Penetration Testing Report

[redacted]

DinoBank

November 24, 2019

Table of Contents

Executive Summary

DinoBank contracted our consulting firm to perform an additional penetration test, limited in scope, of their corporate network to be conducted on November 22nd, 2019 and November 23rd, 2019. All testing activity took place between the hours of 8:30 PM - 9:00 PM on the 22nd and 9:00 AM - 6:00 PM EST on the 23rd. The objective of this additional engagement was to identify potential security vulnerabilities in DinoBank's networks and computer systems, as well as identify any vulnerabilities that hadn't been remediated since the initial engagement.

DinoBank's regulators have recently issued a Memorandum of Understanding (MOU) to the bank that details several areas they need to improve on to remain in regulatory compliance and maintain operations. The areas that were identified were lack of security governance, weak passwords, banking core weaknesses, outdated software, poor management, and insufficient audit coverage. We tailored our assessment to cover these areas to provide the most useful results to DinoBank.

During the engagement, our team identified several security issues and misconfigured services on DinoBank's infrastructure. Many systems and services were observed to be out-of-date and misconfigured, however, some systems were up-to-date and securely configured. Should an actual breach occur by an advanced threat actor, this could lead to extensive compromise and reputational damage to DinoBank.

The report for our previous assessment of DinoBank's network noted fifteen security findings and provided recommendations to fix them. We were able to verify that three of the previously identified issues were fully remediated and the rest were either partially fixed or still in a vulnerable state. For instance, we noted that there was an effort to remediate unauthenticated access to the database, but we still managed to gain unauthorized access to the database by logging in with the 'postgres' user. For our previous finding of a lack of encryption, we noticed that several services implemented some form of encryption, but it was not widespread or always enforced.

As an overall ranking, our firm assesses DinoBank's overall security posture to be somewhat below industry peers, posing a high overall risk to the Bank and its customers. It is reasonable to assume that a sufficiently advanced threat actor could do significant harm. By remediating the findings outlined in this report and following the recommendations provided, DinoBank can become an industry-leading organization, in terms of security, that provides financial services with the most cutting edge technology.

Regulatory Considerations

As a financial institution that operates in the United States and has offices in the states of New York and Illinois, DinoBank is subject to several regulatory frameworks and laws. While many regulations and laws for the financial industry do not necessarily mention cybersecurity explicitly, information security controls are often indirectly implied by the requirements for fraud and abuse protection, personal data protection and privacy, audit requirements, and risk mitigation that these measures are meant to enforce.

One of the most significant of these laws, Gramm-Leach-Bliley Act of 1999 requires financial institutions to prevent unauthorized disclosure of consumer's personal data using a combination of risk assessment, informing customers how their data is shared, and implementing safeguards for the privacy and security of sensitive data. The Banking Secrecy Act mandates that banks have the necessary controls in place to detect potential criminal activity and be able to provide the appropriate notification to relevant law enforcement agencies. It is important to note that there are many other additional regulations that DinoBank may be subject to such as the Dodd-Frank Act, the Federal Reserve Act, and the National Bank Act.

While our firm does not specialize in legal affairs and cannot offer legal advice, we believe that following the recommendations outlined in this report will help DinoBank meet the regulatory expectations it is subject to in the various jurisdictions it conducts business in.

Methodology

Our testing methodology aims to be as precise and efficient as possible, minimizing impact to business operations while maximizing actionable results. Throughout this engagement, our team made sure to prioritize DinoBank customer and employee data. Methods used by our firm align with industry-leading practices and conform to the widely recognized Penetration Testing Standards Execution Standard Technical Guidelines (PTES). These techniques are employed to allow discovering the greatest number of critical flaws possible within a limited timeframe and facing an unfamiliar network environment. We always aim to emulate the actions of an advanced threat actor during a breach scenario. After an initial information discovery phase, focusing on open-source intelligence gathering (OSINT), the team began to assess potential vulnerabilities, how to exploit them, and documented our findings.

Scope of Assessment

- Five network ranges were identified as in-scope for penetration testing activities:
    - 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
- One onsite DinoBank Automated Teller Machine (ATM)

- DinoBank's Interactive Voice Response (IVR) system
- No further information about the environment was disclosed to our firm. Diagrams of discovered hosts and their primary services are available in Appendix A.
- All of our consultant's network traffic originated from the systems provided by DinoBank to access the corporate environment and assess the bank's security posture.

Risk Classification

The following definitions detail the risk classifications used to rate vulnerabilities based on their impact and exploitability. This scale is used as a high-level description; further information related to the risk of an individual finding can be found in the technical analysis section.

Risk Classification	**Description**	**Examples**
Critical	Vulnerabilities that affect all users on the platform, and/or affect the security of the platform or host system(s).	Remote code execution, Vertical authentication bypass, SSRF, XXE, SQL Injection
High	Vulnerabilities that affect more than one user of the platform, and that require little or no user interaction to trigger.	Stored XSS, Directory object reference, User authentication bypass
Medium	Vulnerabilities that affect more than one user, but may also require interaction or a specific configuration.	Open Redirect, Reflected XSS, CSRF
Low	Issues that affect singular users and require interaction or significant prerequisites (MiTM) to trigger.	Common flaws, Debug information, Host header

Assessment Findings

This section provides a detailed description of the findings identified during the engagement. A summary of identified issues, finding severity, our recommendations to remediate, steps to reproduce the finding, and additional resources are included. Our team guarantees the thoroughness and validity of the technical findings included in this report. However, due to the ever-changing nature of modern digital threats, the content below may not fully encompass every vulnerability that may potentially have been exposed. Each finding is color-categorized by severity according to our risk classification system identified above and labeled by the identifier in the upper left corner.

Our assessment identified many findings that decrease DinoBank's overall security posture. The most severe finding that our team found involved the PostgreSQL database on the core-01.bank.dinobank.us server, that was connected to the "DinoBank-core" API on the same server. This database contains highly-sensitive information including usernames and passwords for user's online banking accounts, personally identifiable information for DinoBank customers and employees, as well as current account and loan balances. We also observed that many of the services on the bank's network did not use encryption to protect data both at rest and in transit across the network. This could lead to situations where unauthorized persons could access confidential information that they would not otherwise be able to read, were it to be encrypted. Also, there were multiple instances where applications and services were using outdated versions that contained known vulnerabilities which are detailed in this report. It is in part because of these outdated applications that our firm was able to successfully penetrate DinoBank's network.

DinoBank-01	PostgreSQL Passwordless Authentication
Finding Summary	It is possible to connect to the remote PostgreSQL database server using any account without a password. This could potentially allow a threat actor to modify banking records and compromise sensitive personal information.
Affected Hosts	• 10.0.2.100 (core-01.bank.dinobank.us), port 5432
Remediation	Edit the authentication configuration file found in "/etc/postgresql/10/main/pg_hba.conf" to remove the line containing "host all all 0.0.0.0/0 trust". Implementing the least

	privilege security model is recommended. Further mitigation details are included in the linked resources below.
Reproduction Steps	1. Connect to the postgresql host at 10.0.2.100 on port 5432 2. Login with the postgres user without specifying a password, as demonstrated in the figures below.
Additional Resources	• https://cwe.mitre.org/data/definitions/284.html • http://www.postgresqltutorial.com/postgresql-change-password/ • https://chartio.com/resources/tutorials/how-to-set-the-default-user-password-in-postgresql/

Unauthorized access into the PostgreSQL database was obtained by logging in with the 'postgres' user. The remote host IP address, username, and blank password was provided to gain access to the database as seen in the figure below.

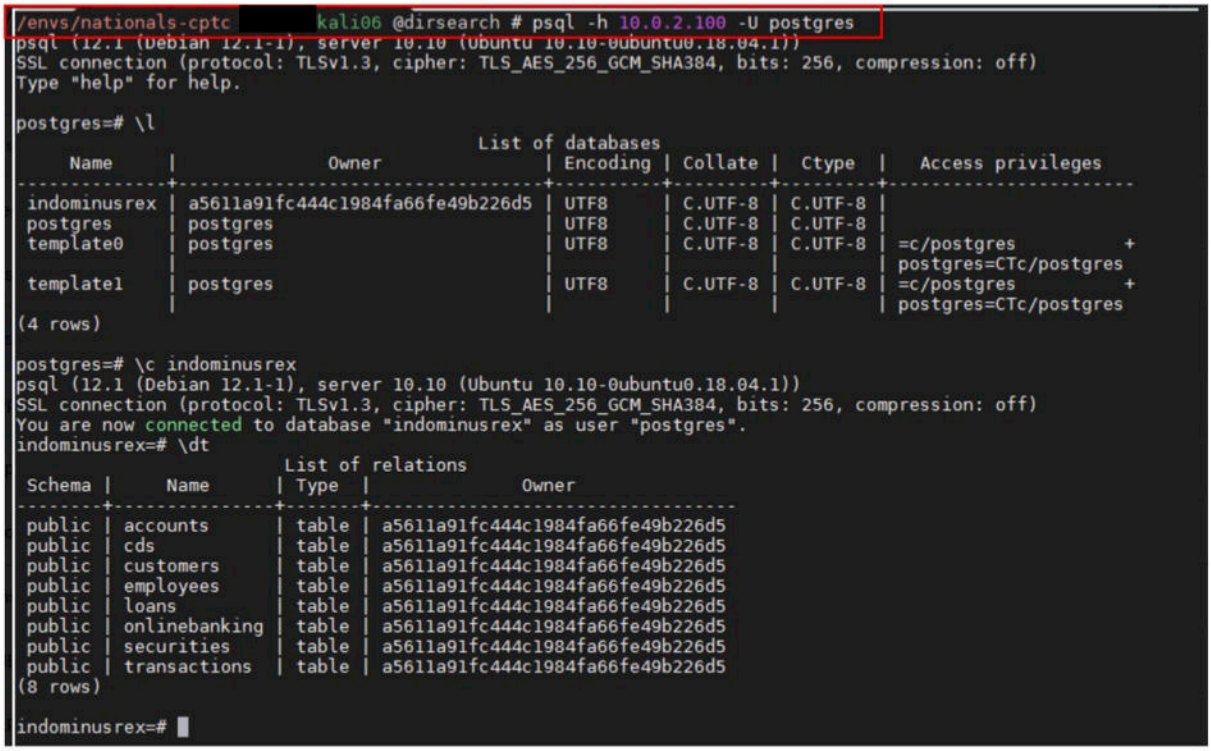

Figure 1. Connecting to Database Unauthenticated

Database access was leveraged to read the contents of files on the system by using a Metasploit module, as seen in the figure below.

