia no uso de suas recomendações, deve ser entendido exatamente dentro deste conceito
  - Não é e nem tem a intenção de ser uma lista exaustiva das vulnerabilidades que podem ocorrer em uma Aplicação Web
  - Deve ser utilizada como uma referência básica para adequação do nível de segurança em Aplicações Web e posterior desenvolvimento de um programa de proteção amplo