[redacted]

2019 Security Assessment Report Prepared For

Report Issued: 23 November 2019

Sensitive: The information in this document is strictly confidential and is intended for DinoBank

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The [redacted] penetration testing team performed a security assessment on the internal corporate and customer network of DinoBank on 22 and 23 November 2019. The penetration test simulated an attack from an outside threat attempting to exploit vulnerabilities within the DinoBank network. The purpose of the assessment was to identify these vulnerabilities and provide possible mitigations. There were a total of 16 vulnerabilities discovered, consisting of 2 critical vulnerabilities, 1 high-risk vulnerability, 7 medium-risk vulnerabilities, and 6 low-risk vulnerabilities. These risk classifications are defined per Table 1.

To ensure data integrity, availability, and confidentiality, proposed threat mitigations should be promptly implemented because many of the vulnerabilities violate federal compliance regulations for financial institutions. If left uncorrected, these violations would undoubtedly lead to significant financial and legal repercussions as well as significant privacy breaches. Information on reproducing vulnerabilities are also provided to guide Dinobank's security and development teams in understanding and preventing these threats in the future.

Please note that this assessment does not, nor is it intended to, disclose every possible vulnerability that may be present on the system. Any changes made to the environment during testing may affect the results of the assessment. However, the team made a significant effort to prevent modifications to the bank's network and maintain it's availability as it was still being used as a production system.

RISK ASSESSMENT

Risk is composed of a combination of the likelihood of a detrimental event occurring and the impact of such an event. For DinoBank, any degradation of the integrity of the banking system or unauthorized access to private customer information would have catastrophic consequences. Improper maintenance of customer information would violate numerous federal compliance regulations for financial institutions, including the Gramm-Leach-Bliley Act (GLBA), the Payment Card Industry Data Security Standard (PCI DSS), General Data Protection Regulations (GDPR), and Title III of the US PATRIOT Act, among others. These violations would lead to significant financial and legal repercussions. For example, each violation of the GLBA, which specifies that financial institutions are obligated to protect the confidentiality of customers' private information, could lead to a fine of $100,000.

The likelihood of these events occurring is variable as they can be caused by both external and internal actors. While external threats are more common than internal threats, internal threats can easily lead to catastrophic consequences. For this reason, a healthy focus on security should be carefully cultivated within the company's culture, which includes extending security-oriented behavior beyond the end of this assessment. Also, both external and internal actors could exploit many existing vulnerabilities that include but are not limited to, logical issues in the core banking application, unencrypted personal information, unhashed passwords, usability issues, and potential malware.

These factors point to the undeniable truth that, if left unaddressed, these vulnerabilities have the potential to threaten DinoBank's reputation as a reliable financial institution, or even DinoBank's future as a company.

Table 1. Security level descriptions

Security Level	Description
Critical	Successful exploitation will result in a major disruption of business functionality. It is recommended that the vulnerability is remediated or mitigated immediately.
High	Successful exploitation will result in elevated privileges, significant data loss, or any other severe detriment to business functions.
Medium	Successful exploitation will result in the inadvertent disclosure or modification of sensitive information or the interference with user interactions or corporate auditing procedures.
Low	These findings have minimal impact on business functions and typically require local or physical system access.

Table 2. Summary of vulnerabilities

#	Risk	Observation	Impact to DinoBank	Service
1	C	Banking Logic Issue	Money can be created by any user with no interaction by DinoBank employees.	Core Banking Application (Site)
2	H	Unencrypted PII	Extremely sensitive information in large volumes. Extremely serious violation of banking regulations.	Core Banking Application (Database)
3	M	Plaintext Passwords	Customer passwords easily revealed should a breach occur.	Core Banking Application (Database)
4	C	Database Blank Password	Anyone can login remotely with no password to the consumer database.	Core Banking Application (Database)
5	L	Database to Shell Escalation	Users logged in to the database can escalate out due to misconfigured login controls.	Core Banking Application (Database)
6	L	Core API Running as root User	Misconfiguration in Core API service could allow a privileged shell on successful exploit.	Core Banking Application (API)
7	L	Developer Notes	Developer notes leftover in customer-facing site leaks infrastructure information.	Core Banking Application (Site)

8	L	Poor User Interface and information disclosure	Poor User Interface damages DinoBank's reputation and security	Core Banking Application (Site)
9	M	Unencrypted HTTP	Unencrypted communication between client and server renders DinoBank customers at-risk.	Core Banking Application (Site)
10	M	Reused Administrator Passwords	Windows Administrator password reused on 11 machines.	Active Directory Infrastructure
11	M	Excessive Domain Admins	High number of domain admins may allow many employees to modify the network.	Active Directory Infrastructure
12	M	Plaintext Passwords within Logs	Windows Administrator password present in logs on 5 machines, available in plaintext.	Active Directory Infrastructure
13	M	Anonymous FTP Login	FTP login with no password allows access to sensitive information inside corporate network.	Corporate Network, Corp Subnet (10.0.1.12, WSUS-01)
14	L	Cryptocurrency Miner	Cryptocurrency mining malware present on DinoBank wastes DinoBank resources and could present an additional attack vector for adversaries.	Corporate Network
15	M	Sensitive Information on Windows	Plaintext passwords stored on TLR* machines present large security risk.	Corporate Network (TLR-*)
16	L	Sensitive information in Wiki	Internal DinoBank wiki exposes default username schema and password for new users.	Corporate Network

TESTING METHODOLOGY

[redacted]testing methodology used to conduct the penetration test can be split into three phases - reconnaissance, target assessment, and execution of assessment. Reconnaissance is comprised of gathering information about DinoBank's network systems. Port scanning and other enumeration methods were used to aid in our collection of targetable information. After refining our target set, we began our target assessment, researching vulnerabilities and their potential effects on DinoBank services. We then executed our assessment in a manner that would not disrupt normal business operations. The following figure is a graphical representation of this methodology.

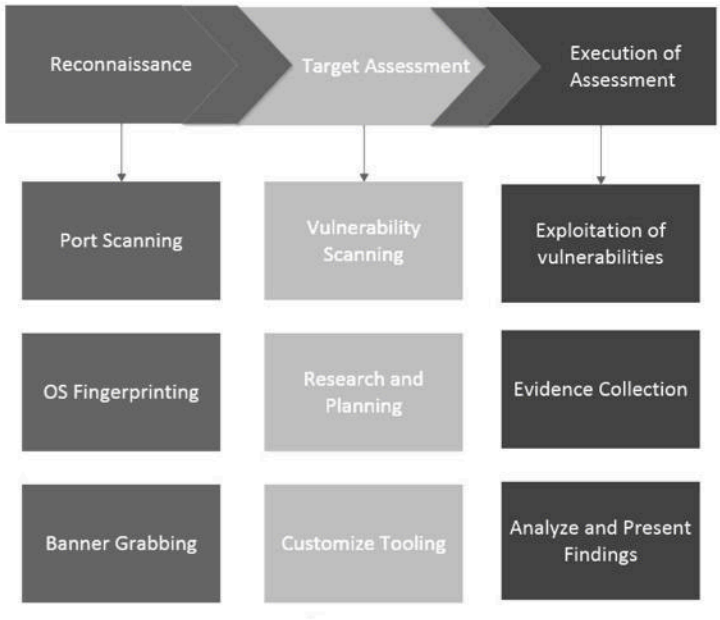


Figure 1. Testing methodology

The topology of the network was found using nmap and scanning the five given subnets, which were 10.0.1.0/24, 10.0.2.0/24, 10.0.10.0/24, 10.0.11.0/24, and 10.0.12.0/24. To gain more information besides which hosts were up, the following nmap scan was run overnight:

```
nmap 10.0.1.0/24 10.0.2.0/24 10.0.10.0/24 10.0.11.0/24 10.0.12.0/24 -T4 -A -p-
```

Using the information obtained by this scan, we created the following topology of DinoBank's network:

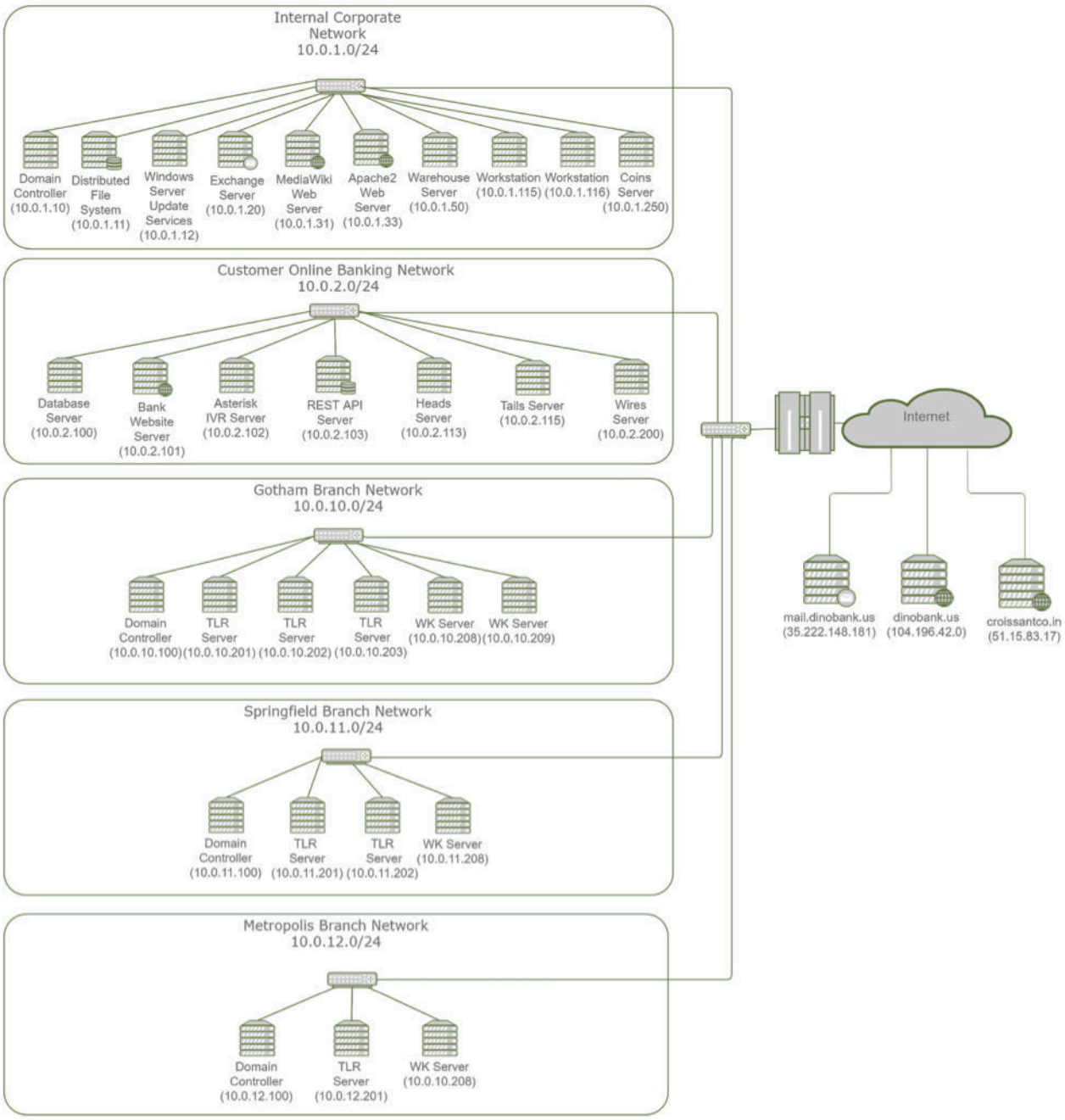


Figure 2. DinoBank network topology

Once the reconnaissance phase was completed, the target assessment and execution assessment phases were completed simultaneously. Each of the six penetration test team members were assigned to different hosts and services. Each team member would first research the host. With the information obtained, the team member would look into potential, non-intrusive, vulnerabilities and attempt to produce a proof of concept without causing significant modification to the host or service while maintaining its functionality.

FINDINGS

Core Banking Application

Banking Logic Issue

During the previous security assessment, logic issues in the core banking application, accessed at 10.0.2.101, allowed users to transfer negative amounts and create money by transferring to the same account. While negative amounts are no longer allowed on the backend, users could still transfer money from one account to the same account, and the money would be created. To reproduce, first check the initial account balance:

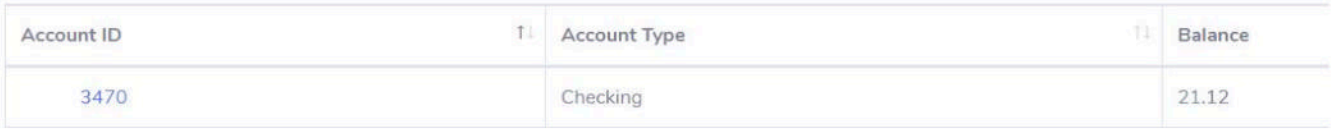


Figure 3. Banking logic issue POC, part 1

Next, transfer a positive amount from one account number to the same account number ($5 in this case):

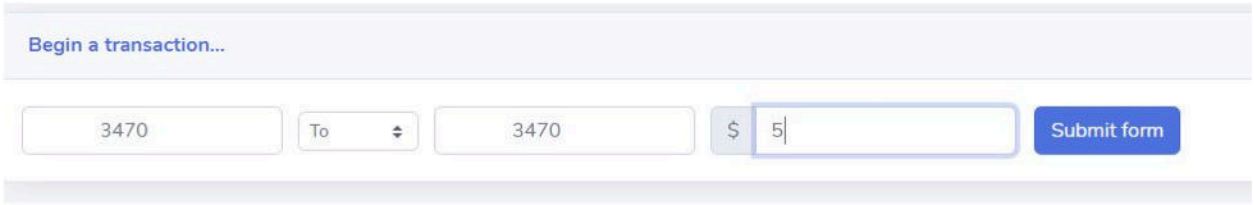


Figure 4. Banking logic issue POC, part 2

Finally, notice that the account balance has increased by $5:

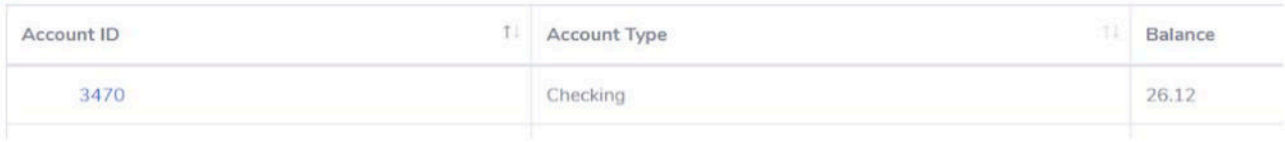


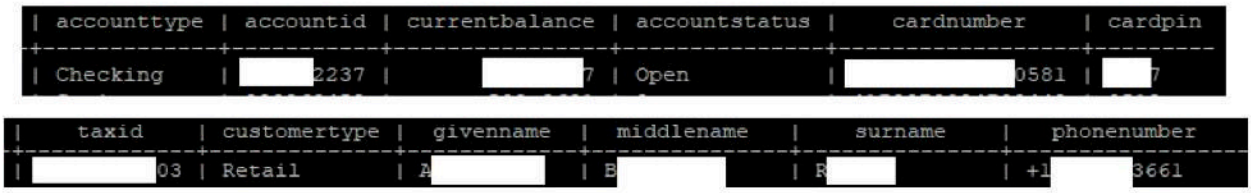
Figure 5. Banking logic issues POC, part 3

This vulnerability would allow a customer to create currency without withdrawing it from another account, leaving DinoBank or the customer liable to a fine of up to $1,000,000 and 30 years in prison under 18 U.S.C. § 1344 and critically damage DinoBank's reputation.

The backend code of the core banking application should include logic that disallows transfers from one account number to the same account number.

Unencrypted PII

This was also an issue identified in the first security assessment. The PostgreSQL database at 10.0.2.100 contained personally identifiable information for both employees and customers, including account numbers, social security numbers, and PIN. The data was unencrypted and stored in plaintext. Below is an example:



accounttype	accountid	currentbalance	accountstatus	cardnumber	cardpin
Checking	[redacted]2237	[redacted]7	Open	[redacted]0581	[redacted]7

taxid	customertype	givenname	middlename	surname	phonenumber
[redacted]03	Retail	A[redacted]	B[redacted]	R[redacted]	+1[redacted]3661

Figure 6. Example of plaintext SSN, account number, and PIN

In the event an attacker was able to gain access to the database, they would have immediate access to customer and employee PII. This information could be used for malicious purposes, which would reduce customer and employee confidence in DinoBank. Additionally, such a breach would result in a violation of the Gramm-Leach-Bliley Act (GLBA), which would result in a $100,000 fine for each violation.

To significantly reduce the chances of an attacker obtaining PII, the data should be encrypted using an algorithm such as AES. Thus, the attacker would also need to obtain the AES key to view the data, increasing the difficulty for the attacker to view customer and employee PII.

Plaintext Passwords

This was also an issue identified in the first security assessment. The PostgreSQL database at 10.0.2.100 contains passwords in plaintext for both customers and employees. Below is an example:

loginid	passwd
m[redacted]@gmail.com	[redacted]E4cP

Figure 7. Example of plaintext user password

Provided an attacker gained access to the database, they would also be able to discreetly login to user accounts using the plaintext passwords. If a user's account was hijacked, it would reduce their confidence in DinoBank's security. Since users may also reuse passwords on other services, an attacker may be able to take advantage of a breach to access a user's account on other websites. Since a password would also be considered PII, this would also be a violation of the GLBA, resulting in a fine of $100,000.

Passwords should be hashed in the database using a secure hashing algorithm, such as SHA-256 or bcrypt. Salting the hash would also improve its security.

