OWASP TOP 10

PolarGram



|  |  |
| --- | --- |
| **Date** | 01-06-2022 |
| **Version** | 0.1 |
| **Student Name** | Lars van den Brandt |
| **Student Number** | 434565 |
| **Teacher Fontys** | Tom Meulensteen & Marcel Boelaars |

Inhoud

[Wat is de owasp top 10? 3](#_Toc104992507)

[Owasp top 10 2021 binnen mijn applicatie 3](#_Toc104992508)

[1. Broken Access Control 3](#_Toc104992509)

[2. Cryptographic Failures 4](#_Toc104992510)

[3. Injection 5](#_Toc104992511)

[4. Insecure Design 6](#_Toc104992512)

[5. Security Misconfiguration 7](#_Toc104992513)

[6. Vulnerable and Outdated Components 8](#_Toc104992514)

[7. Identification and Authentication Failures 9](#_Toc104992515)

[8. Software and Data Integrity Failures 10](#_Toc104992516)

[9. Secure Logging and Monitoring Falures 11](#_Toc104992517)

[10. Server-Side Request Forgery 12](#_Toc104992518)

# Wat is de owasp top 10?[[1]](#footnote-1)

De OWASP Top 10 bestaat uit een document met een checklist[[2]](#footnote-2), dat bewustzijn creëert over het beveiligen van een web applicatie. Het beschikt over de top 10, meest voor de hand liggende, maar wel gevaarlijke veiligheidsrisico’s Het document wordt samengesteld uit ervaringen van experts over de hele wereld.

OWASP Top 10 is misschien wel de meest effectieve eerste stap om de softwareontwikkeling binnen een bedrijf of organisatie te veranderen in software dat over veiligere code beschikt.

# Owasp top 10 2021 binnen mijn applicatie[[3]](#footnote-3)

## Broken Access Control

94% van programma’s zijn getest op een vorm van Broken Access Control. De 34 Common Weakness Enumerations (CWE's) die aan Broken Access Control zijn gekoppeld, bleken zeer vaak van toepassing te zijn.

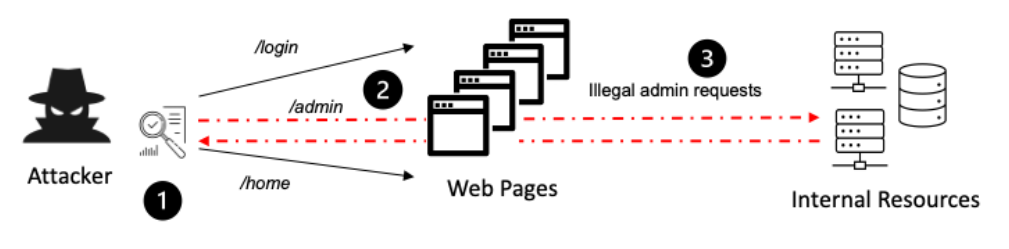
Afbeelding met tekst

Automatisch gegenereerde beschrijvingIn mijn persoonlijke project maak ik gebruik van Auth0 voor user management. Auth0 maakt standaard gebruik van ‘Click Validation’. Dit houd in dat bij het naderen van een nieuwe URL, auth0 checkt over de gebruiker is ingelogd. Zo kan Broken Access Control worden voorkomen.   
Daarnaast maak ik binnen mijn applicatie gebruik van authorisatie op de gateway en in de controller waar nodig, zo kan alleen een gebruiker die is ingelogd gebruik maken van deze controller.

Door het toevoegen van RateLimitOptions op de Gateway in Ocelot, kan er een maximum aantal calls worden toegelaten van een persoon. Zo kan er geen URL scan worden uitgevoerd.

Hoe dit scenario kan plaatsvinden (nummers gelinkt aan afbeelding):[[4]](#footnote-4)

1. Een aanvaller (hacker) gebruikt een tool om verschillende paden op een pagina te proberen. Bijvoorbeeld: */login, /admin, /home*
2. De aanvaller vind een beschikbare pagine, bijvoorbeeld *https://example.com/admin*
3. De aanvaller is op de admin pagina en kan hier verschillende, niet toegestane, commando’s uitvoeren.