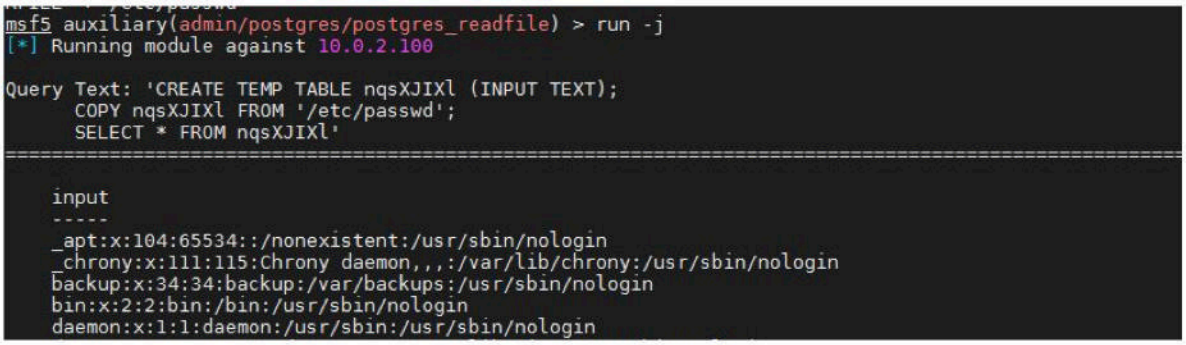

Figure 2. Reading Files on Remote Host

The database access was also leveraged to gain command-line access to the host, which eventually led to our team gaining SSH access to the machine under the context of the "postgres" user.

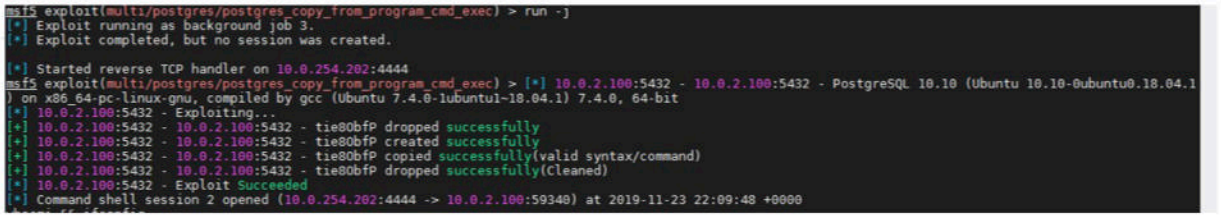

Figure 3. Command Injection - Executing Commands on Remote Host

Figure 4. Interactive SSH session Logged In As ‘postgres’ User

Although the “postgres” user did not have sufficient permissions to access the DinoBank data stored within the database, we were able to use the access we had to enumerate other existing users. Throughout this enumeration, we found the “a5611a91fc444c1984fa66fe49b226d5” user, which did have access to the “indominusrex” database. This user had a password assigned, but due to a misconfiguration of the PostgreSQL service, we were able to authenticate as the user without needing to enter a password. The misconfiguration causing the lack of authentication enforcement can be found in the figure below.

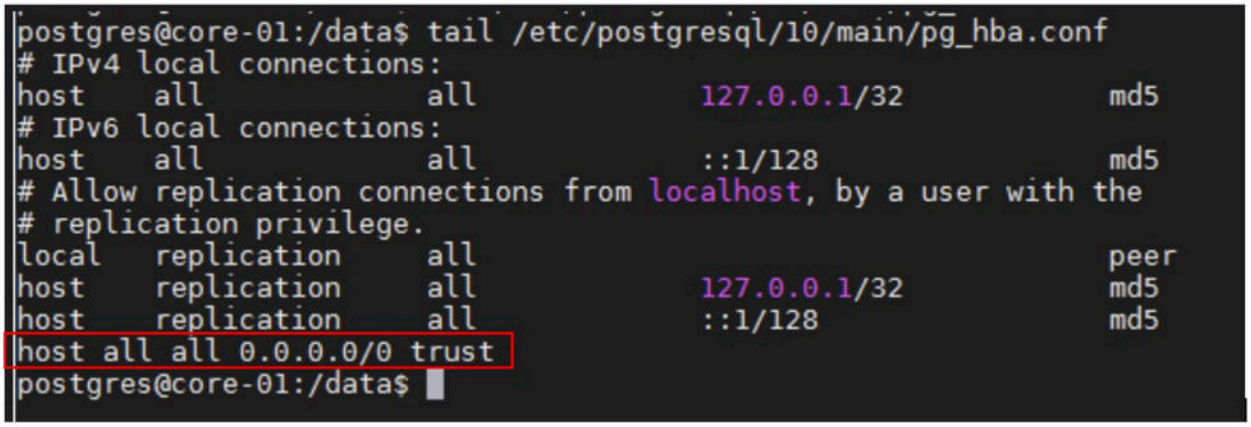

Figure 5. PostgreSQL Misconfiguration

After gaining access to the database via the “a5611a91fc444c1984fa66fe49b226d5” user, personally identifiable information (PII) was exposed in plaintext in the customers and employees table. This included employee and customer tax ids, found under the “taxid” element in the database as seen in the figure below.

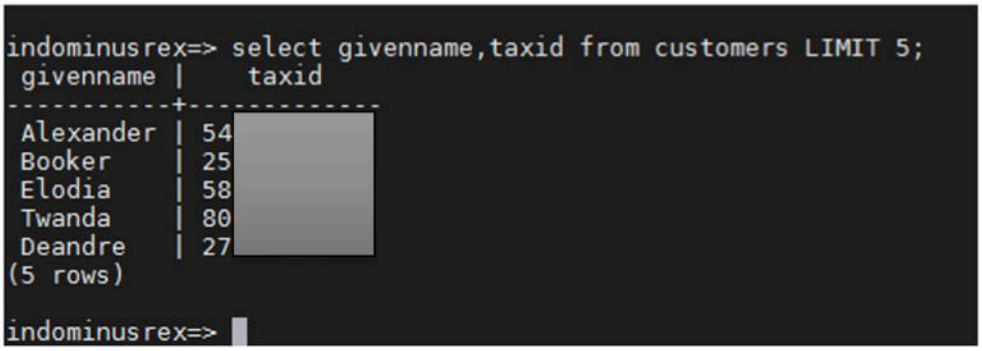

Figure 6. Social Security Numbers Exposed

Not only did we have the ability to read sensitive customer and employee information, but we also confirmed that we had the ability to alter information in the database, including account balances. All testing was performed on the online banking account provided to us.

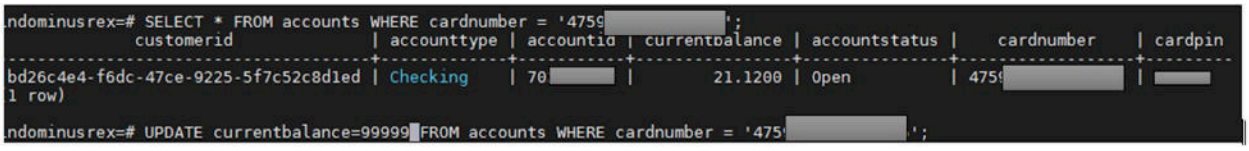

Figure 7. Control Over All Banking Functionality

DinoBank-02	Plaintext Credentials in Database
Finding Summary	Username and passwords are stored in the PostgreSQL database in plaintext.
Affected Hosts	• 10.0.2.100 (core-01.bank.dinobank.us), port 5432
Remediation	Use industry standards for authentication with a third-party service like LDAPS or hash the passwords in the database (with a salt).
Reproduction Steps	1. Login to database 2. Look at the employee's table and view usernames and passwords.
Additional Resources	• https://cwe.mitre.org/data/definitions/257.html • https://cwe.mitre.org/data/definitions/522.html

DinoBank employees' PII were stored in plaintext in the DinoBank Core-01 PostgreSQL database. Not encrypting sensitive data at rest exposes DinoBank to a greater risk of further compromise and exposure in the event of a breach. These credentials could be leveraged by a threat actor to move laterally throughout the environment or could even be sold on the black market.

```
        loginid          |   passwd   |    taxid
-------------------------+------------+-------------
 ali.gamble@dinobank.us | Winter2019 | 444-79-8779
(1 row)
```

Figure 8. Employee Passwords Stored in the Core-01 database

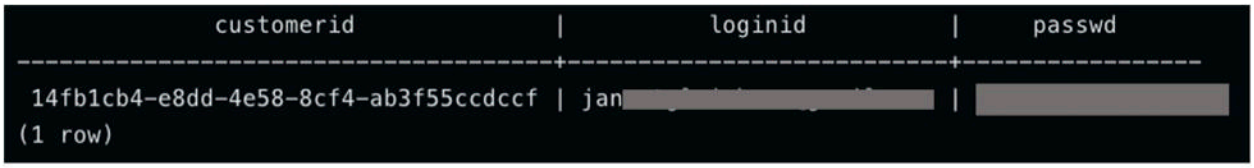

Figure 9. Customer Passwords Stored in the Core-01 database

DinoBank-03	Unencrypted API Communication
Finding Summary	The core Bank API transmits sensitive information in cleartext due to the use of HTTP instead of HTTPS.
Affected Hosts	• 10.0.2.100 (core-01.bank.dinobank.us), port 80
Remediation	Bind the Nodejs app to localhost only and use a SSL reverse proxy to encrypt the traffic.
Reproduction Steps	1. Visit the API URL using a web browser and observe that no encryption is being used.
Additional Resources	• https://cwe.mitre.org/data/definitions/319.html

The core banking API was found to be communicating over the plaintext HTTP protocol. Anyone on the network between a client and the banking API could capture the information that is being transferred. This presents a security risk, due to the sensitive nature of the information being sent to and from the banking API; this information includes Social Security Numbers, transaction information, and other financial data.