Database Blank Password

Unauthorized access to the database was a vulnerability identified in the first security assessment. Controls were sufficiently implemented for the database user meant for the production web application. However the default `postgres` user on the PostgreSQL server (10.0.2.100) allowed for remote logins from any host with an empty password. The following command is used to connect to the database:

```
psql -h 10.0.2.100 -U postgres
```

This allows any user with network access to access all customer and employee data present on the database as a privileged user. A malicious actor would also be able to modify the information in the database, such as account balances.

To prevent unauthorized access, host filtering should be enabled to allow logins from only local connections, and the `postgres` user should have a secure, complex password set.

Database Shell Escalation

A misconfigured shell in `/etc/passwd` allows from escalation from the PostgreSQL shell to a bash shell as the `postgres` user. Outputting a valid SSH public key to `/var/lib/postgresql/.ssh/authorized_keys` allows for authorized SSH connections:

```
COPY cmd_exec FROM PROGRAM 'echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDUtbtAJfzGoCPXkxu6BPMRRl1jIBnAC8TE2grORRt43rTcZxr8wJ1IqjMyM7x09OztJySFCUvyAFSL72dbhPoirz3xdk6bkug8NtK/
QcP1L75OD4JqxASCcPwH5L2YYW+jgMm+AEen48iqXUH+/T5YkxCYweGVUTXvL0TLhtJ//1DSHZJGDwWRmN7Af1QubtWs+LnvvEymHKEXRaKqOYQ3iL5DY2qhY/
nnzAcLDuab/4yVr4z02F7o4v8/ho8vTzP1HlRjiNr1Wt6v14zpzGVC6k44dKrSUmYKIQIeWmyVuH2czKQVp3cWeQ/rWfxT6SFu3yNjpTYmrOI7MD9Sngyr" > /var/lib/postgresql/.ssh/
authorized_keys'; select * from cmd_exec; delete from cmd_exec;
```

Figure 8. Process to output to authorized_keys from PostgreSQL

Given this vulnerability, an attacker can easily use the bash environment to explore and enumerate local vulnerabilities on the local machine and use it as a pivoting foothold to further explore the network from the inside.

To prevent this issue, the `postgres` user should be configured to use a shell of `/usr/sbin/nologin`.

NodeJS Running as Root

The core banking application's internal API (https://10.0.2.100/core/v1/) is misconfigured in such a way that it would provide a remote attacker with a root local shell given a success exploit. The `root` user owns the API files and runs the NodeJS process:

```
nodejs:x:1001:1002::/data/web:/bin/bash
```

Figure 9. nodejs user present within /etc/passwd

```
postgres@core-01:/data/web/api/test$ ls -la /data/web
total 12
drwxr-xr-x 3 root root 4096 Nov 21 17:52 .
drwxr-xr-x 3 root root 4096 Nov 21 17:52 ..
drwxr-xr-x 8 root root 4096 Nov 21 17:52 api
postgres@core-01:/data/web/api/test$ ls -la /data/web/api
total 256
drwxr-xr-x   8 root root    4096 Nov 21 17:52 .
drwxr-xr-x   3 root root    4096 Nov 21 17:52 ..
drwxr-xr-x   2  501 staff   4096 Sep 11 17:36 bin
drwxr-xr-x   2  501 staff   4096 Nov 20 00:07 config
-rw-r--r--   1  501 staff   1574 Sep  5 23:41 db.js
drwxr-xr-x   2  501 staff   4096 Nov 18 15:59 docs
-rw-r--r--   1  501 staff    262 Nov 20 01:30 ecosystem.config.js
-rw-r--r--   1  501 staff   1889 Nov 18 23:51 index.js
drwxr-xr-x 357 root root   12288 Nov 21 17:52 node_modules
-rw-r--r--   1  501 staff    644 Nov 18 23:51 package.json
drwxr-xr-x   3  501 staff   4096 Sep  4 17:36 routes
drwxr-xr-x   2  501 staff   4096 Jul 23 19:20 test
-rw-r--r--   1  501 staff  74005 Nov 18 20:48 yarn-error.log
-rw-r--r--   1  501 staff 123477 Nov 18 23:51 yarn.lock
postgres@core-01:/data/web/api/test$
```

Figure 10. Current ownership of /data/web/

```
root      1537  0.3  1.2 941836 98952 ?        Ssl  13:05   1:38 node /data/web/api/index.js
```

Figure 11. root user hosting core API

Provided a malicious actor can gain remote code execution via the API, they can gain a more privileged shell, allowing them to access and modify more files and processes.

To correct this issue, `nodejs` user shell should be changed to `/usr/bin/nologin`. Additionally, the ownership of `/web/data/` should recursively be changed to `nodejs`. Finally, the core API process should be run using the `nodejs` user.

Developer Notes

Developers left hints of unimplemented and potentially vulnerable services within the source code of the webpages. For example, the following code was easily found on a client-facing web pages on https://my.dinobank.us/ by viewing a page's source code.

```
<!--    text: 'insert into customers (taxId, customerType, givenName, middleName, surName,
phoneNumber, emailAddr, streetAddr1, streetAddr2, cityName, stateCode, postalCode, companyName)
values ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13 ) returning customerId' -->
<div class="alert alert-danger" role="alert">error: invalid input syntax for type uuid: "a"</div>
```

```
<!--<div class="text-center">
  <a class="small" href="forgot-password.php">Forgot Password?</a>
</div>-->
```

Figure 12. Developer Notes found in HTML

This practice opens DinoBank services up to unnecessary risk, revealing SQL queries and unimplemented services allow an attacker to gain a better understanding of the backend of DinoBank's critical infrastructure as well as identify valuable services.

This issue can be remediated by removing irrelevant and potentially revealing comments from webpages.

Poor User Interface and Information Disclosure

Currently, it is not possible to create an account on the core banking service, reducing potential new customers for DinoBank.

Also, there are instances of sensitive information being disclosed by the core banking service. The error messages given by the account creation form directly print SQL error messages, giving an attacker valuable information on the makeup of the DinoBank's customer database, allowing an attacker to better craft SQL injections. The following screenshots were taken when trying to create a DinoBank account:

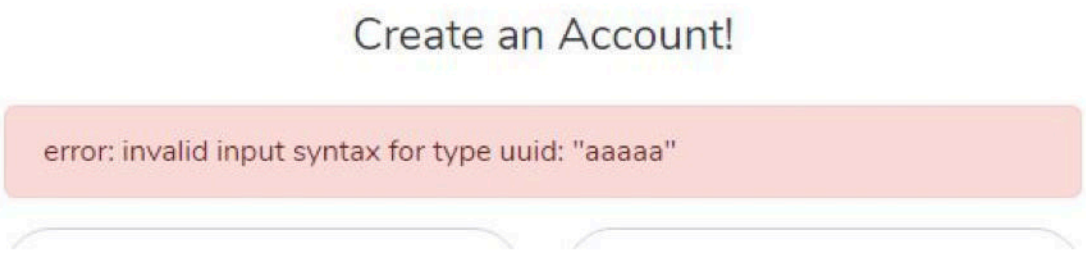


Figure 13. When the password field is "aaaaa"

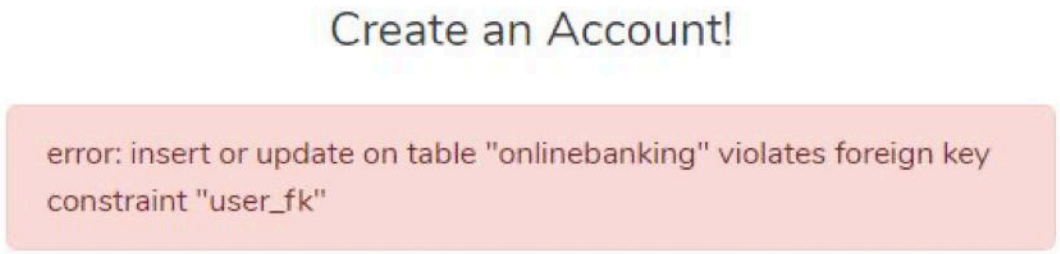


Figure 14. When the password field is a UUID

The core banking application also reveals social security numbers in plaintext. This is a poor security practice because a customer would be at risk when using the application in a public area.