## Cryptographic Failures

Eerder stond dit bekend als Sensitive Data Exposure, wat eerder een symptoom was dan een oorzaak. De hernieuwde aandacht gaat hier uit naar storingen in verband met cryptografie, die vaak leiden tot blootstelling van gevoelige gegevens of het in gevaar brengen van het systeem.

Cryptographic Failures kunnen steeds makkelijker worden voorkomen. Zo maakt bijna iedere applicatie tegenwoordig gebruik van SSL, om ‘data-in-transit’ te beschermen.[[5]](#footnote-5)   
Mijn applicatie runt in kubernetes, waarbij ik gebruik maak van NGINX, door mijn applicatie hierin te draaien. NGINX heeft standaard instellingen om cryptographic failures te verhelpen.

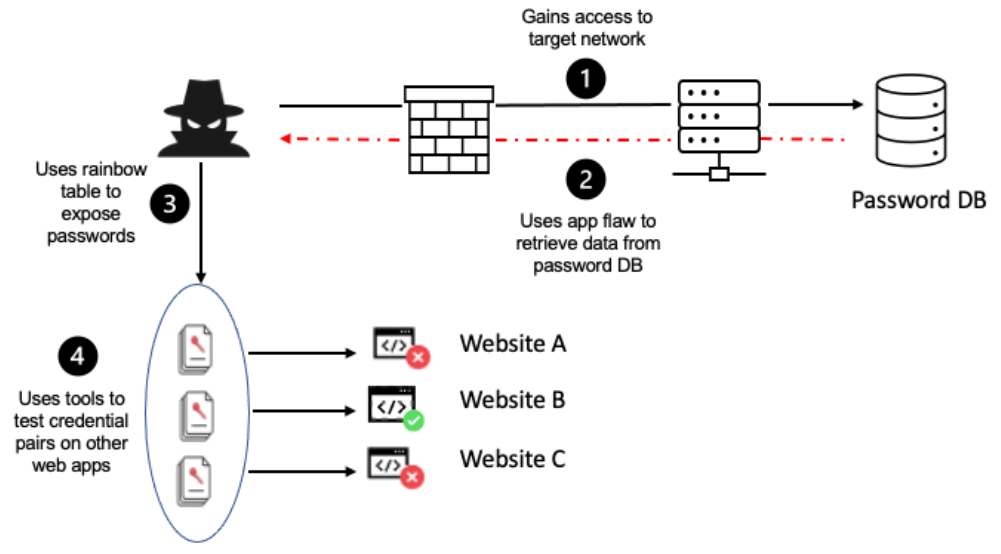
Afbeelding met tekst

Automatisch gegenereerde beschrijving

Auth0 maakt gebruik van HMAC-SHA256[[6]](#footnote-6). Hierbij worden de wachtwoorden van een gebruiker beschermd door gebruik te maken van een algoritme dat alleen van toepassing is als de ‘key’ bekend is. Ook zijn de hashes ‘salted’,[[7]](#footnote-7) waardoor er willekeurige data wordt toegevoegd aan de hash.

Hoe dit scenario kan plaatsvinden (nummers gelinkt aan afbeelding):

1. De aanvaller (hacker) krijgt toegang tot het netwerk.
2. Door een fout binnen de applicatie, krijgt de aanvaller toegang tot de wachtwoorden binnen de applicatie.
3. Omdat de database geen gebruik maakt van salted hashes, kan de aanvaller de eerder verkregen wachtwoorden door middel van een Rainbow table gebruiken om de applicatie.
4. Door het gebruiken van simpele tools kan de aanvaller de wachtwoorden testen.



## Injection

94% van de toepassingen werd getest op een of andere vorm van injectie, en de 33 CWE’s van deze categorie, komen het op een na meest voor in toepassingen. Cross-site Scripting maakt in de 2021 versie deel uit van deze categorie.