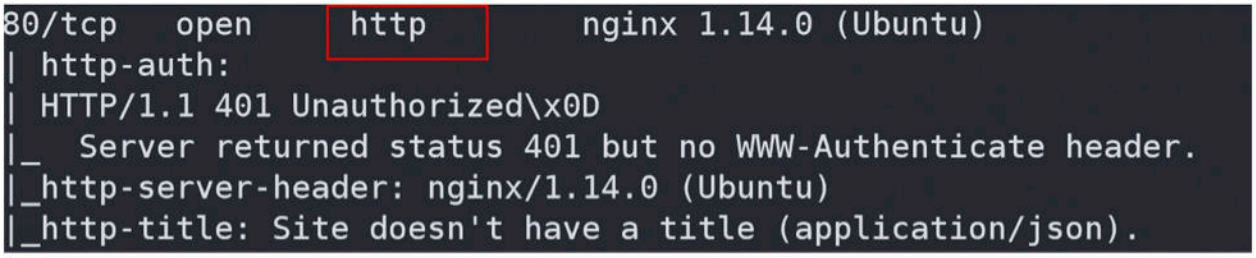

Figure 10. Scan Output Showing the API communicating over HTTP

DinoBank-04	Apache Privilege Escalation Vulnerability
Finding Summary	Apache version 2.4.29 is susceptible to vulnerability CVE-2019-0211 which allows privilege escalation to the root user.
Affected Hosts	• 10.0.2.101 , 10.0.1.33
Remediation	Update Apache to the newest possible version. For this specific vulnerability, it must be at least version 2.4.38
Reproduction Steps	1. Log in to the server and verify the version of Apache that is running on the system
Additional Resources	• https://www.exploit-db.com/exploits/46676 • https://cwe.mitre.org/data/definitions/269.html

The specific version of Apache that was running, is vulnerable to several exploits, including one that would allow privilege escalation to the "root" user. The below screenshots illustrate these findings. They were also vulnerable to CVE-2018-1333 and CVE-2019-0197 which would lead to a denial of service

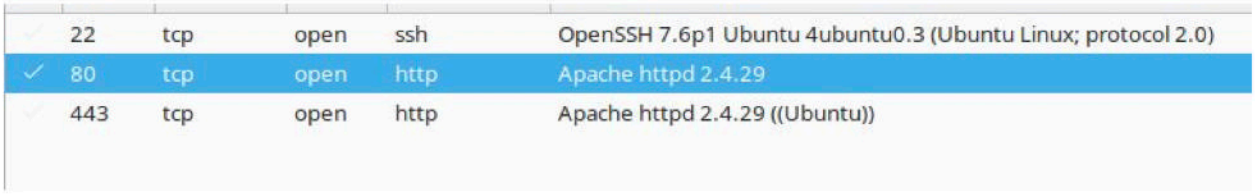

Figure 11. Vulnerable Apache Version

DinoBank-05	Anonymous FTP Access Enabled
Finding Summary	The FTP server allows anonymous access.
Affected Hosts	• 10.0.1.12 (corp-wsus-01.dinobank.us), port 21

Remediation	Ensure that the FTP service is configured to disable anonymous access and restrict FTP user accounts to certain directories such as the user's home directory or only what is needed for a user's job function.
Reproduction Steps	1. Use any FTP client to connect to 10.0.1.12 on port 21 2. Log in using the username "anonymous" and any password
Additional Resources	• https://cwe.mitre.org/data/definitions/284.html • https://www.bu.edu/tech/about/security-resources/bestpractice/ftp/

Anonymous FTP Access was found on the corporate Windows Server Update Services (WSUS) server while running a scan to enumerate the host, as seen in the figures below. This list of available directories shown in the scan indicates that the FTP service has been configured to serve the entire C:\ drive of the host, allowing an attacker unfettered read access to all system files. Because the filesystem is an operating system's primary data store, it often contains sensitive information such as configuration data, source code, hashed and/or plaintext user credentials, along with each user account's files. Direct access to this information is a security vulnerability.

```
root@kali01 ~# ftp 10.0.1.12
Connected to 10.0.1.12.
220-FileZilla Server 0.9.60 beta
220-written by Tim Kosse (tim.kosse@filezilla-project.org)
220 Please visit https://filezilla-project.org/
Name (10.0.1.12:root): anonymous
331 Password required for anonymous
Password:
230 Logged on
Remote system type is UNIX.
ftp> ls
200 Port command successful
150 Opening data channel for directory listing of "/"
drwxr-xr-x 1 ftp ftp              0 Nov 13 23:21 $Recycle.Bin
drwxr-xr-x 1 ftp ftp              0 Nov 13 22:53 Boot
-r--r--r-- 1 ftp ftp         388880 Nov 13 22:48 bootmgr
-r--r--r-- 1 ftp ftp              1 Jul 16  2016 BOOTNXT
drwxr-xr-x 1 ftp ftp              0 Nov 14 06:57 Documents and Settings
drwxr-xr-x 1 ftp ftp              0 Nov 21 18:11 inetpub
-r--r--r-- 1 ftp ftp     1073741824 Nov 23 13:06 pagefile.sys
drwxr-xr-x 1 ftp ftp              0 Nov 13 22:51 PerfLogs
drwxr-xr-x 1 ftp ftp              0 Nov 21 18:18 Program Files
drwxr-xr-x 1 ftp ftp              0 Nov 21 18:17 Program Files (x86)
drwxr-xr-x 1 ftp ftp              0 Nov 21 18:10 ProgramData
```

Figure 12. Scan Output Showing Anonymous FTP Access

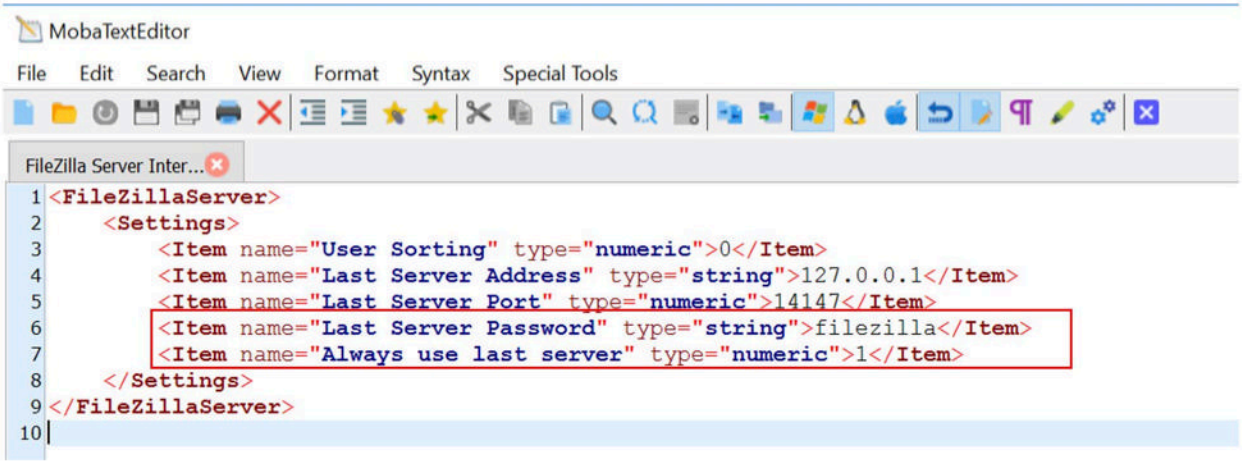

Figure 13. Utilizing the Anonymous Read Access to Find Service Credentials

DinoBank-06	Windows Update Over Insecure Channel
Finding Summary	The WSUS server is not configured to require the use of encrypted communication channels for Windows Updates.
Affected Hosts	• 10.0.1.12 (Corp.WSUS-01.dinobank.us), port 80
Remediation	1. Configure the WSUS server within IIS Manager to use an SSL certificate trusted by the domain. 2. Configure the client machines to use the encrypted version of the site by replacing HTTP with HTTPS in the GPO that specifies the WSUS server. 3. Additional information is provided in the links below.
Reproduction Steps	1. Log in to the server and verify that TLS is not enabled in the IIS configuration manager
Additional Resources	• https://www.blackhat.com/us-15/briefings.html#wsuspect-compromising-the-windows-enterprise-via-windows-update • https://www.contextis.com/en/resources/white-papers/wsuspect-compromising-the-windows-enterprise-via-windows-update

	• https://jackstromberg.com/2013/11/enabling-ssl-on-windows-server-update-services-wsus/

After running a port scan on the remote host, WSUS was revealed to be served over unencrypted HTTP traffic as seen in the figure below. It is possible for a threat actor to intercept network traffic to inject malicious Windows Update patches, due to the lack of encryption, to compromise hosts on the network.

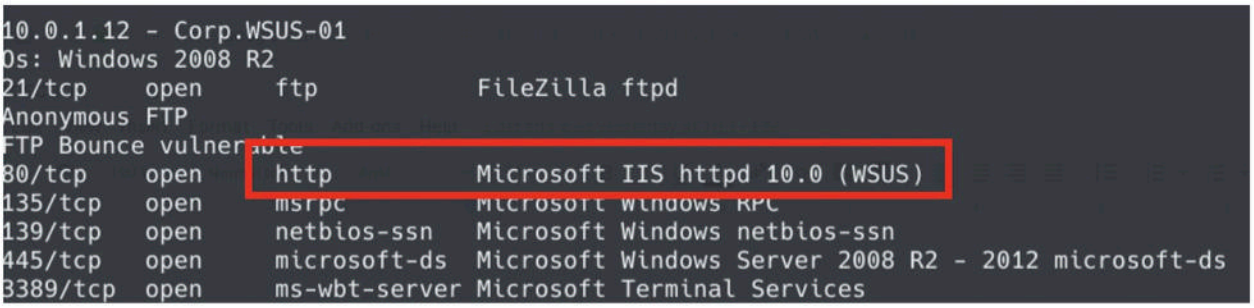

Figure 14. Port Scan - WSUS being served over unencrypted HTTP

DinoBank-07	Sensitive Information in Error Message
Finding Summary	The core banking application account registration page, located at https://my.dinobank.us/register.php displays the contents of the password field in an error message when registering a new user.
Affected Hosts	• 10.0.2.101 (bankweb-01.bank.dinobank.us), port 80,443
Remediation	Any error messages should not contain input supplied by the user, particularly sensitive information such as passwords. If an error must be generated, describe the allowed input instead of displaying the input that caused the error back to the user.
Reproduction Steps	1. Browse to the account registration page. 2. Using an example password, register a user and the contents of the password field will be shown in an error message on the page following submission of the form.