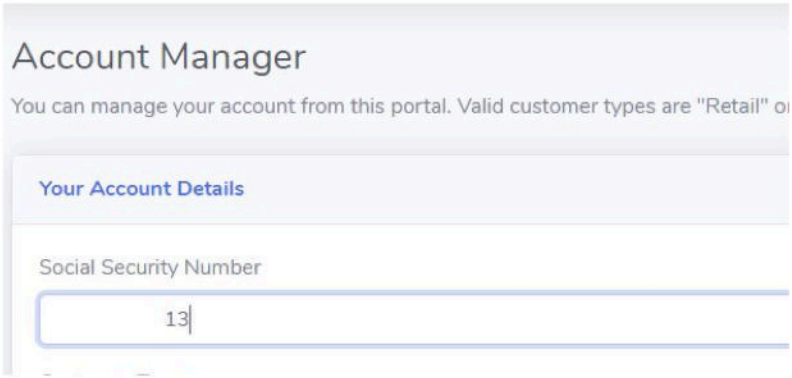


Figure 15. Plaintext SSN in account manager

Unencrypted HTTP

In the first security assessment, multiple websites were found that only implemented HTTP. In the second assessment, most of the websites were forcing the use of HTTPS. However, an exception was port 8000 on 10.0.2.15, which did not implement HTTPS:

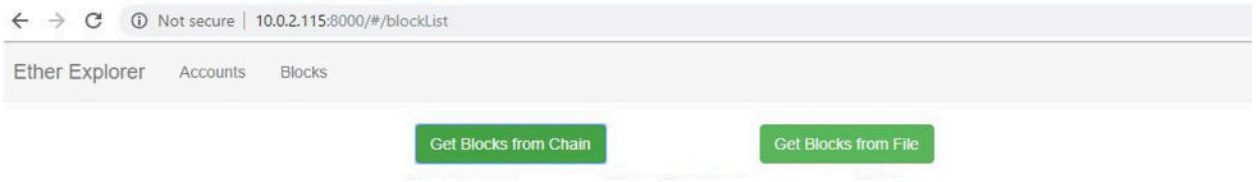


Figure 16. HTTP allowed on 10.0.2.15

HTTP does not encrypt data that is transferred between the user and the server. Therefore, provided that a malicious actor was intercepting traffic on a Wi-Fi network, they would be able to view the user's data, reducing customers' confidence in DinoBank's security.

The vulnerability can be easily mitigated by using HTTPS to communicate between the client and the server. HTTPS uses asymmetric encryption to secure communications. This disallows adversaries from viewing sensitive customer information. A free and easy certificate authority for implementing HTTPS is Let's Encrypt (https://letsencrypt.org).

Automated Teller Machine

The ATM model that DinoBank used was a Hyosung Mini-Bank 1500 NH-1520. A manual was found online (http://atmofamerica.com/Manuals/Hyosung/MB1500.pdf), which included default passwords for the operator menu. The master key was still the default two passwords. However, DinoBank succeeded in changing all other passwords so that the default passwords could not be used.

During the security assessment, a DinoBank employee logged into the operator menu on the ATM but did not logout. Therefore, the team had access to the operator menu for some time. This could have allowed a malicious actor to change the amount of money that is dispensed by the ATM, which would allow them to steal money from DinoBank. Employees should practice sufficient operational security by logging out of the operator menu when done.

Interactive Voice Response System

The primary concern with IP telephony security is denial-of-service(DoS) attacks and eavesdropping, which can compromise operations and the security of customer and corporate data. DoS attacks are accomplished by repeatedly dialing the IVR system to the point where it becomes overwhelmed and cannot respond to legitimate users. This assessment was not able to replicate this vulnerability in this system. The vulnerability of eavesdropping is inherent to the nature of this machine, whether from people listening physically to the phone call or a man-in-the-middle intercepting the trafficking over the network. The best practice to mitigate the release of information is to limit what information is shared during the communication between the client and the system.

Mild fingerprinting was accomplished with network scanning. The Interactive Voice Response system ran on a Ubuntu server, only two ports were open, ports 22 and 5038. An OpenSSH version 7.6p1 service was running on Port 22, but was secured adequately. The Asterisk Manager Interface was found to be running on Port 5038, which is the default port for that service. However, there were not any vulnerabilities identified.

Windows Active Directory

Reused Administrator Passwords

The domain administrator account and the local password for many of the local administrator accounts used the same password. This can be verified by logging into the domain administrator account and the local administrator account on most of the computers.

Provided a malicious actor obtains the password for one local administrator account, they could login to the domain administrator account. This escalation of privileges could allow anyone with one computer administrator's password to make changes across the organization.

All administrator account passwords should be unique and complex for every computer. Access to computer systems should be given at the lowest possible level and compartmentalized as much as possible. When higher access is needed, there should be another account with special permissions, rarely used and able to access and modify at levels higher than that of the other accounts. Domain administrator accounts should never have the same password as anything else on the network. Because the domain administrator account can affect so much sensitive information, users, and computers on the network, its password should be very complex and unique.

Excessive Domain Admins

In the active directory, when viewing the registered users, they can be sorted by their rights as a user. In total, 23.8% of the users have domain level administration access:

Figure 17. 23.8% of users of domain admin rights

Domain admins can change policies and control every network connected to the domain controller. Domain administrators can install new software, change DNS settings (redirect all computers traffic), and modify which other computers have access to the network. More domain administrators increase an attacker's possibility of gaining domain admin rights and the chance of an insider threat.

To prevent an excessive number of users from being domain administrators, a review of the access levels that every employee needs should occur. The number of users with domain admin rights should be reduced to those that require the permissions for their position.

Plaintext Passwords in PowerShell Logs

On the unsecured ftp server, the presence of Windows PowerShell logs (C:\pstrans\) exposed the passwords, in plaintext, of multiple local Administrator accounts as well as the DINO\Administrator account.

This is an extreme security risk, as it is a trivial matter to find given FTP access and quickly allows an attacker lateral movement through nearly all of Dinobank's internal infrastructure.

To prevent this, all logs present on DinoBank machines should be sanitized and cleared of any sensitive data. In the future, no passwords should be present in plain-text within any command-line interface command.

Anonymous FTP Login

Access the FTP by going to the following address in Windows Explorer:

```
ftp://10.0.1.12/
```

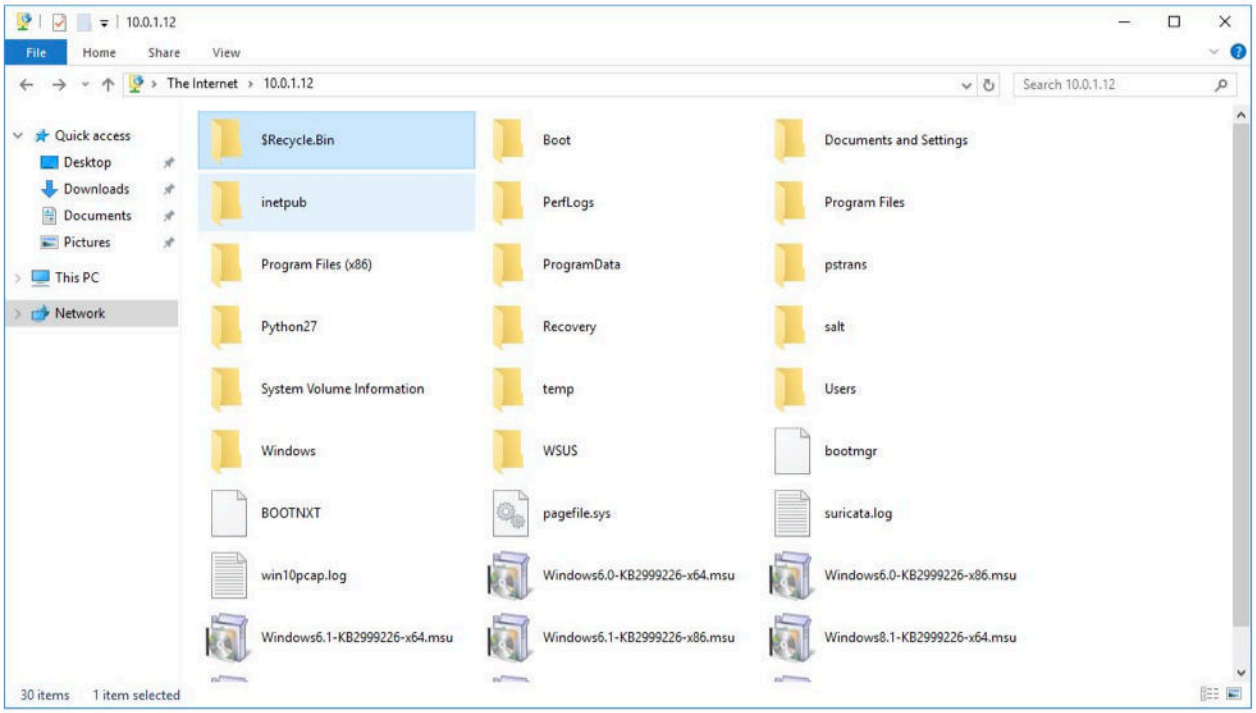


Figure 18. File system of 10.0.1.12

This allows anyone on the network to be able to read almost all files in the 10.0.1.12 filesystem. Attackers can use this information found to increase unauthorized access to the network. For example, PowerShell logs containing a plaintext password were found on the FTP server.

Shut down the FTP server or disable anonymous login on the FTP server.

Miscellaneous

Cryptocurrency Miner

An application, miner.exe, can be found on five computers throughout the network. This seemingly malicious application was found within the Windows C:\System32 folder in an

attempt to hide it. Other methods of stealth can be found when the application was reverse-engineered. The computers that the executable was found on are 10.0.1.11, 10.0.1.12, 10.0.1.20, 10.0.1.31, and 10.0.1.50.