Binnen mijn applicatie maak ik gebruik van Entity Framework Linq-to-SQL, hierdoor worden alle gegevens doorgegeven aan de database via SQL parameters. Dus, hoewel de SQL query dynamisch wordt samengesteld, worden de waardes aan de serverzijde ingevuld, met parameters die bescherming bieden tegen de meest voorkomende oorzaak van SQL injectie aanvallen.[[8]](#footnote-8)[[9]](#footnote-9)

Hoe dit scenario kan plaatsvinden (nummers gelinkt aan afbeelding):[[10]](#footnote-10)

1. De aanvaller (hacker) stuurt een injectie op naar de web server. Dit kan bijvoorbeeld een SQL-Injection zijn zoals [*https://example.com/account?id='1'='1*](https://example.com/account?id='1'='1)
2. De server voert de injectie uit. Waarbij hij een query uitvoert met de vraag om gebruikers te selecteren, met alle gebruikers.
3. De server ontvangt de opgevraagde data en stuurt deze terug naar de aanvaller.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

## Insecure Design

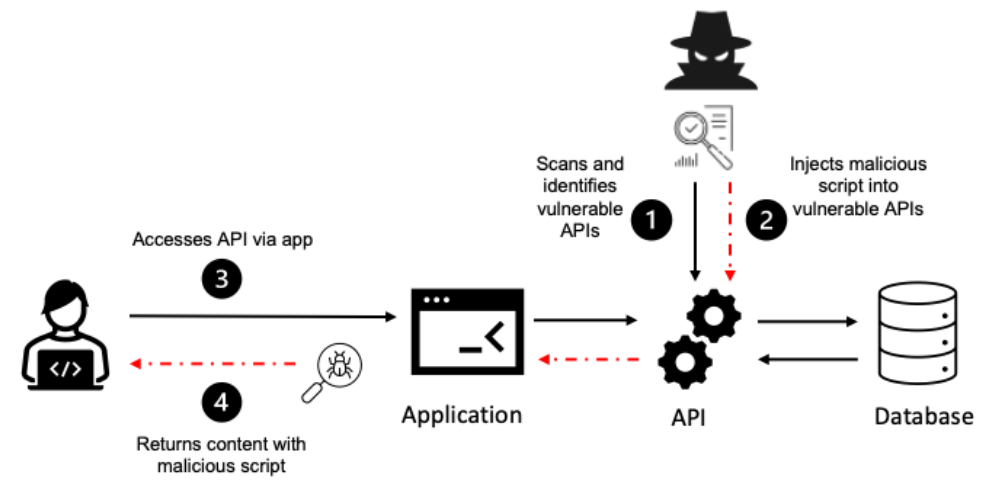
Afbeelding met tekst

Automatisch gegenereerde beschrijvingInsecure Design legt de nadruk op risico's in verband met ontwerpfouten. Om dit probleem te voorkomen zal er moeten worden gewerkt met veiligere ontwerpen.[[11]](#footnote-11)

Door het vooraf bepalen van het model van mijn applicatie kan dit probleem niet worden verholpen, maar wel verminderd worden.   
Bij het filteren van de inputs wordt deze kwetsbaarheid verminderd.  
Door het bepalen en uitvoeren van testen binnen de applicatie kunnen kritieke paden binnen de applicatie worden getest. Deze testen worden automatisch uitgevoerd door middel van de CI/CD pipeline. [[12]](#footnote-12)

Hoe dit scenario kan plaatsvinden (nummers gelinkt aan afbeelding):[[13]](#footnote-13)

1. De aanvaller (hacker) scant voor kwetsbaarheden binnen de applicatie.
2. Na het vinden van kwetsbaarheden kan de aanvaller gevaarlijke script naar de API sturen.
3. Een gebruiker wilt de applicatie benaderen, zoals hij/zij dit normaal gesproken doet.
4. Door het injecteren van de gevaarlijke code in stap twee, kan de aanvaller scripts naar de gebruiker sturen, aangezien hij/zij in stap drie, de applicatie benaderde.



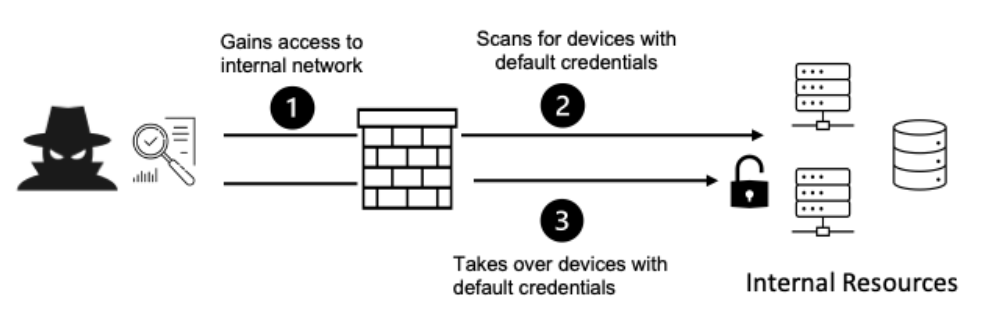
## Security Misconfiguration

90% van de toepassingen werd getest op een of andere vorm van een foute configuratie. Nu er meer verschuivingen zijn naar zeer configureerbare software, is het niet raar dat deze categorie omhoogschuift. De vroegere categorie voor XML External Entities (XXE) maakt nu deel uit van deze categorie.