Additional Resources	• https://cwe.mitre.org/data/definitions/522.html

In general, passwords should be protected at all times. Evidence of this practice can be seen whenever a password field on a website uses dots to represent the characters, instead of in plaintext. A situation could arise where a user is registering an account for this site and someone looking over their shoulder would see the password they entered. In addition, the Social Security Number was also shown in plaintext in this form.

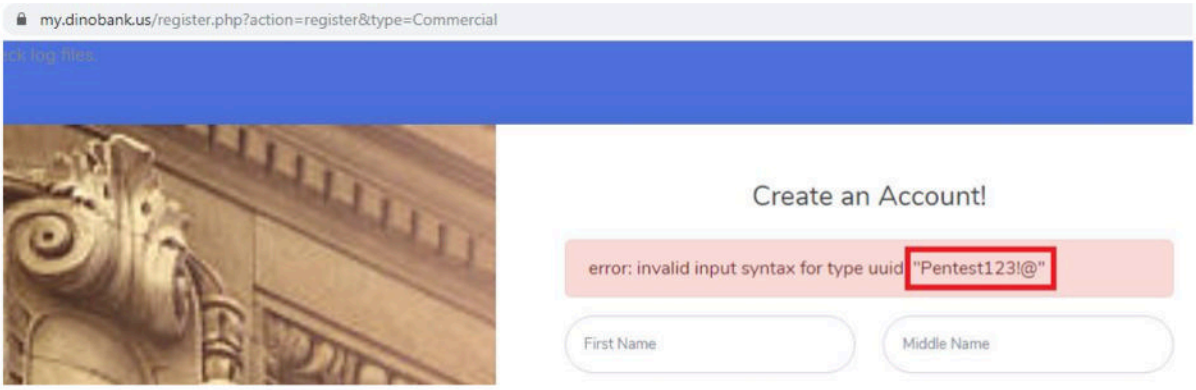

Figure 15: Contents of the Password Field Shown in Plaintext

DinoBank-08	Reflected Cross-Site Scripting
Finding Summary	The core banking application account registration page, located at https://my.dinobank.us/register.php is vulnerable to reflected cross site scripting.
Affected Hosts	• 10.0.2.101 (bankweb-01.bank.dinobank.us), port 80,443
Remediation	Ensure that untrusted or unfiltered data is not included within the code of any HTML page. User input should be sanitized before being included in any page.
Reproduction Steps	1. Browse to the account registration page 2. After filling in the other fields in the registration form, place a script tag in both password fields 3. An example payload is <script>alert(“XSS”)</script>
Additional Resources	• https://cwe.mitre.org/data/definitions/79.html

	• https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

A reflected cross-site scripting vulnerability was discovered on the DinoBank online banking application during a manual review of the web application's security. The account registration page on the site was found to display the contents of the password field in an error message when attempting to register a new account. The error details are not sanitized before being added to the page, which allows a client to inject a script into a request and have the server return the script to the client in the response. This could be used to generate a malicious link which appears to be the official DinoBank site but could steal the victim's banking information. The figure below shows an example of the reflected cross-site scripting, where a script is injected into the page that creates an alert on the page containing the text "XSS".

Figure 16. Reflected Cross-Site Scripting Payload Insertion

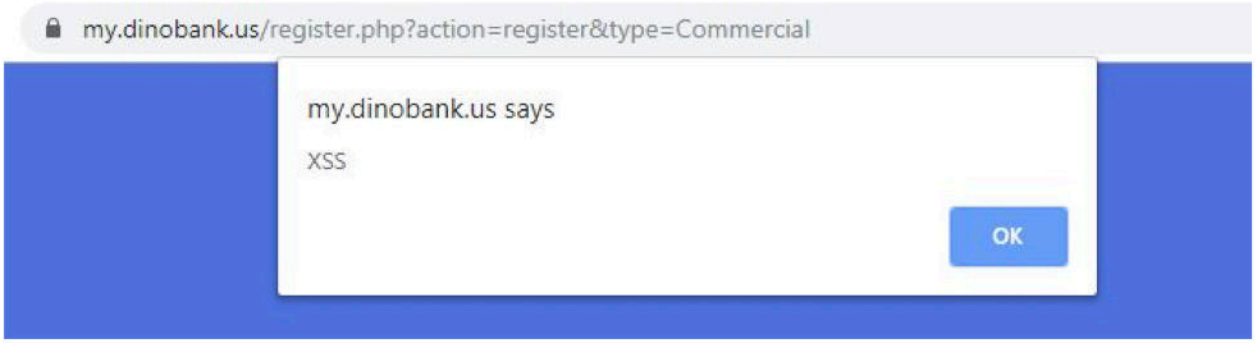

Figure 17. Reflected Cross-Site Scripting on Core Banking Site

DinoBank-09	API Keys Stored in Logs
Finding Summary	The Core Banking API write sensitive information, including API keys, to log files.
Affected Hosts	• 10.0.2.100 (core-01.bank.dinobank.us), port 9001,443
Remediation	Modify code to redact most, if not all, of the API key in the logs.
Reproduction Steps	1. View logs generated by the banking api and look for the string “accepted request for API Key”
Additional Resources	• https://cwe.mitre.org/data/definitions/532.html

The storage of sensitive information in logs was found during a code review of the Core Banking API. Sensitive Information written to log files can give valuable guidance to an attacker or expose credentials.

```
// TODO: support for JWT etc. in future versions
if (authType == 'ApiKey') {
  redisClient.get('apikey:legacy', function (err, keyValue) {
    if (err) {
      throw err
    } else {
      if (keyValue === authValue) {
        console.log('accepted request for API Key: ' + authValue)
        next()
      } else {
        console.log('rejected request for API Key: ' + authValue)
        returnAuthError(res, 'Invalid Token')
      }
    }
```

Figure 18: Contents of the Password Field show in plaintext

DinoBank-10	FTP Bounce
Finding Summary	The FTP server allows for the use of the PORT command even when the host referred to is not the client.
Affected Hosts	• 10.0.1.12 (corp-wsus-01.dinobank.us), port 21
Remediation	Ensure that the FTP service is reconfigured to disable anonymous access and ensure that the FTP PORT command is either properly scoped or disabled.
Reproduction Steps	1. Run the command: 'nmap -Pn -b 10.0.1.12 google.com'
Additional Resources	• https://cwe.mitre.org/data/definitions/441.html • http://www.ouah.org/ftpbounce.html • https://nmap.org/book/scan-methods-ftp-bounce-scan.html

A lack of proper configuration on the file transfer service was found on the Windows Server Update Service (WSUS) server while running a scan to enumerate the host. This allows an attacker to use the PORT function on the FTP service to scan additional networks through the FTP service.

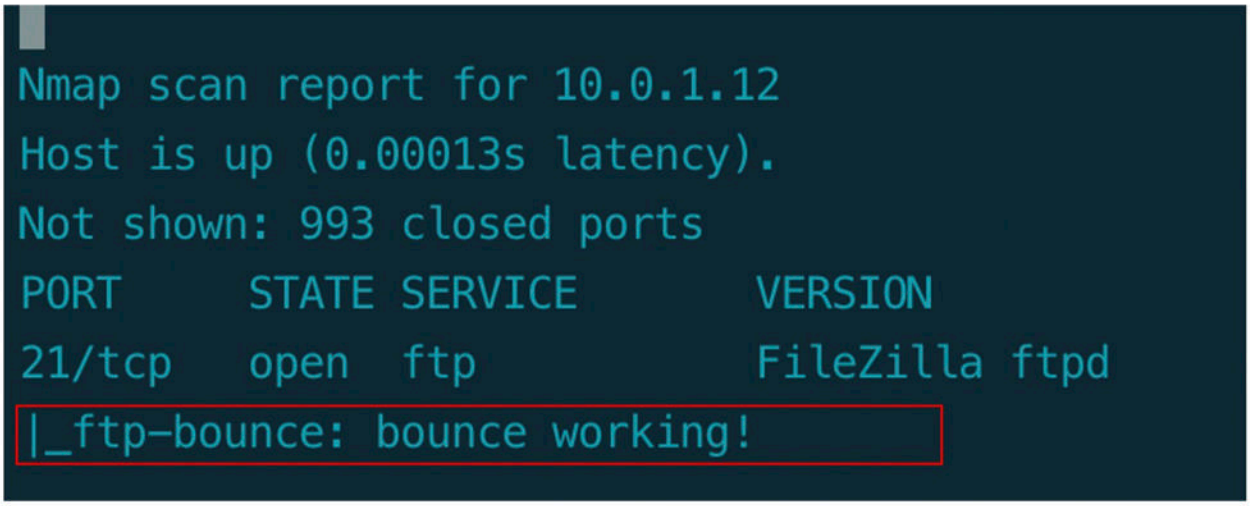

Figure 19. Nmap Showing FTP Bounce Working

The figure below demonstrates scanning the 10.0.10.203 system by “bouncing” through the misconfigured 10.0.1.12 FTP server.

