

# **Validação de entrada inadequada**

# Validação de entrada inadequada

- ❑ Este é um típico problema de vulnerabilidade em que o software não faz validações adequadas das entradas de dados, podendo afetar o fluxo de execução do programa
- ❑ Ocorre pois o programador não acredita que certas entradas podem ser modificadas pelo agente
- ❑ Fontes de entrada podem ser:
  - Formulários
  - Cookies
  - APIs
  - Parâmetros na URL
  - Cabeçalhos HTTP
  - Arquivos de upload

# Exemplo – Campos ocultos

Campo 1

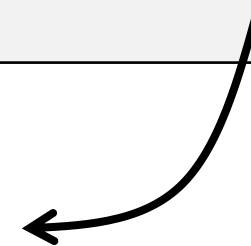
Campo 2

Campo 3

Campo 4

Campo 5

```
{  
  "campo1" : "abc",  
  "campo2" : "def",  
  "campo3" : "ghi",  
  "campo4" : "jkl",  
  "campo5" : "mno",  
}
```




# Exemplo – Campos ocultos

```
<input type="hidden" name="campo1" value="abc">
```

Campo 1

Campo 2

Campo 3



Campo 4

Campo 5

<http://servidor.com/pagina2.html?campo1=abc&campo2=def&campo3=ghi>

# Validação de entrada inadequada

- ❑ Os campos ocultos são utilizados para enviar um valor do servidor para o cliente e (sem que o valor seja alterado), enviar novamente para o servidor.
- ❑ Em aplicações web, muitos programadores acreditam que os cookies e campos ocultos do formulário não podem ser modificados no navegador.

# Validação de entrada

## *consequências*

- ❑ Um agente pode submeter valores inesperados que podem abortar a execução do software, causar loopings infinitos ou consumo excessivo de recursos
- ❑ Para evitar tais vulnerabilidades, deve-se aplicar a seguinte prática para desenvolvimento de sistemas seguros:

Tratar todas as entradas do sistema  
como não confiáveis

- ❑ “(...) sempre que receber dados, seja através de entrada direta de dados pelo usuário, seja pelo recebimento de dados de outro sistema, estes devem passar por uma verificação de integridade e consistência.” (ALBULQUERQUE, 2002)

# Validação de entrada

## *consequências*

- ❑ Um agente pode submeter valores inesperados que podem abortar a execução do software, causar loopings infinitos ou consumo excessivo de recursos
- ❑ Para evitar tais vulnerabilidades, deve-se aplicar a seguinte prática para desenvolvimento de sistemas seguros:

Tratar todas as entradas do sistema  
como não confiáveis

- ❑ “(...) sempre que receber dados, seja através de entrada direta de dados pelo usuário, seja pelo recebimento de dados de outro sistema, estes devem passar por uma verificação de integridade e consistência.” (ALBULQUERQUE, 2002)

# Validação de entrada

- ❑ Deve-se assumir que todas as entradas podem fornecer dados maliciosos.
- ❑ Existem duas técnicas para realizar a validação de dados de entrada:
  - Aplicação de lista de permissões (allowlist) e
  - Aplicação de lista de bloqueios (blocklist)



# Validação de entrada

## Lista de permissões (*allowlist*)

```
def upload_arquivo(nome_arquivo, dados_bytes):  
    _, extensao = nome_arquivo.rsplit(".", 1)  
    extensao = "." + extensao.lower()  
  
    extensoes_permitidas = {".jpg", ".png", ".pdf", ".txt", ".docx"}  
  
    if extensao not in extensoes_permitidas:  
        raise Exception("Extensão não permitida")  
  
    with open(nome_arquivo, "wb") as f:  
        f.write(dados_bytes)  
  
    return
```

# Validação de entrada

## Lista de bloqueios (*blocklist*)

```
def upload_arquivo(nome_arquivo, dados_bytes):  
    _, extensao = nome_arquivo.rsplit(".", 1)  
    extensao = "." + extensao.lower()  
  
    extensoes_bloqueadas = {".exe", ".bat", ".js", ".vbs"}  
  
    if extensao in extensoes_bloqueadas:  
        raise Exception("Upload proibido")  
  
    # Salva o vetor de bytes no disco  
    with open(nome_arquivo, "wb") as f:  
        f.write(dados_bytes)  
  
    return
```

# Validação de entrada

- ❑ A recomendação é criar uma lista de entradas aceitáveis (“lista de permissões”). Desta forma, qualquer entrada que não seguir esta definição, deverá ser recusada (ou transformada).
- ❑ A recomendação é de não confiar exclusivamente em observar entradas maliciosas ou malformadas (isto é, não confiar numa “lista de bloqueio”).
  - É mais provável que seja esquecido alguma entrada não desejada
  - Listas de bloqueio podem ser úteis para detectar ataques potenciais

# Validação de entrada

- ❑ A lista negra pode ser combinada com a técnica de saneamento de dados
  - Nesta técnica, os valores da lista negra são substituídos por outra sequência de caracteres, que são seguros
  
- ❑ Quando for desenvolver validação de dados de entrada, deve-se considerar todas as propriedades relevantes, incluindo:
  - quantidade de caracteres,
  - tipo de dado de entrada,
  - o intervalo completo de valores aceitáveis,
  - consistência entre campos relacionados e
  - conformidade às regras de negócios.

