THOMAS MORE

APPLICATION SECURITY

Pentesting project

2023 - 2024

Inhoud

1. Inleiding

Voor het vak Application Security is er een webserver opgesteld met een bijhorende webapplicatie. Het doel hiervan is dat er op de applicatie een pentest wordt uitgevoerd waarin er gezocht wordt naar kwetsbaarheden in de applicatie. Na het testen van de webapplicatie is het de bedoeling deze te rapporteren in dit document.

Dit rapport bevat een overzicht van alle gevonden kwetsbaarheden en aanbevelingen gericht op het verbeteren van de beveiliging van de webapplicatie.

Deze pentest is belangrijk om een idee te hebben over hoe de applicatie aangevallen kan worden. Het leren ermee om te gaan en preventie hiervan is een prioriteit, dit zorgt voor een veiligere applicatie voor de gebruiker en de organisatie.

2. Samenvatting

De uitgevoerde security test op de ShopMore-webapplicatie heeft merkbare kwetsbaarheden blootgesteld die de beveiliging van het systeem in gevaar brengen. Het testproces omvatte meerdere soorten aanvallen gericht op het identificeren van mogelijke beveiligingsrisico's binnen de applicatie.

2.1. Bevindingen

Risico graad	Gevonden kwetsbaarheid	Gevonden
Critical	Ongeauthoriseerde toegang tot admin pagina	1
High	Injecteren van systeem commando’s	1
Medium	Gebruik van HTTP protocol voor versturen van gevoelige informatie	1
Low	/	/

Er zijn een aantal bevindingen die betrekking hebben tot de webapplicatie, waaronder twee kritieke kwetsbaarheden. Een daarvan is niet-geauthenticieerde toegang tot de administrator pagina, dit zorgt ervoor dat gebruikers zonder toestemming gebruik kunnen maken van admin rechten. De andere kwetsbaarheid is een command injection die ervoor zorgt dat bepaalde bestanden die te vinden zijn op de webserver getoont kunnen worden op de webpagina. Een voorbeeld van zo een bestand is de ‘passwd’ file die de gebruikers op het operating systeem bevat en informatie hierover.

De impact van deze kwetsbaarheden kan leiden tot toegang tot gevoelige gegevens van klanten en het blootstellen van de integriteit van de applicatie. Dit kan resulteren in reputatieschade en financiële verliezen voor de organisatie.

2.2. Aanbeveling

Een van de aanbevelingen is het strikt beveiligen van de administrator pagina zodat deze geen toegang bied aan andere zonder die rechten. Het versterken van de controle om te controleren of een gebruiker bepaalde zaken kan en mag doen, is zeer belangrijk.

Vermijd directe oproepen van systeemcommando's binnen de applicatie zodat deze niet misbruikt kunnen worden. Gebruik alternatieven zoals API's, waarbij er makkelijker toezicht gehouden kan worden in verband met veiligheid. Stel als het invoeren van deze commando's nodig is, valideer deze dan zorgvuldig.

In het algemeen is het aan te raden om regelmatig beveiligingsaudits uit te voeren zodat er vroegtijdige vastellingen kunnen gemaakt worden in geval van kwetsbaarheden, waardoor de impact hiervan verlaagt kan worden. Het gebruiken van makkelijk te raden gebruikersnamen en wachtwoorden is sterk afgeraden omdat dit gebruikers en eventuele administrators in gevaar kan brengen, pas zeker een regelgeving toe zodat het verplicht wordt deze sterker te maken.

3. Scoping

Het gebied van de test bevindt zich op de webapplicatie: ShopMore, en de bijbehorende webpagina's, inclusief de Home, Privacy, Products, Categories, Contact en Login-pagina. De testperiode van de applicatie loopt van 12 Oktober tot en met 17 December 2023. De testomgeving waarin de pentest wordt uitgevoerd is een fictieve omgeving om geen verstoringen te veroorzaken op echte applicaties.

3.1. Soorten kwetsbaarheden

Tijdens de pentest is er getest op bepaalde aspecten binnen de cybersecurity, hieronder meer info over deze onderdelen.

3.1.1. SQL Injection

Bij een SQL-injectie kan er SQL-queries wordt geïnjecteerd via invoervelden op een website. Hierdoor is het mogelijk om de database-commando's te manipuleren en toegang te krijgen tot de database van de website. Dit kan resulteren in het stelen van gevoelige gegevens, manipulatie van gegevens in de database en zelfs controle over de website en de achterliggende systemen verkrijgen.