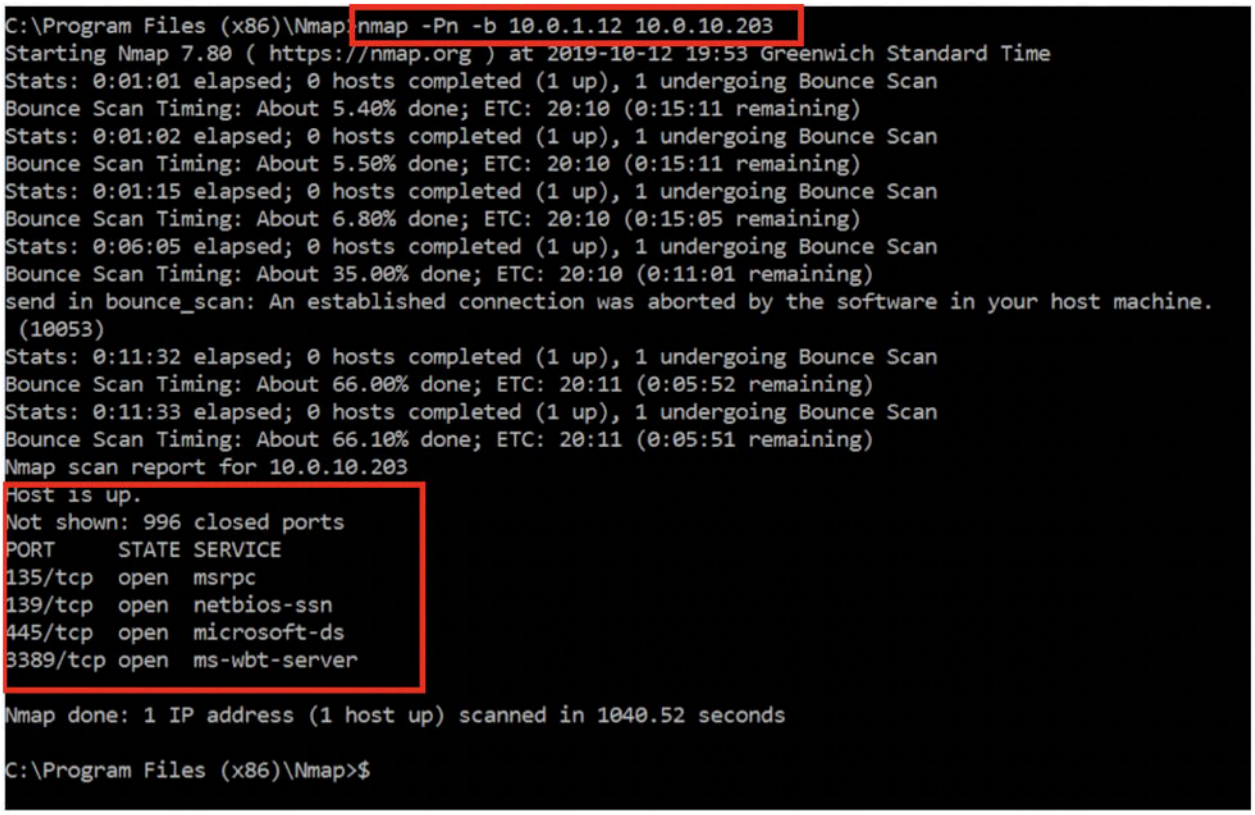

Figure 20. FTP Bouncing Through 10.0.1.12

DinoBank-11	Outdated QueryTree Version
Finding Summary	QueryTree version 0.0.0.0 is out of date. The most recent version of the QueryTree application is 3.0.97.
Affected Hosts	• 10.0.2.103 (REPORTS-01.bank.dinobank.us), port 80
Remediation	Ensure the QueryTree application is updated to the newest possible version.
Reproduction Steps	1. Login to the Administrative portal of the QueryTree application and verify the currently installed version.
Additional Resources	• https://cwe.mitre.org/data/definitions/937.html • https://attack.mitre.org/techniques/T1190

The QueryTree service was running an outdated version of the application. This specific version of QuertyTree needs to be updated to at least version 3.0.97. The figure below shows the version of QueryTree that was running.

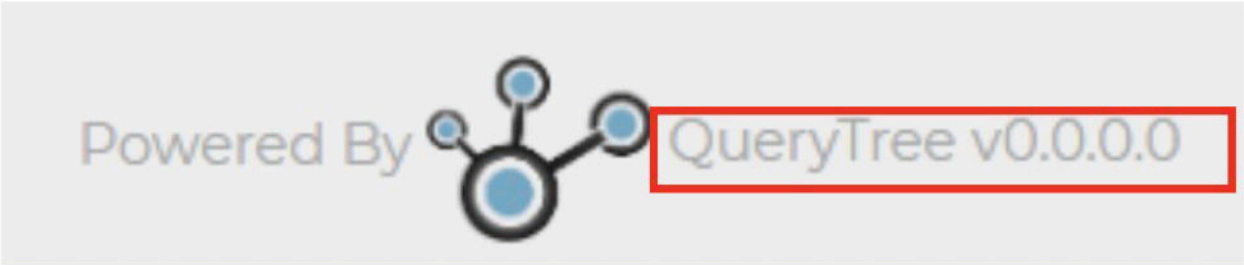

Figure 21. Outdated QueryTree Version

DinoBank-12	Plaintext Credentials in Configuration File
Finding Summary	The core banking API's configuration stores username and passwords in plaintext
Affected Hosts	• 10.0.2.100 (Core-01.bank.dinobank.us), port 5432
Remediation	Ensure industry best practices are followed for API connection details.
Reproduction Steps	1. Access postgres database server 2. Read the contents of /data/web/api/config/development.json
Additional Resources	• https://cwe.mitre.org/data/definitions/260.html • https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/protecting-connection-information

The API service running on the DinoBank database server contained statically set credentials in a configuration file that was readable by any user. This can be used to view configuration details which should be encrypted. The figure below shows the connection details which contains the username, password, IP address, and port for the PostgreSQL database. The configuration file also included credentials for the Redis server, which allowed us to access the key-value store containing API keys.

Figure 22. Hardcoded Credentials In The Configuration File

Figure 23. Reading API Key from Redis Server

The API key acquired from the Redis key-value store could be used to request sensitive customer information from the core banking API, as seen in the figure below.

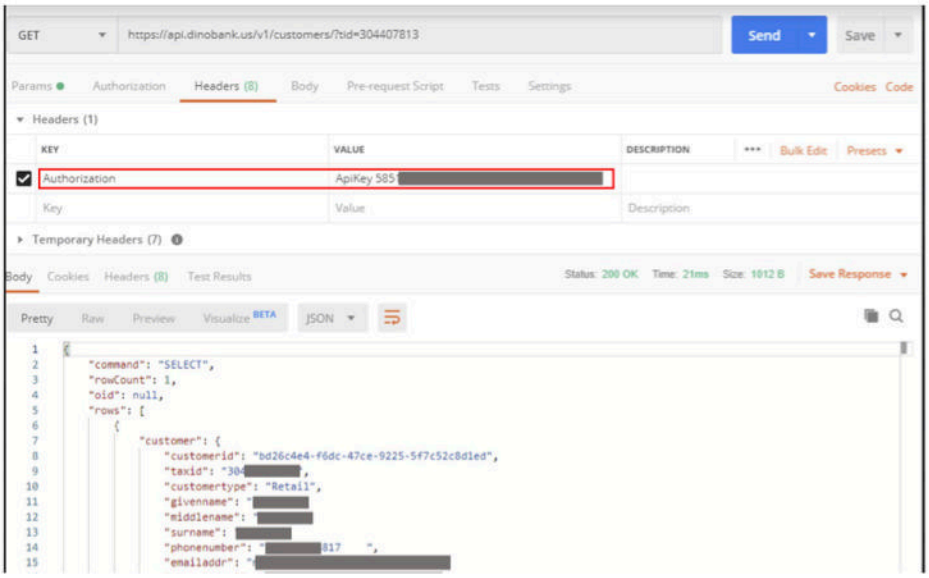

Figure 24. Core Banking API Access

DinoBank-14	Execution with Unnecessary Privileges
Finding Summary	The NodeJS API is running under the context of the root user.
Affected Hosts	• 10.0.2.100 (core-01.bank.dinobank.us), port 80
Remediation	Reconfigure the API to run under the context of a non-privileged user.
Reproduction Steps	1. Run the following command: “ps aux | grep node” and take note of which user the node API is running as.
Additional Resources	• https://cwe.mitre.org/data/definitions/250.html

The core banking API was found to be running under the context of the root user while performing post-exploitation activities on the system. The figure below shows the output of the “ps aux” command, which was used to list information about the NodeJS process. Running applications with unnecessary permissions increases the access that attackers could have if they are able to successfully exploit the application.

```
root     21407  0.0  0.0 108128  7400 ?        Ss   16:32   0:00 sshd: postgres [priv]
root     32430  1.5  1.2 941568 93696 ?        Ssl  14:55   4:03 node /data/web/api/index.js
root     32449  0.0  0.0      0     0 ?        I    18:41   0:00 [kworker/u4:0]
```

Figure 25. Output of ‘ps aux’ Command

DinoBank-15	Information Exposure
Finding Summary	Multiple pages contain debugging or error information that is overly verbose.
Affected Hosts	• 10.0.1.250 (corp-coins-01.corp.dinobank.us), port 80
Remediation	Disable debugging output in production and ensure that proper error handling is implemented without revealing internal details

Reproduction Steps	1. Attempt to login to the at http://10.0.1.250 using a user account that does not exist (for example, the username “user”) 2. Compare the error received with the error received when attempting to log in with an existing user account (for example, the username “alejandra”)
Additional Resources	• https://cwe.mitre.org/data/definitions/200.html • https://www.owasp.org/index.php/Improper_Error_Handling

Overly verbose error information was found on the OpenTrade cryptocurrency exchange while assessing the security of the web application. Overly verbose error information can disclose important information that can be used by attackers. In the case of this server, the errors returned when attempting to log in can be used to enumerate existing user accounts on the application.