Dit is een categorie die moeilijk te verhelpen is binnen een applicatie. Ik heb hiervoor zoveel mogelijk gedaan om dit te voorkomen. [[14]](#footnote-14)  
Ik heb geen admin panel op mijn applicatie, hiervoor hoef ik mij geen zorgen te maken.  
Door het gebruik van auth0 heb ik een standaard password policy neergezet, mensen kunnen dus geen standaard, en daarmee makkelijk ‘hackbaar’ wachtwoord hebben.  
Debuggen staat uit bij het runnen van de applicatie.  
Zoals eerder benoemd maak ik gebruik van autorisatie op mijn applicatie. Zo kunnen gebruikers alleen gebruik maken van een controller wanneer dit is toegestaan.   
Als laatste worden er door middel van de CI/CD pipeline automatische tests uitgevoerd om risico’s binnen de applicatie te ontdekken, en later verhelpen. [[15]](#footnote-15)

Hoe dit scenario kan plaatsvinden (nummers gelinkt aan afbeelding):[[16]](#footnote-16)

1. Een aanvaller (hacker) krijgt toegang tot een netwerk.
2. De aanvaller scant voor apparaten en accounts met standaard wachtwoorden.
3. De aanvaller neemt de accounts en apparaten over, waarmee hij zijn doel heeft bereikt.



## Vulnerable and Outdated Components

Dit is een bekend probleem dat moeilijk getest kan worde. Het is de enige categorie waarvoor geen Common Vulnerability and Exposures (CVE's) aan de opgenomen CWE's zijn gekoppeld, daardoor is er een standaard score voor exploit en impact van 5,0 in de scores is verwerkt.

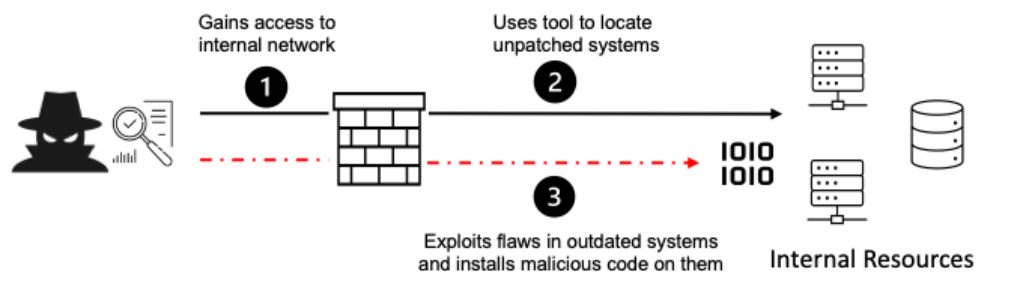
Binnen mijn applicatie heb ik geprobeerd om overal de nieuwste versie van een library te gebruiken. Soms is het echter niet altijd beter om deze versie te gebruiken, omdat nieuw uitgebrachte libraries vaak nog kwetsbaarheden bevatten.   
Voor de veiligheid van de applicatie geef ik ook alleen maar toegang aan functies die voor de applicatie van belang zijn, ik gebruik dus vaak niet de gehele library.   
Ook kijk ik bij het gebruik van de library, de bron, libraries die door mensen zelf zijn gemaakt kunnen ook kwetsbaarheden bevatten, waarmee ongewenst toegang tot de applicatie makkelijker wordt. [[17]](#footnote-17)[[18]](#footnote-18)