3.1.2. Cross-site-scripting (XSS)

XSS is een manier waarbij schadelijke code geinjecteren kan worden via invoervelden op een website. Zo is het mogelijk om uitvoerbare code te injecteren die vervolgens wordt uitgevoerd in de browsers van andere gebruikers die de pagina bezoeken. Hierdoor kan er gevoelige gegevens en sessiecookies gestolen worden, wat de veiligheid van de gebruikers en de website in gevaar brengt.

3.1.3. Command Injection

Met deze manier is het mogelijk om commando's te injecteren, met andere woorden de mogelijkheid om systeemcommando's uit te voeren op de host waar de applicatie draait, in meeste gevallen is dit een server. Hierdoor is er een risico dat gevoelige gegevens, systemen of andere handelingen gemanipuleerd kunnen worden, wat niet veilig is

3.1.4. Path Traversal

Een kwetsbaarheid waarbij er gemanipuleerd kan worden via een onveilige invoer, zoals een pad naar bestanden buiten de mapstructuur van de website. Hierdoor is het mogelijk om ongeautoriseerde toegang te krijgen tot bestanden of directory's, wat de vertrouwelijkheid en integriteit van gegevens in gevaar kan brengen.

3.1.5. Information Disclosure

Dit gaat over het lekken van informatie, dit kan heel gevoelige data zijn of eerder algemene data. Onbedoeld informatie openstellen aan gebruikers is natuurlijk nooit aangeraden, dit kan data in gevaar brengen. Het is dus belangrijk dat dit nagekeken wordt en met zekerheid gezegt kan worden dat het beveiligt is.

4. Oplijsting van de vondsten

De volgende sectie bevat enkele kwetsbaarheden die gevonden zijn tijdens het testen, deze zijn in volgorde van wanneer ze gevonden zijn. Hieronder staat er een classificatie die laat zien op welke basis deze kwetsbaarheden worden geclassificeerd. Bij elke kwetsbaarheid staat er de prioriteit nog bij, tussen ronde haakjes en een CVSS beoordeling die meer zegt over hoe ergnstig deze is.

CVSS staat voor "Common Vulnerability Scoring System". Het is een systeem dat wordt gebruikt om de ernst van beveiligingskwetsbaarheden te beoordelen. Het biedt een numerieke score op basis van verschillende metrieken, zoals de impact, de mate van blootstelling aan een kwetsbaarheid en de complexiteit van de aanval. De score helpt bij het objectief beoordelen van beveiligingskwetsbaarheden.

Classificatie van de risico's

Elke gevonden kwetsbaarheid wordt in een bepaald categorie geplaats naargelang het risico en de impact hiervan. De categorieën zijn als volgt:

Critical
Deze kwetsbaarheden zouden direct aangepakt moeten worden, omdat ze een dringend gevaar zijn voor de webapplicatie en/of server. Het misbruiken van deze kwetsbaarheden is eenvoudig en kan gedaan worden zonder kennis hierover.

High
Kwetsbaarheden van deze graad zouden ook direct moeten worden aangepast omdat deze een bedreiging vormen. Het exploiteren van deze kwetsbaarheden zijn ingewikkelder en vereisen in sommige gevallen kennis hierover.

Medium
Het aanpakken van deze risico's heeft minder prioriteit, omdat het moeilijk te exploiteren is en dit in sommige gevallen interactie vereist van bevoegden.

Low
Deze kwetsbaarheden moeten zeker genoteerd worden, maar hebben een lagere prioriteit. Ze hebben weinig impact op de applicatie of zijn erg moeilijk te misbruiken.

Informational
Deze worden gemeld voor informatie te geven en poseren geen dreiging.

4.1. HTTP protocol (Medium)

CVSS:3.1/AV:A/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N

Gericht op alle pagina's: *10.0.2.100*
Vooral gericht op pagina: 10.0.2.100/Login

Beschrijving

HTTP is een protocol dat wordt gebruikt voor communicatie tussen een webserver en de browser van de gebruiker. Het maakt het zenden van gegevens mogelijk, zoals HTML-bestanden, afbeeldingen en andere bronnen die een webpagina vormen.

Om te zien of een website gebruikt maakt van dit protocol wordt er gekeken naar de zoekbalk. Als de URL begint met "**http://**" dan gebruikt de website HTTP. Als het beveiligd is, zal de URL beginnen met "**https://**" waarbij de letter 's' staat voor 'secure'. Dit betekent dat de website het HTTPS-protocol gebruikt, een veiligere versie van HTTP die zorgt voor een beter beveiligde communicatie tussen de gebruiker en de webserver.