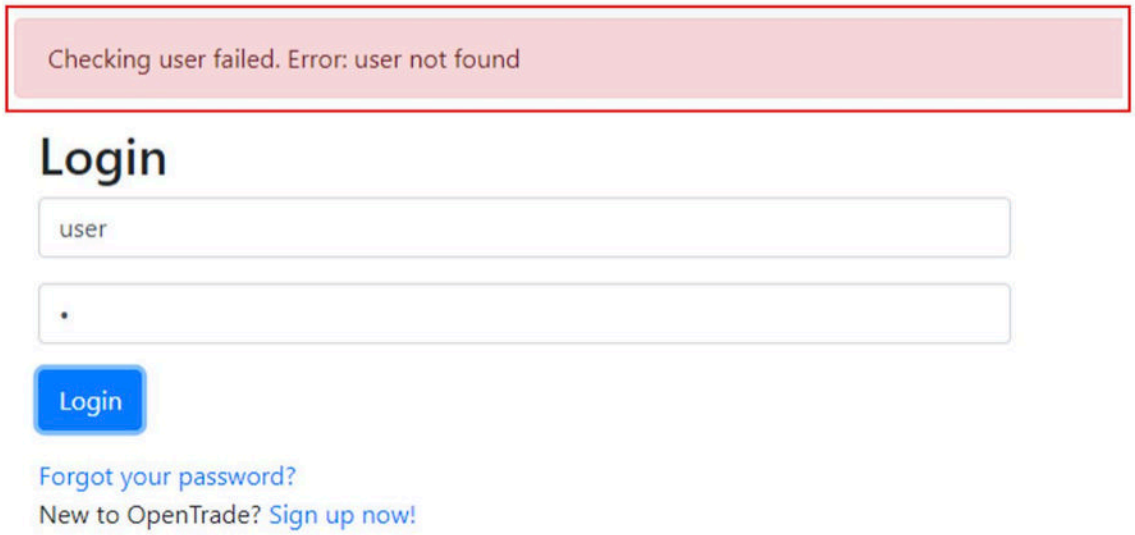

Figure 26. Error Message Revealing User Account Existence

DinoBank-16	Unnecessary Services
Finding Summary	Multiple services exist on the network which is not necessary for business operations.
Affected Hosts	• 10.0.1.250 (corp-coins-01.corp.dinobank.us), port 80 • 10.0.2.113 (heads-01.bank.dinobank.us), port 80 • 10.0.2.115 (coin-tails-01.bank.dinobank.us), port 80
Remediation	Maintain an up-to-date inventory of all services running on the corporate network. Periodically evaluate the business criticality of these hosted services, and swiftly decommission any services that are no longer necessary.
Reproduction Steps	1. Visit http://10.0.1.250 in a web browser 2. Visit http://10.0.2.113 in a web browser 3. Visit http://10.0.2.115 in a web browser
Additional Resources	• https://attack.mitre.org/mitigations/M1042/ • https://security.berkeley.edu/no-unnecessary-services-guidelines • https://www.owasp.org/index.php/Minimize_attack_surface_area

Unnecessary services were found on the DinoBank networks throughout the enumeration phase of our engagement. These services are more likely to be forgotten, remain un-maintained, and increase the attack surface of the network.

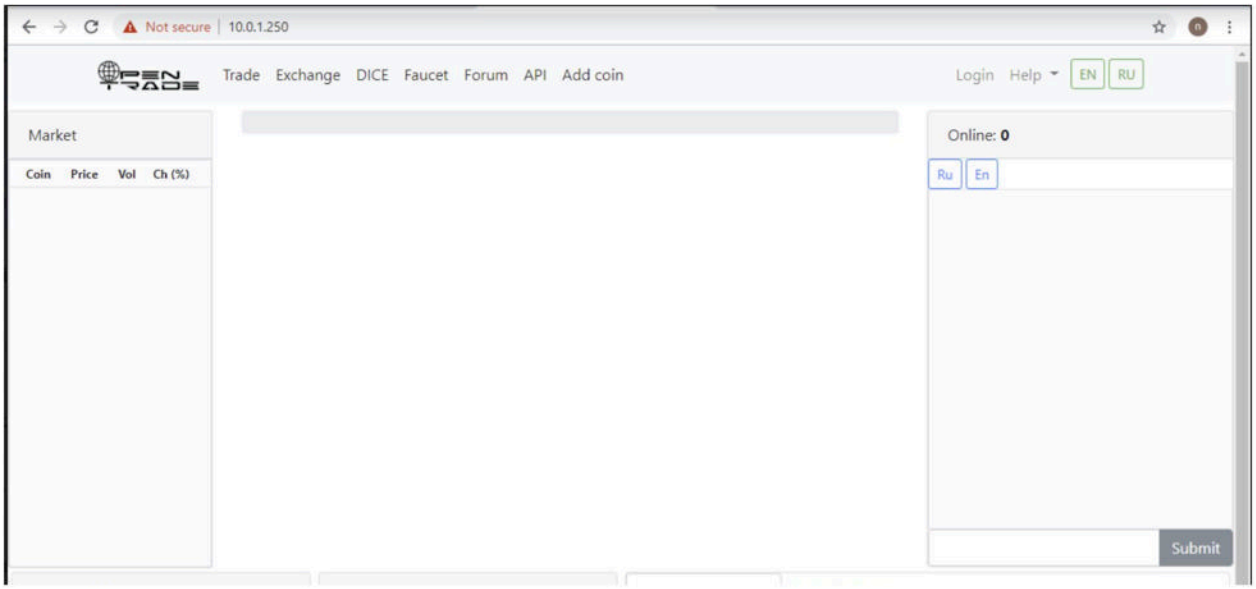

Figure 27. Unnecessary Cryptocurrency Exchange Service

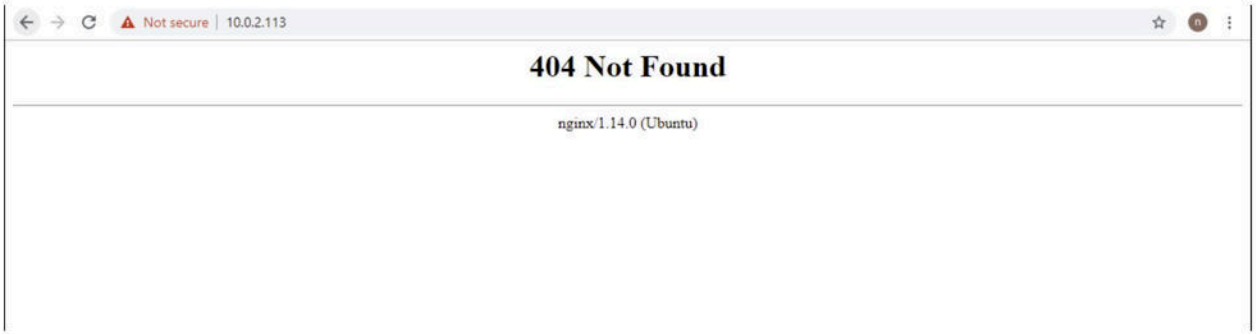

Figure 28. Unnecessary NGINX Server

Figure 29. Unnecessary NGINX Server

Interactive Voice Response (IVR) Security

IVR security risks involve business logic flaws or social engineering-related vulnerabilities, so programming in the necessary monitoring capabilities and logic checks to prevent abuse is essential. Any system that interacts with end-users needs to be monitored for suspicious behavior. To test the end to end flow of IVR system, a tester needs to dial the same number and route through different menu options which become more complex as the tester goes deeper into testing. We used a toll-free number to call and test the IVR system. We observed the data being transferred to an unencrypted endpoint on 10.0.2.100 and found the IVR system sends credentials and TaxIDs (social security numbers) in plain text.

Automated Teller Machine Security

Automated Teller Machines are trusted with significant amounts of money and customer data. The security of these devices is of the utmost importance. While evaluating the security of the ATMs, we noticed several issues chief of which was how outdated the atm was at the time of testing, we also noticed that an atm maintenance staff came by and on his way out left the atm unlocked on the admin panel. The pin requested by the atm did not seem to affect the success of a balance check or withdraw and the encryption used by the keypad is considered outdated.

Recommendations

It is imperative that remediation efforts are timely, precise, and efficient. The following serves as a remediation roadmap to significantly reduce overall risk within the DinoBank network. The first action we recommend, is a corporate-wide password reset on all accounts and devices, this should be done immediately after securing the core banking database in finding DinoBank-01.

Critical Remediations (As soon as possible)

- Implement proper access control on the Core-01 database.

Short-term Goals (0-6 Months)

- Rotate credentials across all networked hosts and services, including an enterprise-wide password reset for user accounts.
- Patch critical software, specifically those with easy to exploit issues
- Ensure that all sensitive data is encrypted at rest and in transit according to relevant regulations.
- Require authentication for all services on the network that handle sensitive information or have the potential to be abused.
- Restrict access to critical databases and applications to only people who need it for their job function.
- Reconfigure any FTP services to prevent proxying and disable anonymous login.
- Develop a robust offline backup strategy for business-critical systems.

Long-term Initiatives (6-12 Months)

- Implement a company-wide vulnerability discovery and patch management policy.
- Evaluate the business criticality of all available systems and decommission those that are unnecessary.
- Implement and enforce security awareness training for all employees with a specific focus on access control, database security, and password security.
- Consistently review network infrastructure to identify and retire end-of-life hardware, software, and protocols.
- Implement the principle of least privilege for all systems.
- Implement MFA on business-critical machines such as the domain controller.
- 3rd party verification of fixes laid out in this document.