There is reason to believe that this application is a cryptominer due to its stealth capability as well as its name 'miner.exe'. The binary was analyzed in Ghidra, a reverse-engineering tool:

```
if (lVar2 < 1) {
  do {
    time.Sleep(1000000000);
  } while( true );
}
time.Sleep(lVar2 * 60000000000);
runtime.closechan(local_60);
return;
```

Figure 19. Ghidra pseudocode of stealth functionality

Further information about the binary is below:

```
File Name: "miner.exe"
File Size: 2040005 Bytes
MD5    : 97f2a687be44677a69db1a94f686edc1
SHA256: e6da3ebed0b7ed4b0db8beb6b4e4e6565b727ce5080051c7aaee28fdb39710a4
Notable Strings   : "/cptc/infra/lucas/apps/miner_binary/main.go"
                  : "main.printHelp"
                  : "main.main"
                  : "main.main.func1"
```

This application was uploaded to VirusTotal, a malware analysis service, which revealed it was not a known malware sample.

This application degraded corporate computer performance. This is because this application was a cryptocurrency miner that uses significant processing power to do computationally intensive tasks. This application was reported to slowing down corporate by Lawrence Hayden (https://www.reddit.com/user/lawrence_hayden/) on the company's subreddit page.

This application was noticed by not only by Lawrence Hayden, but also Alex Woods and Dan Oliver. They attempted to have the application removed by contacting those responsible for it by email, but there was never a total removal of the application. In the future, it is important to make sure that suspicious software is removed promptly and that someone verifies that it is removed.

Sensitive Information on Windows Desktop

When accessing the 10.0.10.201 and 10.0.10.202 Windows servers, there was a passwords.txt file on the desktop that contained plaintext passwords.

A malicious actor could use plaintext passwords stored in text documents to gain unauthorized access to more services.

Train employees to not store plaintext passwords and perform periodic audits on servers to ensure there are no such files.

Sensitive Information in Wiki

Access the DinoBank Wiki at 10.0.1.31 and browse the Network Access and IT-Ops Network-Infrastructure pages:

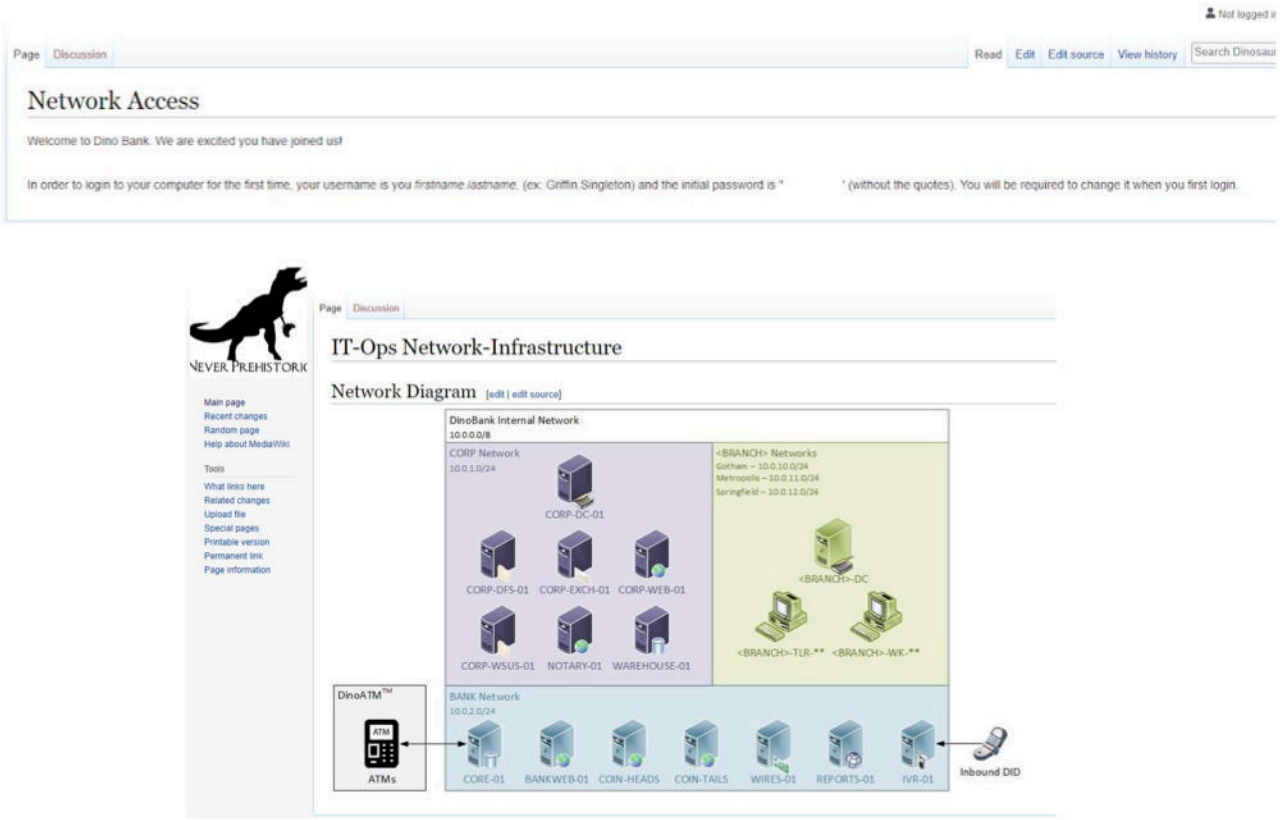


Figure 20. Wiki pages containing sensitive information

An attacker could use default credential to gain unauthorized access to new accounts. An attack could also use the network map increase their knowledge and thus efficiency of attacking the network. Limit access to the wiki to authorized employees.

APPENDIX A: TOOLS USED

TOOL	DESCRIPTION
Burp Suite Community	Used for penetration testing of web applications
Nmap	Used for scanning hosts
Sn1per	Used for scanning and web vulnerability analysis
Metasploit	Used for the exploitation of vulnerable services and vulnerability scanning
FTP	Used for connecting to FTP servers
psql	Used for connecting to PostgreSQL databases
CyberChef	Used to convert between different data formats
Wireshark	Used to sniff traffic for proof of concept
dirb	Used to brute force URLs
Ghidra	Used for reverse-engineering of binaries

APPENDIX B: CORRESPONDENCE

Summary

Network Situation

We have had success in the initial stage of the security assessment. On the DinoBank intranet wiki, we were able to find the default credentials for new users and a network map. Both are publicly accessible.

One issue with the last security assessment was that anonymous logins were enabled on the 10.0.1.12 Windows FTP server. Looking through the server, we were able to find a local administrator password, which has allowed us to gain full access to all the domain controllers. This was found in PowerShell history files. We were then able to login to all the domain controllers. With these credentials, we were also able to login to the mail server for the domain. The emails on this computer contained an API key and port number, which we are looking into.

We were also again able to login to the PostgreSQL database with the username postgresql and a blank password. The password complexity has improved in the database, but they are still not hashed and complexity could be further improved. SSNs are still in plaintext.

On the core banking application, there is an error on the registration and reset password pages. Error messages include SQL queries, which should not be disclosed. An API key for the core banking application was also found in an email.

On multiple computers, we found a miner.exe. From the emails found on the email server, we realized that this is the crypto miner application that was mistakenly installed onto some work computers. This has also been the cause of slow and noisy computers on the network. Despite Alex Faulkner saying in multiple emails that he would remove the application from the network, and he has failed to do so.

We have recently begun researching possible vulnerabilities for both the ATM and IVR system.

PII Controls Report

Our team takes great care when handling both employee and customer personally identifiable information (PII). A breach could result in a loss of employee and customer confidence in DinoBank and repercussions from external auditors. Since it is a financial institution, DinoBank's systems include extremely sensitive information, including social security numbers, addresses, and account numbers.

Our team will take multiple steps to ensure that DinoBank is PII is protected. This includes:

> Our team will avoid coming in contact with DinoBank PII without explicit permission from DinoBank to prevent unnecessary exposure to sensitive data.
>
> If we feel that we would benefit from transferring PII from a DinoBank system to the DinoBank Google Drive, only minimal data will be used and not all the data. Additionally, encryption will be used to prevent others from intercepting the data.
>
> In no circumstance will PII be transferred from a DinoBank system to a non-DinoBank system.
>
> PII will not be included in the report. If an exploit needs to be proven, only minimal and redacted information will be shown.

Additionally, our team will be adhering to the standards outlined in NIST SP 800-122, "Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)" to best ensure that we are doing the best job possible for DinoBank and their customers. Some of these practices include implementing policies that limit access rules for PII within a system, PII retention schedules and procedures, PII incident response and data breach notification, privacy in the system development life cycle process, limitation of collection, disclosure, sharing, and use of PII, and consequences for failure to follow privacy rules of behavior. We also implement security practices that help safeguard PII, which includes separation of duties, least privilege, limiting remote access, and access enforcement. We also perform measures to de-identify information and anonymizing information by generalizing the data, suppressing the data, enforcing noise into the data, wapping the data, and replacing the data with the average value.