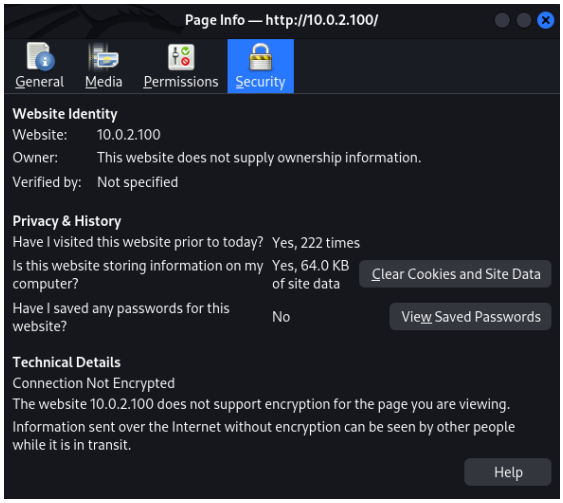
Risico

Omdat de connectie via het protocol HTTP is, wilt dit zeggen dat de verbinding tussen de gebruiker en de server niet beveiligd is. Doordat deze verbinding niet ge-encrypteerd is zal al het verkeer in **tekstvorm** staan. In geval dat een gebruiker wilt inloggen met de gebruikersnaam en wachtwoord zal dus ook deze in tekstvorm over de verbinding gaan.

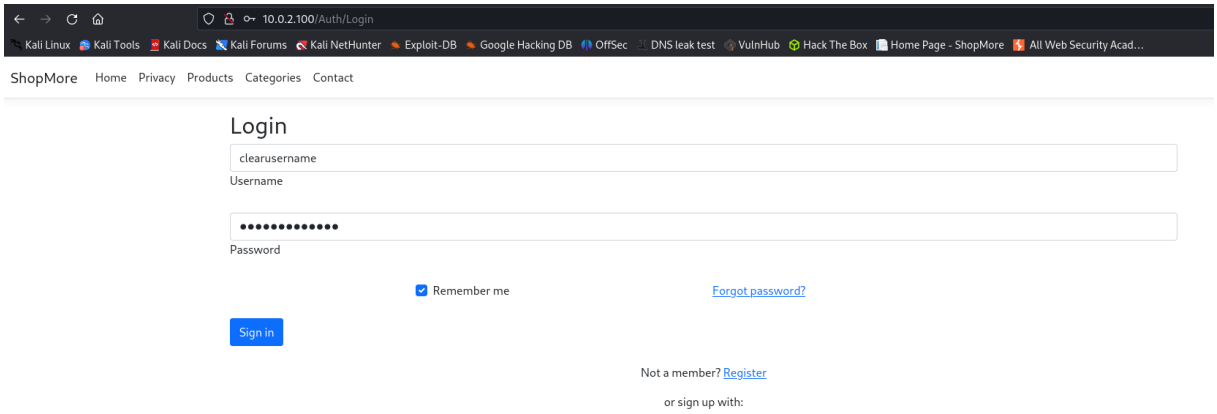
Wanneer deze gebruikersnamen en wachtwoorden via een HTTP-verbinding worden verzonden, kunnen hackers deze informatie onderscheppen met behulp van technieken zoals packet sniffing. Dit geeft hun de kans om inloggegevens te achterhalen en toegang te krijgen tot accounts, wat kan leiden tot identiteitsdiefstal, privacyschendingen en andere zware gevolgen.

Bewijs

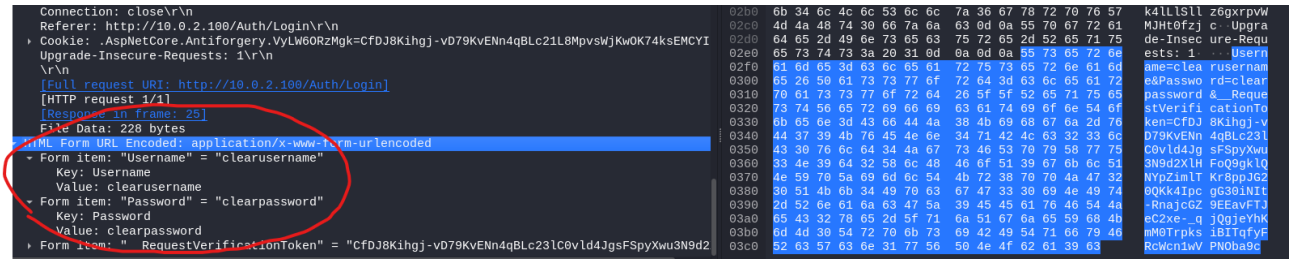
Hieronder is te zien dat de website gebruik maakt van HTTP. Vanboven op de foto staat de link en hier is te zien dat deze start met http://. Vanonder bij Technical Details geeft de browser, in dit geval Firefox, aan dat de verbinding met de server niet ge-encrypteerd is.