Appendices

Appendix A. Network Diagrams

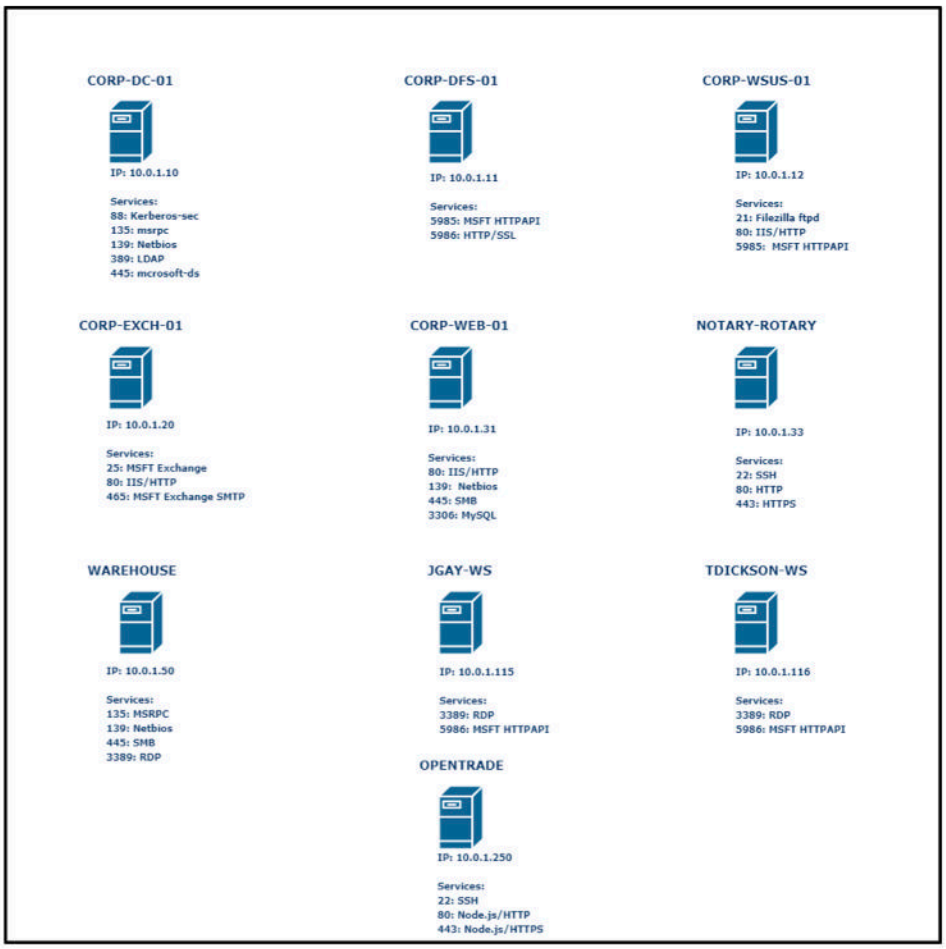

Figure 30. Corporate Network Diagram

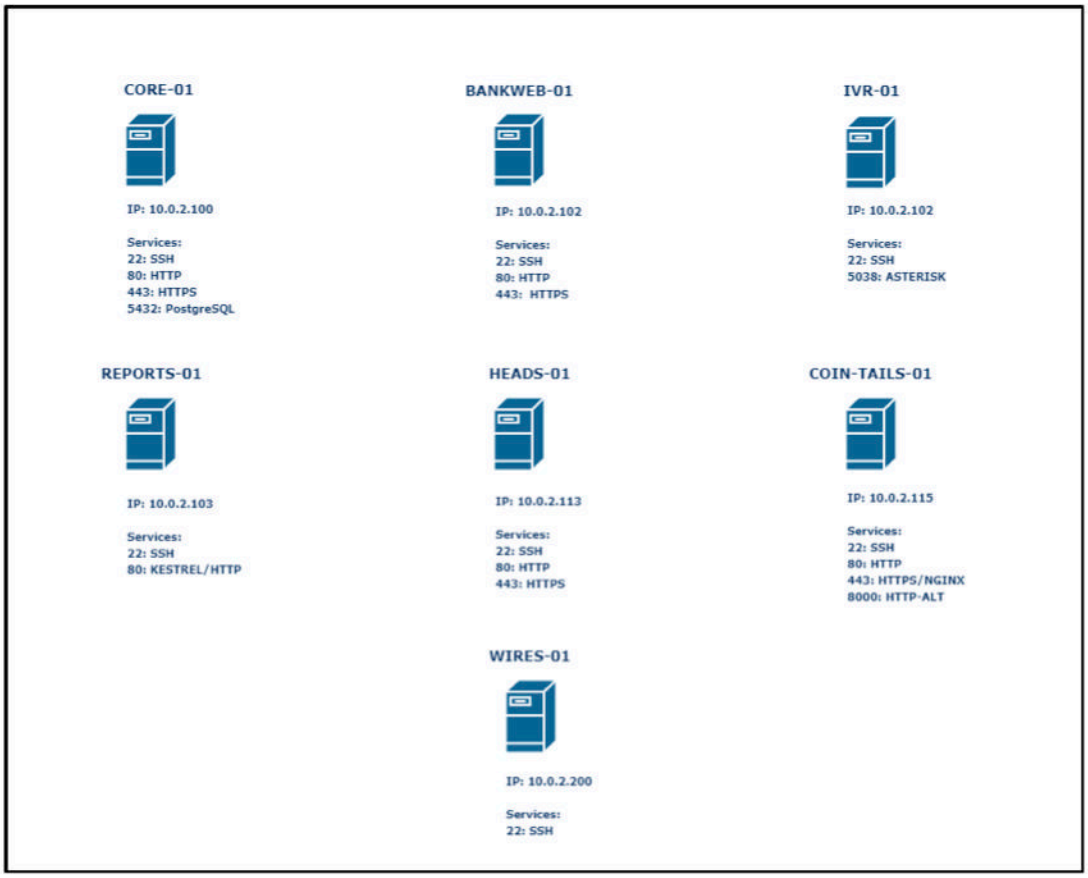

Figure 31. Bank Network Diagram

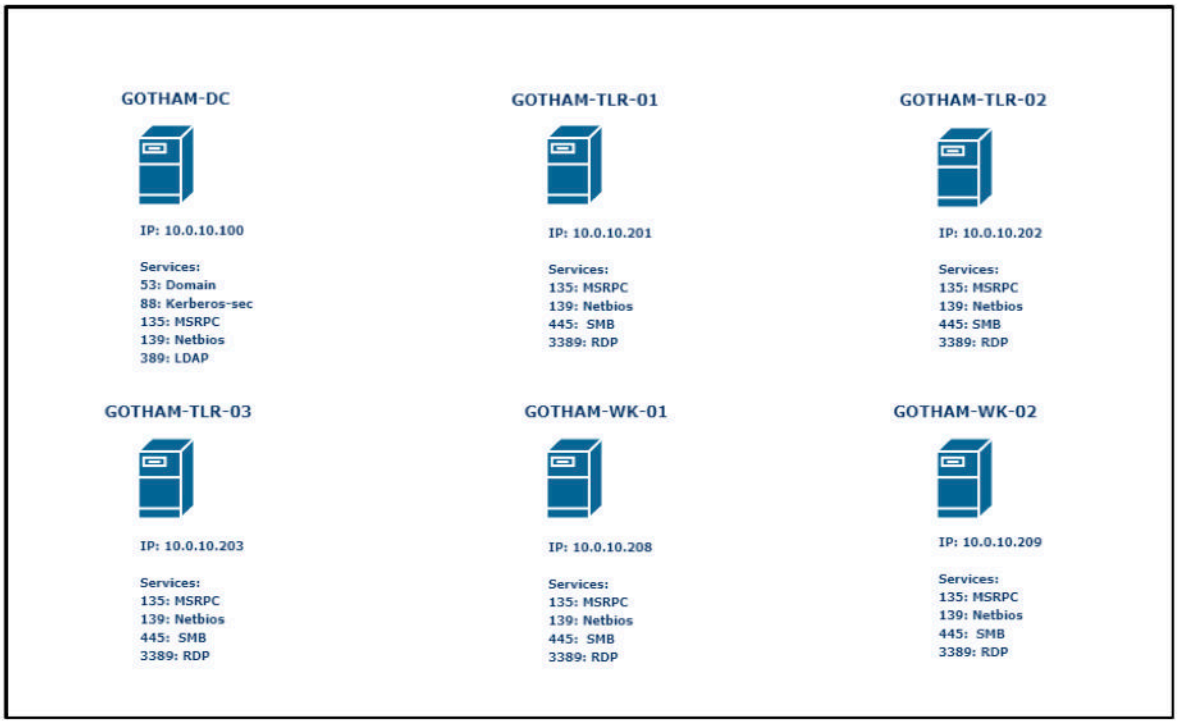

Figure 32. Gotham Branch Network Diagram

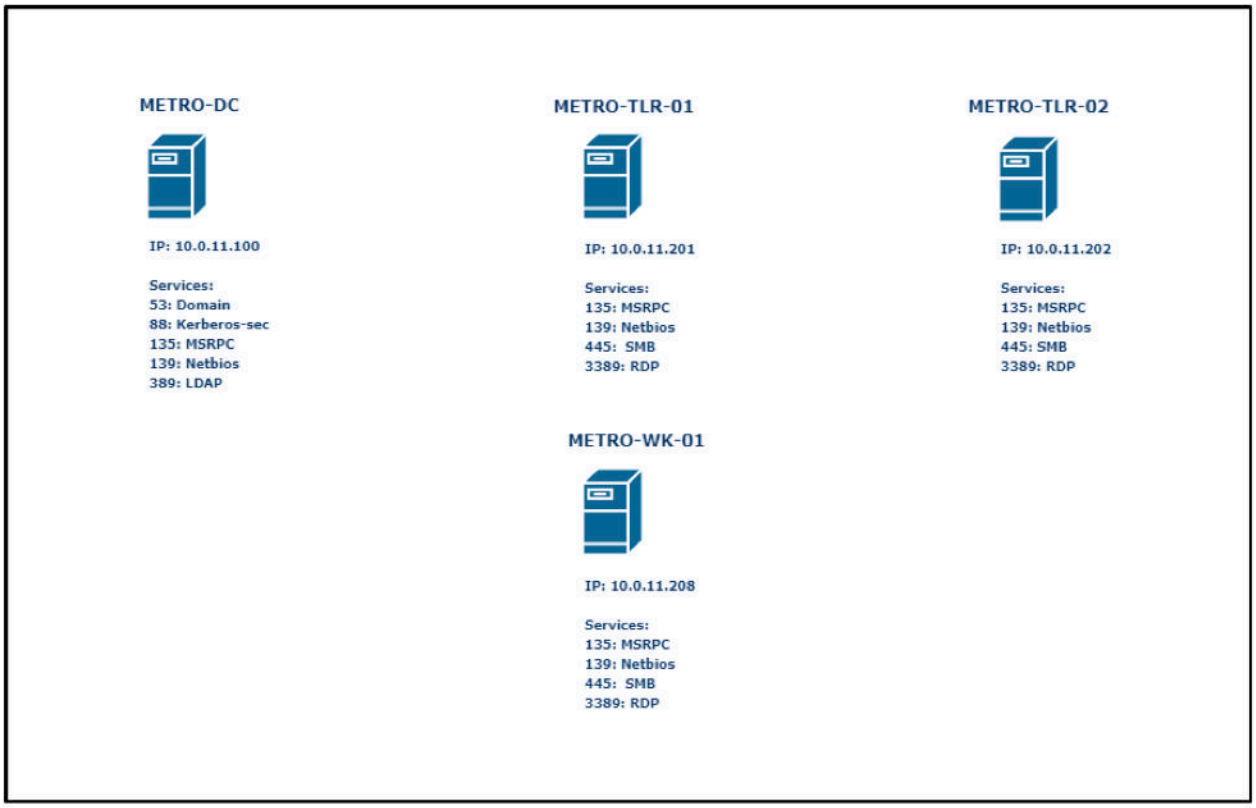

Figure 33: Metro Branch Network Diagram

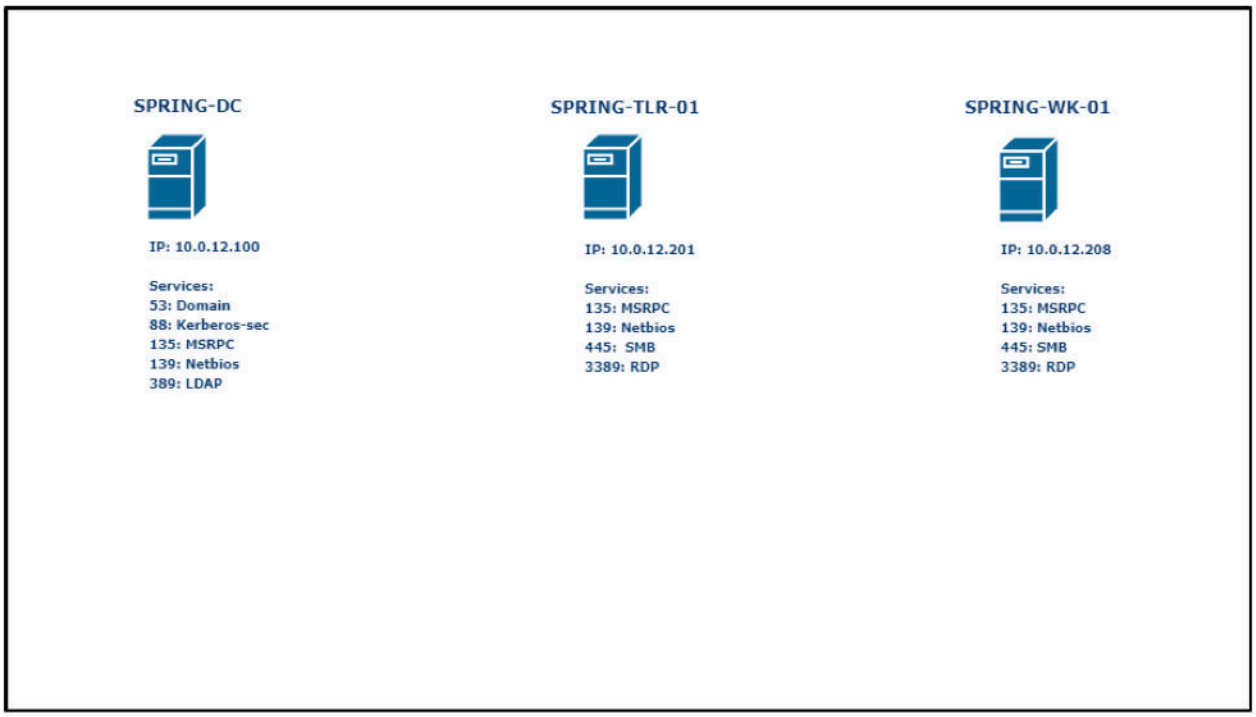

Figure 34: Spring Branch Network Diagram

Appendix B. Smart Contract Audit

A possible vulnerability in a DinoBank smart contract was discovered. The contract CroissantCoin is vulnerable to several risks detailed below.

Contract details:
https://etherscan.io/address/0xb940c3b8a8a27a53dc10a5ecd9cc4d2fcd798892#code

The compiled contract might be susceptible to SignedArrayStorageCopy (low/medium-severity), ABIEncoderV2StorageArrayWithMultiSlotElement (low-severity), DynamicConstructorArgumentsClippedABIV2 (very low-severity), UninitializedFunctionPointerInConstructor_0.4.x (very low-severity), IncorrectEventSignatureInLibraries_0.4.x (very low-severity), ExpExponentCleanup (medium/high-severity), EventStructWrongData (very low-severity), NestedArrayFunctionCallDecoder (medium-severity) Solidity Compiler Bugs.

Figure 35. Solidity Compiler Bugs

Vulnerability	Severity	Description	Line