APPENDIX C: PREVIOUS REPORT

[redacted]

2019 Security Assessment Report Prepared For

Report Issued: 12 October 2019

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The participants of the [redacted] penetration testing team performed a security assessment of the internal corporate and customer network of DinoBank on 12 October 2019. Our penetration test simulated an attack from a threat actor attempting to gain access to systems within the DinoBank corporate and customer network. The purpose of the assessment is to discover and identify vulnerabilities in the corporate infrastructure. We discovered 15 total vulnerabilities totaling 2 critical vulnerabilities, 3 high risk vulnerabilities, 3 medium risk vulnerabilities, and 7 low risk vulnerabilities within the scope of the engagement. The vulnerabilities are classified according to Table 1. In order to ensure data integrity, availability, and confidentiality, security remediation should be implemented to the present network as described in the security assessment findings. Please note that this assessment does not, nor was it intended to, disclose all vulnerabilities that are present on the system. Any changes made to the environment during the period of testing may affect the results of the assessment. However, the team made a significant effort to prevent modifications to the bank's network because it was a production system.

Many of the vulnerabilities found in this security assessment would result in numerous federal compliance issues for financial institutions, including the Gramm-Leach-Bliley Act (GLBA), the Payment Card Industry Data Security Standard (PCI DSS), Financial Crimes Enforcement Network (FinCEN) regulations, and the USA PATRIOT Act. For example, the GLBA requires that financial institutions, such as DinoBank, prevent its customers' nonpublic personal information, including social security numbers, bank account numbers, and addresses, from being accessed by unauthorized personnel. In our security assessment, we found multiple vulnerabilities that would allow an adversary to access this information. Thus, DinoBank could be found in violation of the GLBA and pay a fine of $100,000 for each violation.

This report provides information on how one can reproduce the found vulnerabilities and how to mitigate them. DinoBank's information security and software development teams should carefully read this report and implement its recommendations in order to prevent the company from violating federal regulations or losing a significant number of customers due to privacy breaches.

SECURITY ASSESSMENT SUMMARY

Table 1. Security level descriptions

Security Level	Description
Critical	Successful exploitation will result in a major disruption of business functionality. It is recommended that the vulnerability is remediated or mitigated immediately.
High	Successful exploitation will result in elevated privileges, significant data loss, or any other severe detriment to business functions.
Medium	Successful exploitation will result in the inadvertent disclosure or modification of sensitive information or the interference with user interactions or corporate auditing procedure.
Low	These findings have minimal impact on business functions and typically require local or physical system access.

Table 2. Summary of vulnerabilities

#	Risk	Observation	Impact to DinoBank	Remediation
1	C	Banking Logic Issue	The exploitation of this vulnerability will cause DinoBank significant financial loss, legal issues, and consumer mistrust.	Reimplement DinoBank transaction system.
2	C	Empty Database Password	An attacker can access DinoBank's main PostgreSQL database with no password needed and access/change customer data, employee data, and account data.	Add a password to the bank database.
3	H	Sensitive Data Stored Unencrypted	Once an attacker has access to the database, sensitive data (SSN, passwords, etc.) can be read in plaintext and exported easily.	Convert the website to secure hashing for data not intended to be only stored (passwords), and secure encryption for data intended to be accessed.
4	H	Unencrypted HTTP	Attackers can intercept passwords and other sensitive data and perform Man in the Middle attacks.	Enable HTTPS.
5	H	Reflected XSS	Malicious user input in a crafted URL can lead to session hijacking.	Perform input validation within any strings added to the DOM.

6	M	Insecure Cookies	Coupled with another exploit, session hijacking is possible through access to the document.cookie JavaScript variable.	Set the HttpOnly and SameSite flags on cookies sent to clients.
7	M	Web Application Open Redirect	An attacker could use a link to a DinoBank website to redirect to any arbitrary URL, abusing the customer's trust of DinoBank.	Check links to make sure they're trusted before redirecting to them.
8	M	Anonymous FTP Login	An attacker can login to the FTP service and download configuration files or other confidential data that can be used to facilitate further access.	Disable anonymous FTP login.
9	L	NodeJS Running as Root	If NodeJS is exploited, the permissions given to the attacker would be root.	Run NodeJS with a reverse proxy.
10	L	Input Autocomplete Enabled for Sensitive Fields	Though it makes browsing easier for your users, input autocomplete poses a risk for shared computers because a malicious user can gather the information.	Disable input autocomplete on unnecessary or sensitive fields.
11	L	Weak Password Requirements	Most passwords found in database are weak.	Increase password complexity requirements.
12	L	Employee Use of Git	An employee committed a private SSH key to a public repository.	Train employees on proper security procedures.
13	L	Developer Notes in Production Site	These messages are intended for the developer during the setup of the application, and serve no purpose to the end user.	Remove messages that aren't relevant to the end user.
14	L	Web Application CSRF	An attacker could arbitrarily logout users without their consent.	Require a CSRF token for requests.
15	L	User Interface Issues	Users are not able to log out of accounts, allowing for possible unauthorized access on shared computers.	Ensure that users are able to logout with ease.

TESTING METHODOLOGY

[redacted] testing methodology used to conduct the penetration test can be split into three phases - reconnaissance, target assessment, and execution of assessment. Reconnaissance is comprised of gathering information about DinoBank's network systems. Port scanning and other enumeration methods were used in order to refine targetable information. After refining our target set, we began our targeted assessment. Execution on vulnerabilities was conducted in a manner that would not disrupt normal business operations. The following figure is a graphical representation of this methodology.

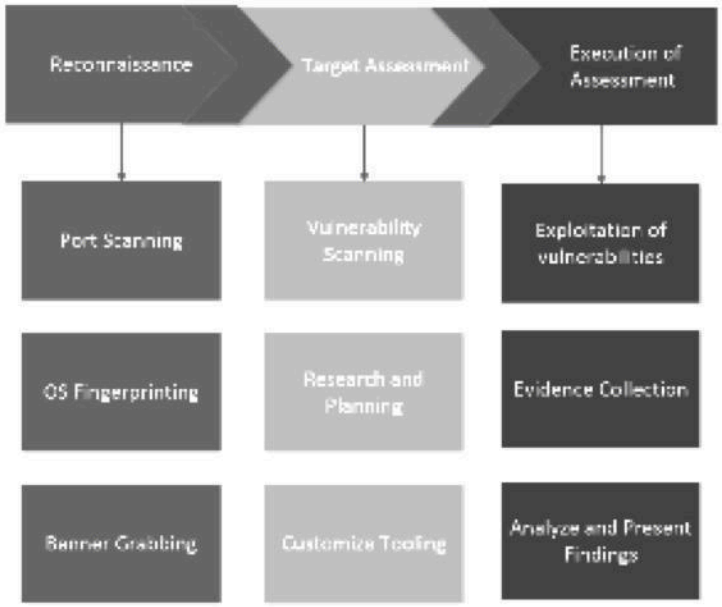


Figure 1. Testing methodology

The topology of the network was found using nmap and scanning the five given subnets, which were 10.0.1.0/24, 10.0.2.0/24, 10.0.10.0/24, 10.0.11.0/24, and 10.0.12.0/24. After ping scanning the given hosts, it was discovered that the 10.0.11.0/24 and 10.0.12.0/24 networks were both down. This can be attributed to the fact that Mr. Dickson informed us before the assessment that the Springfield and Metropolis networks were not operating on the day of the assessment. To gain more information besides which hosts were up, the following nmap scan was run:

```
nmap 10.0.1.0/24 10.0.2.0/24 10.0.10.0/24 -T4 -A -p-
```

The result of this nmap scan can be found in Appendix B. The scan results contained open ports, possible operating system versions (OS), and possible service versions for each host. Using this information, we created a topology of the three networks:

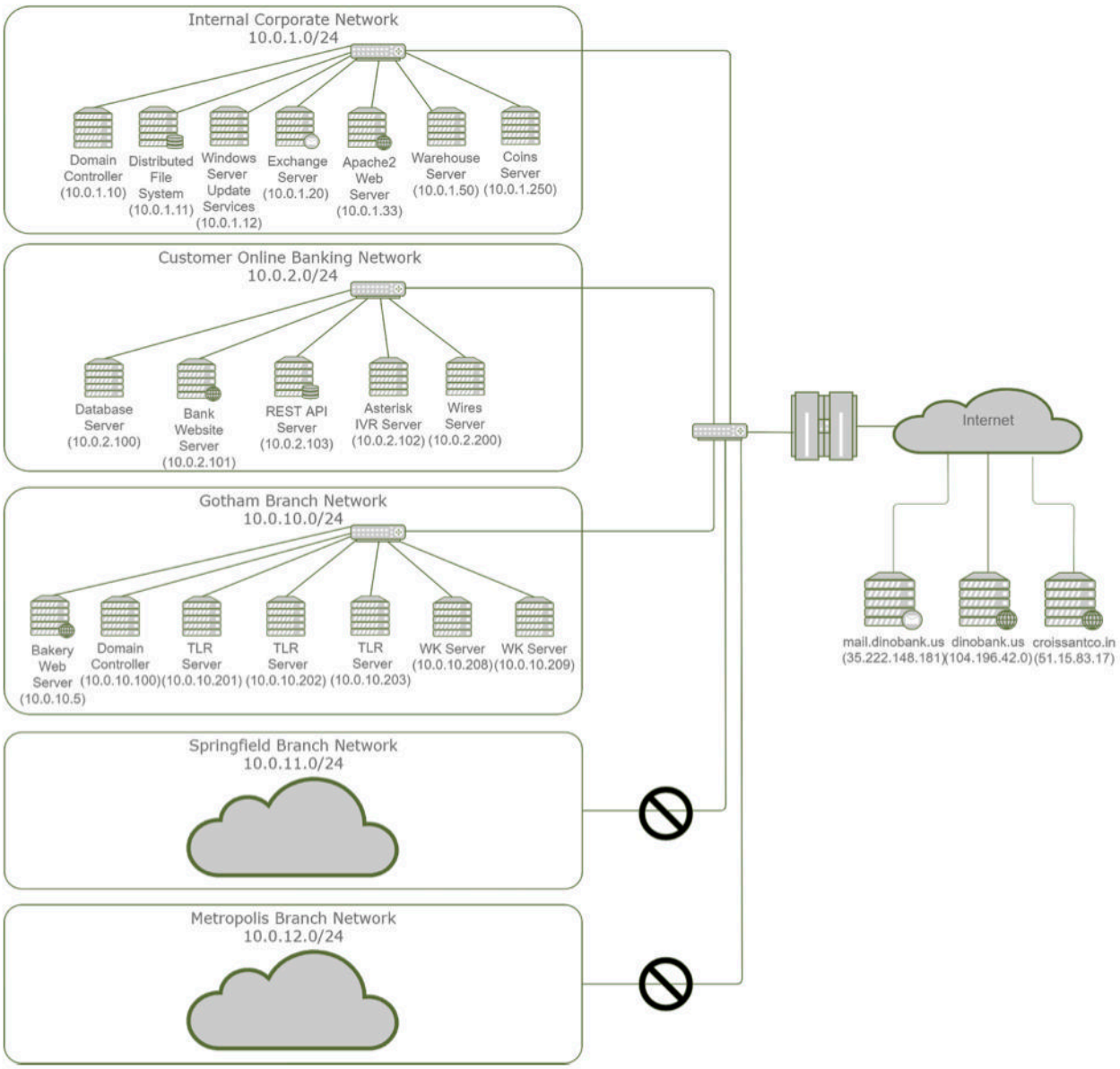


Figure 2. DinoBank network topology

Once the reconnaissance phase was completed (approximately one hour into the assessment), the target assessment and execution assessment phases were completed simultaneously. Each of the six penetration test team members were assigned to a different host. We would first conduct research on the host, such as simply viewing a web server in a browser or researching the version number of its service and operating system. With this information, we looked into possible vulnerabilities and attempted to produce a proof of concept without causing significant modification of the system.

FINDINGS

Banking Logic Issue

On the online banking application located at 10.0.2.101, users can transfer money from their account to another user's account. We noticed, after creating our own account, that we were able to transfer money from our own account to the same account. If the transfer amount was positive, it would increase the user's account balance by the transfer amount. On the contrary, using a negative transfer amount would decrease the user's account balance. This allows anyone to arbitrarily create or destroy money, violating significant federal banking regulations. To prevent this, the PHP code behind the web server should check if the transfer is going to and from the same account. It should be noted that our actions were immediately reversed.

Empty Database Password

One of the most severe vulnerabilities found on DinoBank's networks was a PostgreSQL server that did not require a password. 10.0.2.100, a server hosting a REST API for DinoBank's online banking solution, was found to have port 5432 open, which is primarily used for PostgreSQL servers. Thus, we attempted to connect to the database server. We tried to authenticate using the psql client with numerous usernames, until the username "bank" was successful without prompting for a password. The command used was:

```
psql -h 10.0.2.100 -U bank
```

Once connected, we could run SQL commands and enumerate the database, tables, and column names. The database name of interest was "bank", and it included the tables employees, customers, and accounts. The bank user had both read and write access to all of these tables. The employees and customers tables contained sensitive, personal information, including their username, password, social security number, and job title. This information could result in identity theft or unauthorized access to an employee or customer's account.

Additionally, with the database modification privileges, an attacker could modify account balances via the accounts table, which would violate federal regulations. An attacker could also sabotage the database by deleting tables, which would result in system downtime and a permanent loss of data unless backup procedures are in place.

To mitigate this issue, a password can be set with the following command in the PostgreSQL client:

```
\password bank
```

Furthermore, if the database only needs to be accessed by the NodeJS service running on the same server, one should disable access to the PostgreSQL server from remote hosts. This can be done by modifying the pg_hba.conf configuration file.

Sensitive Data Stored Unencrypted

The passwords stored in the PostgreSQL database at 10.0.2.100 for employees and customers were stored in plaintext. This means that if one were to gain unauthorized access to the database, they would be able to login to these employee and customer accounts without any additional operations. Additionally, an attacker could attempt to reuse the passwords on another website.

Passwords stored in a database should be hashed, which is a one-way function. This allows the server to authenticate users without storing the passwords in plaintext. A secure hashing function that could be used is bcrypt. It salts the passwords and hashes them an explicit number of times to prevent brute forcing.

Furthermore, other personal and sensitive information, such as social security numbers and addresses, are also stored in plaintext. To make this information more secure, they should be encrypted with an algorithm such as AES. An attacker would then need to exploit both the database and the filesystem to obtain the AES key.

Unencrypted HTTP

Multiple web application servers in DinoBank's networks were found to be using unencrypted HTTP rather than the currently accepted and much more secure standard, HTTPS. This means that data sent between the client and the server is in plaintext, which would allow anyone sniffing the network traffic to be able to view sensitive data. Furthermore, cyber-criminals will often attempt to compromise credentials passed from the client to the server using HTTP. This can be conducted via various different Man-in-The-Middle (MiTM) attacks or through network packet captures. A MiTM attack allows an attacker on the same network as the customer to pose as the DinoBank website that could be hosting malware.

One significant example of this vulnerability was on the host with the IP 10.0.2.101, DinoBank's online banking solution for customers. For example, when a user creates a DinoBank account (http://10.0.2.101/register.php), personal information can easily be stolen by an attacker because their data is sent unencrypted in the body of the POST request. Wireshark, a network traffic capture tool, was able to view the following data when a user registers:

```
HTML Form URL Encoded: application/x-www-form-urlencoded
> Form item: "companyName" = "None"
> Form item: "customerType" = "Enrolled"
> Form item: "givenName" = "Some"
> Form item: "middleName" = "Guy"
> Form item: "surName" = "WhoDoesntKnow"
> Form item: "customerType" = "Retail"
> Form item: "taxId" = "123456789"
> Form item: "phoneNumber" = "123456789"
> Form item: "streetAddr1" = "123"
> Form item: "streetAddr2" = "456"
> Form item: "cityName" = "789"
> Form item: "stateCode" = "IL"
> Form item: "id=" = "777777"
> Form item: "emailAddr" = "abademail@gmail.com"
> Form item: "password" = "1234"
> Form item: "repeatPassword" = "1234"
```

Figure 3. Wireshark capture of http://10.0.2.101/register.php POST request

For example, a user's taxId (their social security number) and password can be viewed unencrypted by anyone sniffing traffic. This could result in identity theft and an attacker gaining unauthorized access to a customer's account and personal information. Not only can the registration and login page's information to include a current user's password) be intercepted, but the user's cookies as well, which would allow an attacker to hijack a customer's session.

This vulnerability can be easily mitigated by using HTTPS to communicate between the client and server. HTTPS uses asymmetric encryption to secure communications. This disallows adversaries from viewing sensitive customer information. HTTPS also prevents MiTM attack because its certificate verifies the authenticity of the website. A free and easy certificate authority is Let's Encrypt (https://letsencrypt.org). It is highly recommended that DinoBank's system administrators look into implementing HTTPS on all of DinoBank's web servers to prevent the disclosure of customers' sensitive information.

Reflected XSS

A reflected cross-site scripting (XSS) vulnerability was found on the online banking application at 10.0.2.101. The error page, alert.php, allows the addition of arbitrary error messages encoded as Base64 in the errorid GET parameter. However, an attacker can add a <script> tag to the page using this parameter. The below figure shows an attacker is able to run JavaScript on the page (an alert message in this example). An attacker could use the document.cookie variable to send one's cookie back to a malicious server, and then they would be able to hijack their session.