Dit zijn de gegevens die we in dit voorbeeld gebruiken, de gebruikersnaam is “clearusername” en het paswoord is “clearpassword”. Dit zijn fictieve gegevens die enkel gebruikt zijn voor het testen en laten zien.



Na het insturen van deze inloggegevens kan er gebruik worden gemaakt van packetsniffers. Op de foto wordt Wireshark gebruikt om het pakket met de inloggegevens te bekijken. De gegevens staan in tekst zichtbaar, dus op deze manier wordt het pakket verstuurd. Natuurlijk is dit niet veilig omdat een potentiele aanvaller hetzelfde kan zien.



Aanbeveling

Het is essentieel om HTTPS te gebruiken, vooral bij het verwerken van gevoelige informatie. HTTPS versleutelt deze informatie tijdens de verzending, waardoor de communicatie tussen de browser en de webserver veiliger wordt en de kans op onderschepping van gevoelige gegevens aanzienlijk wordt verminderd.

Hieronder een referencie over hoe het probleem best wordt aangepakt.
https://jcarpizo.github.io/owasp-info/cheatsheets/DotNet_Security_Cheat_Sheet.html#asp-net-web-forms-guidance

4.2. Admin Vertical Privilege Escalation (Critical)

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

Gericht op pagina: *10.0.2.100/Admin*

Beschrijving

Een niet-geauthorizeerde gebruiker kan toegang krijgen tot de /admin pagina door het aanpassen van de URL in de zoekbalk. Op deze manier kan een gebruiker zonder authenticatie of authorizatie gebruik maken van admin rechten.

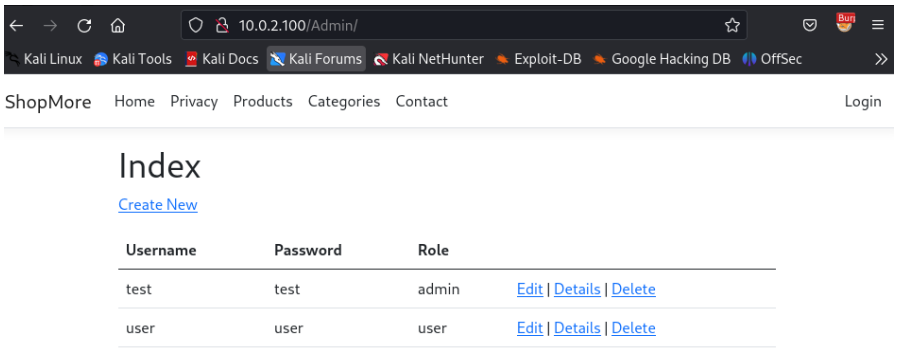
Er is geen account nodig om deze kwetsbaarheid te kunnen gebruiken, dit zorgt ervoor dat het een ernstige kwetsbaarheid is.

Risico

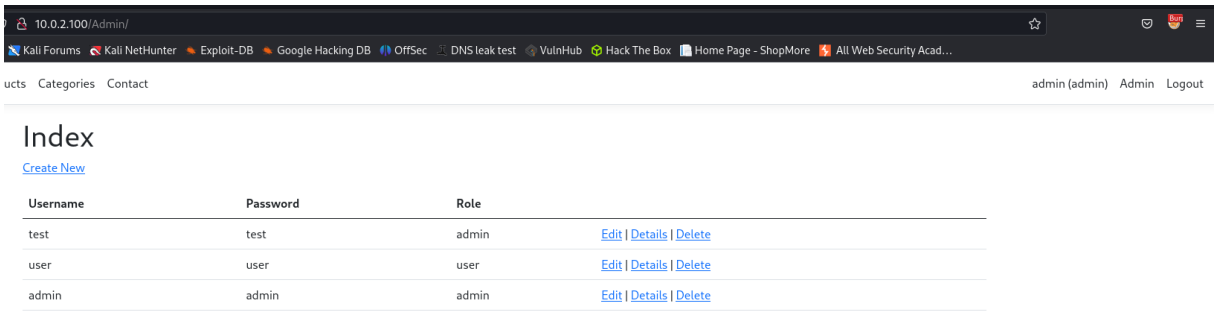
Doordat de admin pagina gebruikt kan worden door praktisch iedereen die op de site komt, is dit een ernstig risico. Op de admin pagina staan alle accounts met hun gebruikersnaam en wachtwoord. De opties om deze aan te passen, verder te bekijken en te verwijderen staan ook op deze pagina. Dit maakt het mogelijk om gevoelige gegevens te lezen, manipuleren of te verwijderen.