Afbeelding met tekst

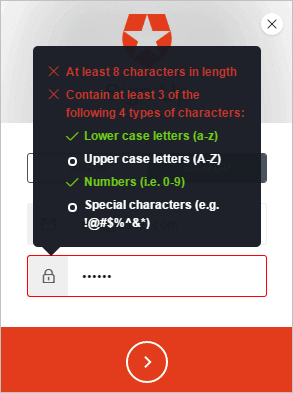
Automatisch gegenereerde beschrijving

Hoe dit scenario kan plaatsvinden (nummers gelinkt aan afbeelding):[[19]](#footnote-19)

1. De aanvaller (hacker) krijgt toegang tot het netwerk.
2. De aanvaller scant voor systemen die niet zijn geüpdatet.
3. De aanvaller gebruikt de kwetsbaarheden binnen de niet geüpdatet systemen, en installeert gevaarlijke code op het netwerk.



## Identification and Authentication Failures

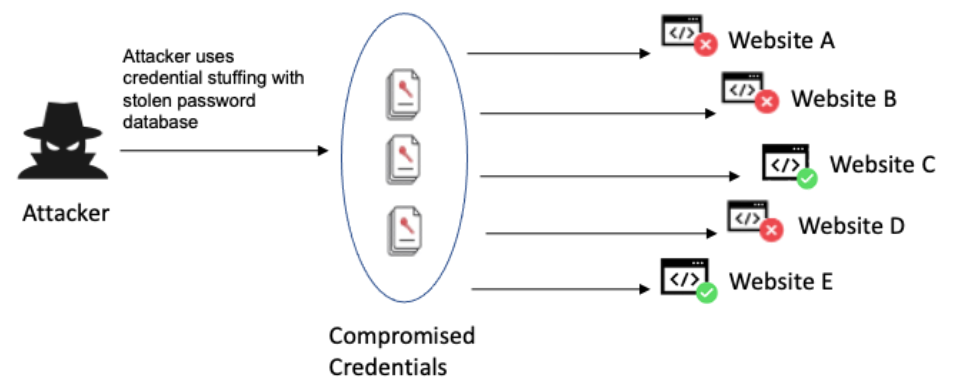
Bevat CWE's die meer te maken hebben met identificatiefouten. Deze categorie maakt nog steeds integraal deel uit van de Top 10, maar de toegenomen beschikbaarheid van gestandaardiseerde kaders lijkt te helpen.

Zoals eerder benoemd, maak ik voor user management gebruik van Auth0. Auth0 maakt gebruik van HMAC-SHA256[[20]](#footnote-20). Hierbij worden de wachtwoorden van een gebruiker beschermd door gebruik te maken van een algoritme dat alleen van toepassing is als de ‘key’ bekend is.

Ook heb ik eerder benoemd dat ik gebruik maak van een password policy binnen Auth0, zo moet het wachtwoord van een gebruiker aan bepaalde eisen voldoen, om de complexiteit van het wachtwoord te verhogen.[[21]](#footnote-21)[[22]](#footnote-22)

Als laatste, maakt Auth0 gebruik van tokens om de sessie van een gebruiker te bepalen. Deze token is maar voor een bepaalde tijd beschikbaar, zodat de gebruiker niet voor altijd ingelogd blijft.

Hoe dit scenario kan plaatsvinden:[[23]](#footnote-23)  
Een hacker heeft standaard wachtwoorden die hij uit kan proberen op websites. Door het niet goed instellen van een password policy, kan hij de toegang krijgen tot accounts op website C en E.



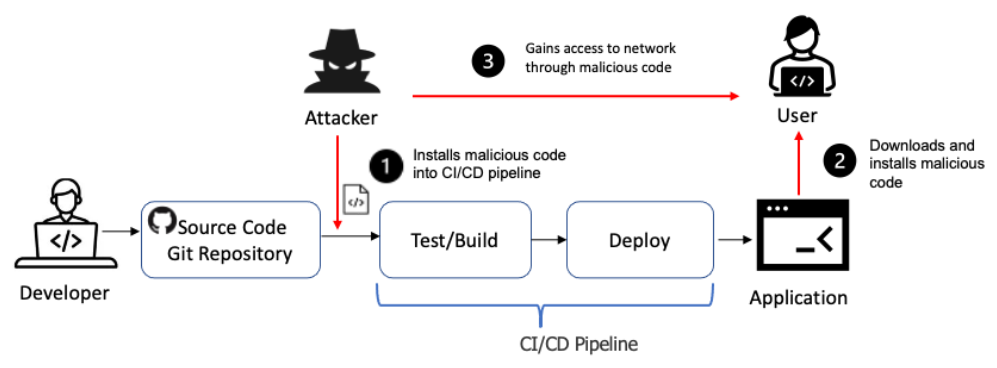
## Software and Data Integrity Failures

Hierbij ligt de nadruk op het maken van aannames met betrekking tot software-updates, kritieke gegevens en CI/CD-pijplijnen zonder de integriteit te verifiëren.