# Funções seguras

- ❑ Outra prática para construção de software seguro é:

Criar e usar funções intrinsecamente seguras

- ❑ Todas as funções deveriam validar os dados de entrada para impedir perda de controle do sistema ou qualquer tipo de falha.
  - Isso significa que cada função (método) deve verificar se os parâmetros estão dentro de um intervalo de valores aceitáveis e não contém caracteres estranhos ou má formação, antes de qualquer tentativa de uso.
- ❑ Também recomenda-se sempre retornar exceção quando algum problema for detectado na entrada ou durante a execução da função

# Implicação de práticas seguras

- ❑ “(...) muitas vezes um código seguro implica uma certa perda de desempenho. Isto é inevitável devido ao maior controle dentro do sistema, mas pode ser facilmente compensado com um hardware mais rápido. Já para um código inseguro, não há hardware que dê jeito” (ALBUQUERQUE, 2002).

# **Catálogo de vulnerabilidades**

# Catálogo de vulnerabilidades

- ❑ Os seguintes catálogos cobrem vulnerabilidades comuns na codificação de aplicações web:
  - OWASP – Open Web Application Security Project
  - PCI DSS – Payment Card Industry Data Security Standard
  
- ❑ CWE – Common Weakness Enumeration
  
- ❑ Os catálogos fornecem guias para orientar desenvolvedores para identificar e prevenir fraquezas.



CWE - 2024 CWE Top 25 Most Dangerous

https://cwe.mitre.org/data/top25most/

Home > CWE Top 25 > 2024

HomeAbout

2024 CWE Top 25 Most Dangerous

Top 25 Home

Share via:

1

Improper Neutralization of Scripting'')  
[CWE-79](#) | CVEs in KEV: 1 | Rank Last Year: 16 (down 3) ▼

2

Out-of-bounds Read  
[CWE-787](#) | CVEs in KEV: 1 | Rank Last Year: 15 (down 3) ▼

3

Improper Neutralization of Injection'')  
[CWE-89](#) | CVEs in KEV: 1 | Rank Last Year: 14 (down 3) ▼

4

Cross-Site Request Forgery (CSRF)  
[CWE-352](#) | CVEs in KEV: 1 | Rank Last Year: 13 (down 3) ▼

5

Improper Limit Checking of User Input  
[CWE-22](#) | CVEs in KEV: 1 | Rank Last Year: 12 (down 3) ▼

6

Out-of-bounds Write  
[CWE-125](#) | CVEs in KEV: 1 | Rank Last Year: 11 (down 3) ▼

7

Improper Neutralization of Command Injection'')  
[CWE-78](#) | CVEs in KEV: 1 | Rank Last Year: 10 (down 3) ▼

8

Use After Free  
[CWE-416](#) | CVEs in KEV: 1 | Rank Last Year: 9 (down 3) ▼

9

Missing Authorization  
[CWE-862](#) | CVEs in KEV: 1 | Rank Last Year: 8 (down 3) ▼

10

Unrestricted Upload of File with Arbitrary Extensions  
[CWE-434](#) | CVEs in KEV: 1 | Rank Last Year: 7 (down 3) ▼

11

Improper Control of Resource Interactions  
[CWE-94](#) | CVEs in KEV: 1 | Rank Last Year: 6 (down 3) ▼

12

Improper Input Validation  
[CWE-20](#) | CVEs in KEV: 1 | Rank Last Year: 5 (down 3) ▼

13

Improper Neutralization of Injection'')  
[CWE-77](#) | CVEs in KEV: 1 | Rank Last Year: 4 (down 3) ▼

14

Improper Authentication  
[CWE-287](#) | CVEs in KEV: 1 | Rank Last Year: 3 (down 3) ▼

15

Improper Privilege Management  
[CWE-269](#) | CVEs in KEV: 1 | Rank Last Year: 2 (down 3) ▼

16

Deserialization of Untrusted Data  
[CWE-502](#) | CVEs in KEV: 1 | Rank Last Year: 1 (down 3) ▼

17

Exposure of Sensitive Information to an Unauthorized Actor  
[CWE-200](#) | CVEs in KEV: 1 | Rank Last Year: 0 (down 3) ▼

18

Incorrect Authorization  
[CWE-863](#) | CVEs in KEV: 1 | Rank Last Year: 0 (down 3) ▼

19

Server-Side Request Forgery (SSRF)  
[CWE-918](#) | CVEs in KEV: 2 | Rank Last Year: 19 (up 2) ▲

20

Improper Restriction of Operations within the Bounds of a Memory Buffer  
[CWE-119](#) | CVEs in KEV: 2 | Rank Last Year: 17 (down 3) ▼

21

NULL Pointer Dereference  
[CWE-476](#) | CVEs in KEV: 0 | Rank Last Year: 12 (down 9) ▼

22

Use of Hard-coded Credentials  
[CWE-798](#) | CVEs in KEV: 2 | Rank Last Year: 18 (down 4) ▼

23

Integer Overflow or Wraparound  
[CWE-190](#) | CVEs in KEV: 3 | Rank Last Year: 14 (down 9) ▼

24

Uncontrolled Resource Consumption  
[CWE-400](#) | CVEs in KEV: 0 | Rank Last Year: 37 (up 13) ▲

25

Missing Authentication for Critical Function  
[CWE-306](#) | CVEs in KEV: 5 | Rank Last Year: 20 (down 5) ▼

Page Last Updated: November 20, 2024

BACK TO TOP