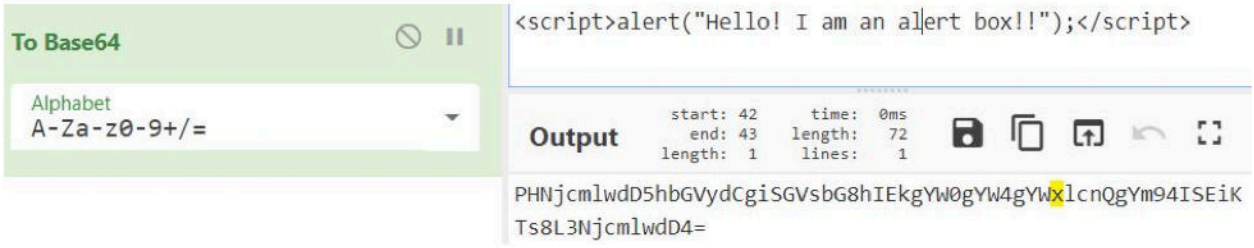


Figure 4: Encoding of JavaScript function

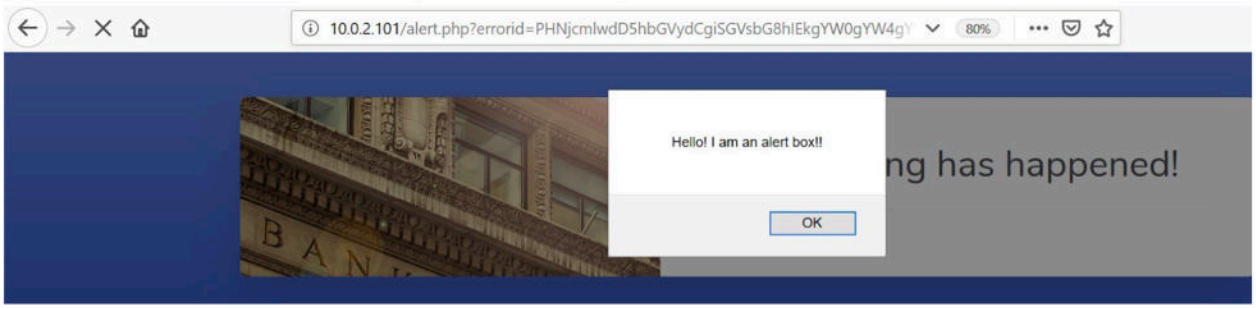


Figure 5: Proof of concept for XSS

To mitigate this issue, the GET parameter should be filtered with the htmlspecialchars PHP function.

Insecure Cookies

The current implementation of authentication cookies put clients' DinoBank accounts at risk to session hijacking. The HttpOnly flag is not set on the cookie, which would allow a cross-site scripting (XSS) attack to steal a user's cookie with the document.cookie JavaScript variable. Setting the HttpOnly flag on all cookies would prevent session hijacking via XSS.

Web Application Open Redirect

The new.php endpoint on 10.0.2.101 takes in a GET parameter called source. The page will redirect to the URL in the source parameter. However, the URL can be set to a remote server. Therefore, an attacker could redirect a user to a malicious website using a DinoBank URL a user would trust (such as a phishing email). To prevent this, new.php should check if the URL is on a remote server and prevent the redirect if it is.

Anonymous FTP Login

The nmap scan revealed that FileZilla server was running on port 21 on the host 10.0.1.12. Running the Metasploit module “auxiliary/scanner/ftp/anonymous” on the host exposed that anonymous FTP logins were allowed with read permissions. Using an FTP client, one can view numerous files in the C: drive on the Windows server. This leads to information disclosure, allowing an adversary to discover sensitive files, such as server configurations, on the remote server. To prevent this vulnerability, system administrators should disable anonymous logins on the FTP server via the FileZilla configuration file, which is located at “C:\Program Files (x86)\FileZilla Server\FileZilla Server.xml”. FTP could also be disabled altogether if it is not needed.

NodeJS Running as Root

The nmap scan revealed that NodeJS is running directly on ports 80 and 443. In order to run on these ports, the service must have root access. Therefore, NodeJS is running as a root, which would give an attacker more privileges if they exploit the REST API. To combat this, NodeJS should be run behind a reverse proxy such as Nginx so it is no longer running as root.

Input Autocomplete

Webpages found on network, such as 10.0.2.101, were found to have autocomplete enabled and allowed a browser to cache previously entered credentials. While this may be used to quickly re-enter credentials on the same page it may also allow an attacker with access to the machine the ability to retrieve those same credentials by visiting the page. This can be remediated by disabling the autocomplete feature on the password field.

Weak Password Requirements

Passwords found in PostgreSQL database are weak and easy to guess. Almost all passwords found, to include passwords used by network administrators, were not very complex. For example, many passwords were only 8 characters and only included one uppercase character and one number. DinoBank should apply strict password requirements, such as requiring at least 12 characters, one uppercase character, one lowercase character, one number, and one special character.

Employee Use of Git

A public GitHub repository (https://github.com/DinoDanOliver/.files) was found for a DinoBank employee, Dan Oliver. Viewing the commit history, it was discovered that Dan Oliver originally had a private SSH key in the repository, but later removed it. However, since Git is a version control system, the file can still be found (https://github.com/DinoDanOliver/.files/commit/cb765593bd416c9f573f21ca1c1b07bdccfe14d6). Employees should be trained to not commit any sensitive information to Git repositories, especially a public one.

Developer Notes

Developers left hints of unimplemented and potentially vulnerable services within the source code of the webpages. For example, the following code was found on http://10.0.2.101/login.php:

```
<!--<div class="text-center"><a class="small"
href="forgot-password.php">Forgot Password?</a></div>-->
```

Although forgot-password.php was not found to be vulnerable, a continuation of this practice will lead lead to unintended information leakage. This issue can be remediated by removing irrelevant and potentially revealing comments from webpages.

Web Application CSRF

The online banking application at 10.0.2.101 does not utilize cross-site request forgery (CSRF) tokens, which allows for CSRF to be possible. For example, an attacker could have a link on a website that forces a user to logout using logout.php (http://10.0.2.101/logout.php). Most endpoints on the web application should implement a CSRF token as a hidden form field to prevent CSRF.

User Interface Issues

Poor user interfaces put DinoBank clients at risk when using a shared computer. On the online banking solution at 10.0.2.101, the logout button is not able to be pressed with current implementation, forcing clients to stay logged in even when they no longer need to interact with DinoBank, putting client's personal information and money at risk.

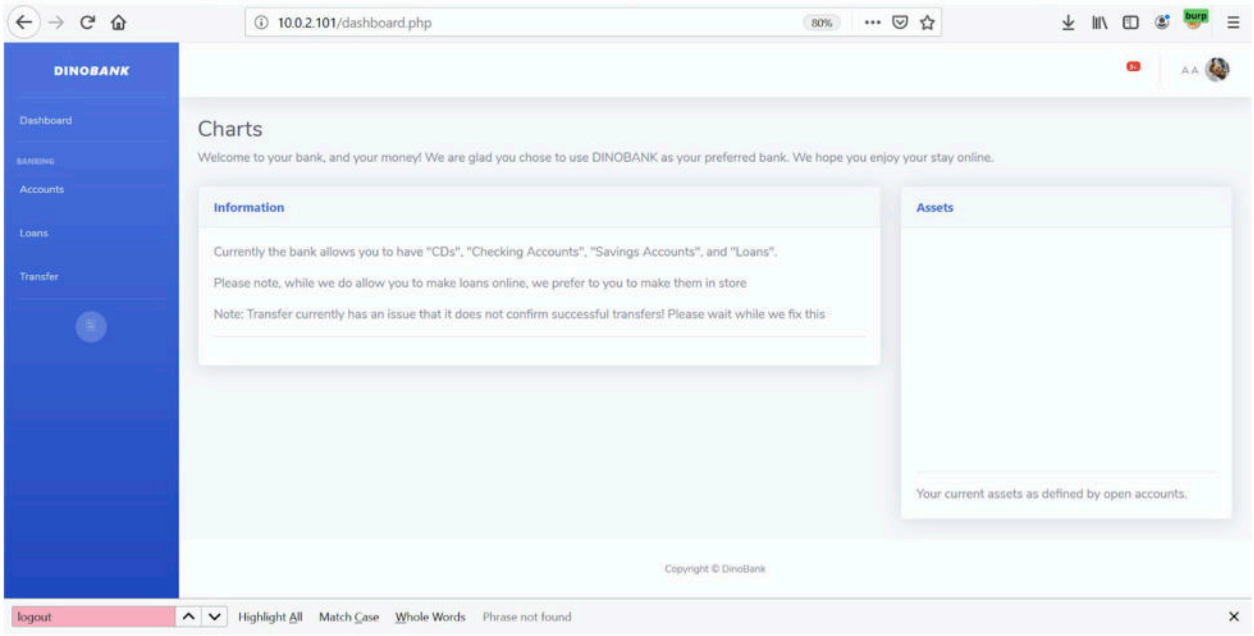


Figure 6: A search finds no way to logout

APPENDIX A: TOOLS USED

TOOL	DESCRIPTION
Burp Suite Community	Used for penetration testing of web applications
Nmap	Used for scanning hosts
Sn1per	Used for scanning and web vulnerability analysis
Metasploit	Used for exploitation of vulnerable services and vulnerability scanning
FTP	Used for connecting to FTP servers
psql	Used for connecting to PostgreSQL databases
CyberChef	Used to convert between different data formats
Wireshark	Used to sniff traffic for proof of concept
dirb	Used to brute force URLs