Voor het voorkomen van deze kwetsbaarheid, link ik deze aan [nummer 6 uit de OWASP Top 10](#_Vulnerable_and_Outdated).

Hoe dit scenario kan plaatsvinden (nummers gelinkt aan afbeelding):[[24]](#footnote-24)

1. De aanvaller (hacker) vind de CI/CD pipeline die niet goed beschermd is, en installeert hier gevaarlijke code op.
2. Een gebruiker/klant weet niet dat de gevaarlijke code is geïnstalleerd en download door de pipeline de code.
3. Door de gevaarlijke code kan de aanvaller toegang krijgen tot het netwerk van de klant.



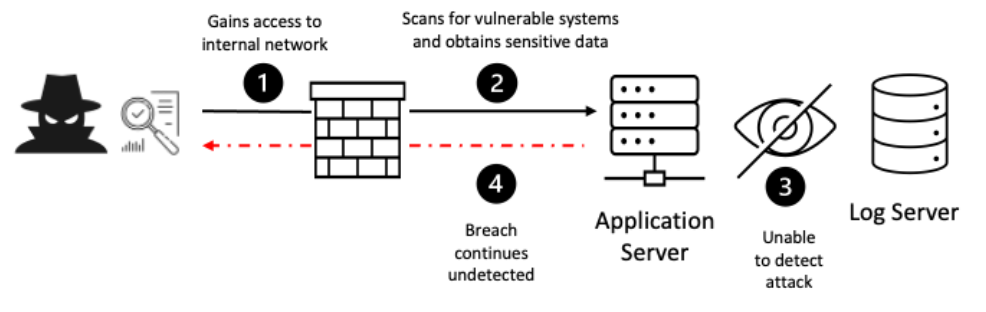
## Secure Logging and Monitoring Falures

Deze categorie is uitgebreid met meer soorten storingen, het is moeilijk om op te testen en is niet goed vertegenwoordigd in de CVE/CVSS-gegevens. Storingen in deze categorie kunnen echter een directe invloed hebben op de zichtbaarheid, de waarschuwing bij incidenten en het forensisch onderzoek.

Dit is een kwetsbaarheid die ik binnen mijn applicatie nog niet heb verholpen.

Hoe dit scenario kan plaatsvinden (nummers gelinkt aan afbeelding):[[25]](#footnote-25)

1. Een aanvaller (hacker) heeft/krijgt toegang binnen een netwerk.
2. De aanvaller gebruikt een tool voor het scannen van het netwerk en vind plekken waar gevoelige data wordt bewaard.
3. Omdat er geen logging en monitoring plaatsvind, kan de organisatie/bedrijf de aanval niet detecteren.
4. De scan is voltooid, de aanvaller heeft zijn doel bereikt.



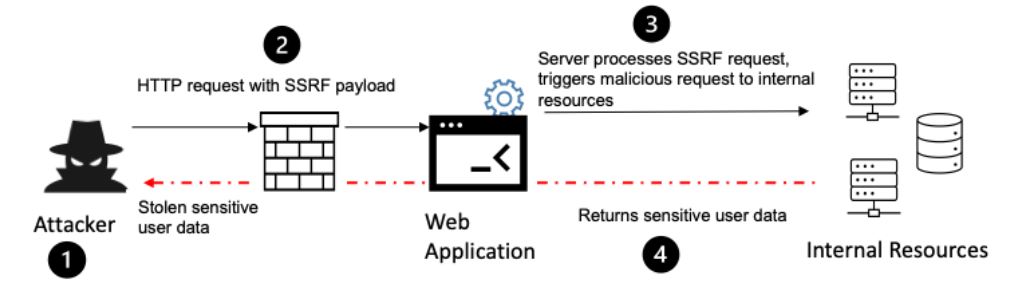
## Server-Side Request Forgery

De gegevens laten een relatief laag incidentiecijfer zien met een bovengemiddelde testing coverage, samen met bovengemiddelde waarderingen voor exploit- en impactpotentieel.