Bewijs

Door de url **10.0.2.100/Admin** in te geven of **/Admin** achter de url te zetten, kan een gebruiker zonder in te loggen op de admin pagina geraken. Hieronder is te zien hoe er op de admin pagina is geraakt zonder in te loggen.



Het valt dus op dat rechts nog “Login” staat, wat er op duid dat er niet is ingelogd. Hieronder is te zien hoe het zou moeten zijn.



Nu staat er rechts dat admin is ingelogd en tussen de haken staat welke “functie” deze gebruiker heeft, in dit geval admin.

Aanbeveling

Het is noodzakelijk om ervoor te zorgen dat alleen geauthenticeerde administrators toegang hebben tot deze pagina. Het is **ten sterkste aanbevolen** dit zo snel mogelijk te doen.

Voor extra info betreft dit probleem is het om te refereren naar de link: https://jcarpizo.github.io/owasp-info/cheatsheets/DotNet_Security_Cheat_Sheet.html#a7-missing-function-level-access-control

4.3. Command Injection (High)

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N

Gericht op pagina: *10.0.2.100/Export*

Beschrijving

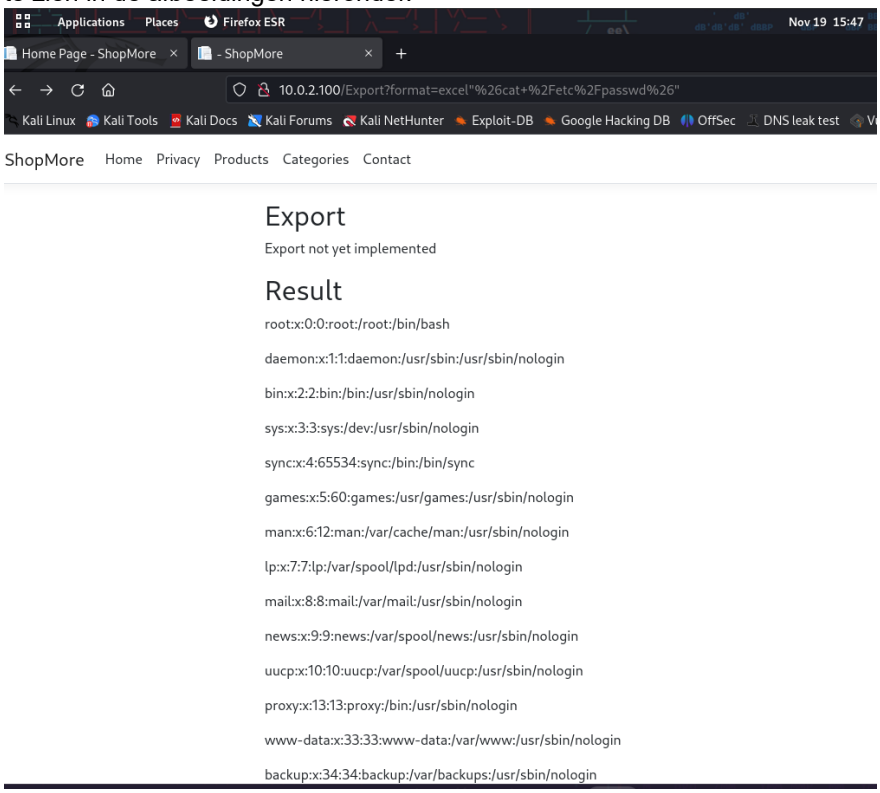
Via de zoekbalk is het mogelijk om de URL aan te passen op een manier dat deze bestanden laat zien op de **/Export** pagina. Op deze manier is het mogelijk om commands uit te voeren op de server. De server maakt gebruik van een Linux operating system waardoor bepaalde commands zoals: ‘cat’ mogelijk is.