Dit is een kwetsbaarheid waar mijn applicatie geen last van zal hebben. Deze kwetsbaarheid zal namelijk plaatsvinden door verzoeken te plaatsen binnen de URL. Aangezien mijn applicatie dit niet doet, is dit niet van toepassing. [[26]](#footnote-26)

Hoe dit scenario kan plaatsvinden (nummers gelinkt aan afbeelding):[[27]](#footnote-27)

1. Een aanvaller (hacker) heeft een applicatie gevonden waar SSRF van toepassing kan zijn.
2. De aanvaller stuurt een HTTP request naar de applicatie:  
   Bijvoorbeeld: *GET /index.php?url=http://192.0.2.100/admin/*
3. De server krijgt de request binnen en roept de gevraagde data op.
4. De server stuurt de data terug naar de aanvaller, waarmee zijn doel bereikt is.



1. <https://www.whitehats.nl/owasp-top-10/#:~:text=De%20OWASP%20Top%2010%20is,zicht%20op%20deze%20complexe%20materie>. [↑](#footnote-ref-1)
2. <https://www.computest.nl/nl/knowledge-platform/hot-topics-explained/wat-is-de-owasp-top-10/> [↑](#footnote-ref-2)
3. <https://owasp.org/www-project-top-ten/#:~:text=The%20OWASP%20Top%2010%20is,step%20towards%20more%20secure%20coding>. [↑](#footnote-ref-3)
4. <https://support.f5.com/> [↑](#footnote-ref-4)
5. <https://support.f5.com/csp/article/K00174750> [↑](#footnote-ref-5)
6. <https://auth0.com/docs/get-started/applications/signing-algorithms> [↑](#footnote-ref-6)
7. <https://nl.wikipedia.org/wiki/Salt_(cryptografie)> [↑](#footnote-ref-7)
8. <https://www.devx.com/dotnet/Article/34653> [↑](#footnote-ref-8)
9. <https://www.lifars.com/2020/04/injection-attacks-explained/#:~:text=An%20injection%20attack%20is%20a,in%20the%20OWASP%20Top%2010>. [↑](#footnote-ref-9)
10. <https://support.f5.com/> [↑](#footnote-ref-10)
11. <https://owasp.org/Top10/A04_2021-Insecure_Design/> [↑](#footnote-ref-11)
12. <https://gupta-bless.medium.com/insecure-design-vulnerabilities-what-are-they-and-why-they-occurs-3a56ae080ca4> [↑](#footnote-ref-12)
13. <https://support.f5.com/> [↑](#footnote-ref-13)
14. <https://hdivsecurity.com/owasp-security-misconfiguration> [↑](#footnote-ref-14)
15. <https://www.manageengine.com/vulnerability-management/misconfiguration/#:~:text=Security%20misconfigurations%20are%20security%20controls,could%20lead%20to%20a%20misconfiguration>. [↑](#footnote-ref-15)
16. <https://support.f5.com/> [↑](#footnote-ref-16)
17. <https://support.f5.com/csp/article/K17045144> [↑](#footnote-ref-17)
18. <https://medium.com/@shivam_bathla/a06-2021-vulnerable-and-outdated-components-a5d96017049c> [↑](#footnote-ref-18)
19. <https://support.f5.com/> [↑](#footnote-ref-19)
20. <https://auth0.com/docs/get-started/applications/signing-algorithms> [↑](#footnote-ref-20)
21. <https://www.indusface.com/blog/broken-authentication-vulnerability/> [↑](#footnote-ref-21)
22. <https://medium.com/@shivam_bathla/a07-2021-identification-and-authentication-failures-b585c856eb24> [↑](#footnote-ref-22)
23. <https://support.f5.com/> [↑](#footnote-ref-23)
24. <https://support.f5.com/> [↑](#footnote-ref-24)
25. <https://support.f5.com/> [↑](#footnote-ref-25)
26. <https://portswigger.net/web-security/ssrf> [↑](#footnote-ref-26)
27. <https://support.f5.com/> [↑](#footnote-ref-27)