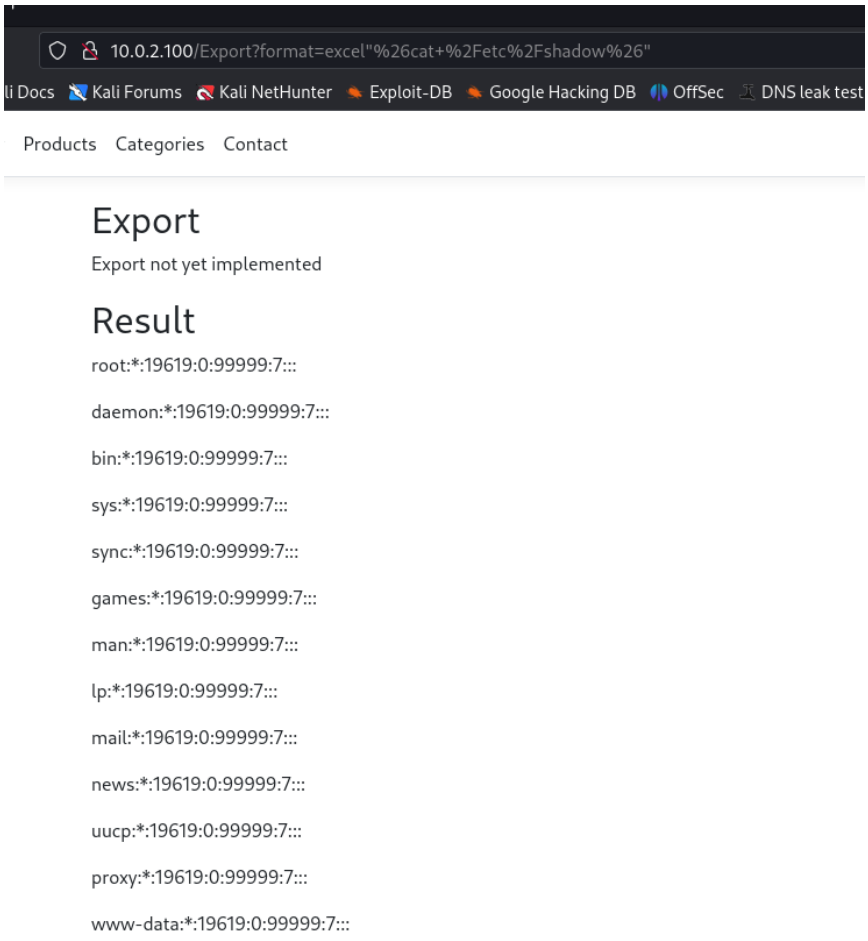
Risico

Door deze kwetsbaarheid in de applicatie kunnen er op afstand systeemcommando’s uitgevoerd worden op de server van de applicatie, dit is uiteraard niet veilig voor de server en de applicatie. Omdat het in dit geval mogelijk is om toegang te krijgen tot bestanden die te vinden zijn op de server is dit een kritiek risico. De bestanden in kwestie bevatten gevoelige info van de server, deze kunnen misbruikt worden door dreigingen. De impact van command injections zijn vrij ernstig omdat het kan resulteren in verlies van data en toegang tot de applicatie en/of server.

Bewijs

Door het ingeven van http://10.0.2.100/Export?format=excel%22%26cat+%2Fetc%2Fpasswd%26%22 in de zoekbalk, is de Export pagina te zien, deze staat niet zichtbaar op de website zelf maar is wel aanwezig. Door deze URL in te geven zijn er files, zoals in dit geval **passwd** en **shadow** zichtbaar op de pagina, zoals te zien in de afbeeldingen hieronder.





Aanbeveling

De meest efficiënte manier om command injections te voorkomen, is om nooit deze te roepen vanuit code op applicatieniveau. In bijna alle gevallen zijn er verschillende manieren om de vereiste functionaliteit te implementeren met behulp van API's, wat veilig gemaakt kan worden. Als je commands moet aanroepen met door gebruikers geleverde invoer, dan is het beter om sterke invoervalidatie uit te voeren, zoals: validatie of de invoer een getal is, alleen tekens bevat, enzovoort.

De twee linken hieronder kunnen een beter beeld geven over dit onderwerp en inzicht geven over hoe dit het best te voorkomen is:
https://owasp.org/www-community/attacks/Command_Injection
https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html

CONTACT

Michael Vervoort | Student
r0844989@thomasmore.be

VOLG ONS

www.thomasmore.be
fb.com/ThomasMoreBE
#WeAreMore

THOMAS MORE