Integer Overflow	High	Safe addition operators are not always used and therefore overflows might be possible. The use of assertions might be useful to catch potential errors.	126,127
Dependence on predictable environment variable	Low	The block.timestamp environment variable is used to determine a control flow decision. These values can be manipulated by malicious miners.	219,221
Potential denial-of-service if block gas limit is reached	Low	A message call is executed in a loop. This can cause a transaction to fail if the loop has no bounds and the contract runs out of gas.	N/A

Figure 36. Findings from a smart contract audit

Appendix C. OSINT Findings

During the open-source intelligence gathering phase of the engagement, an SSH key was discovered in a public repository owned by a DinoBank employee. Although we were unable to successfully gain access to any DinoBank systems with this credential throughout our engagement, it could potentially be used to access other systems or services. DinoBank should implement company policies about posting sensitive information on public channels. The SSH key can be found in this commit.

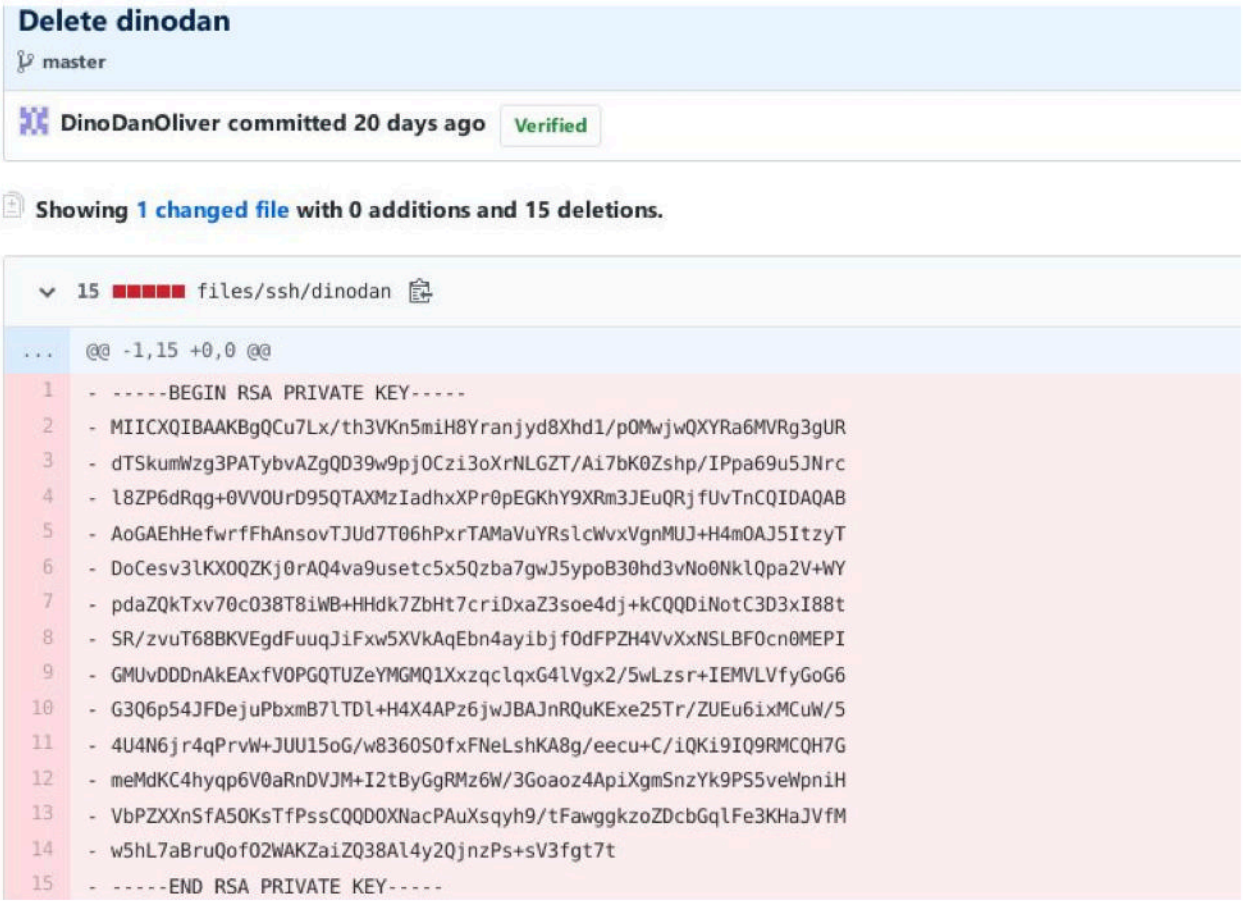

Figure. 37 Exposed Key

We also found a discussion on Reddit of previous passwords in the post employees state that “Password1” was previously used on all computers this “reddit” post can be found here https://www.reddit.com/r/DinoBank/comments/dnobvo/wow_found_some_instances_of_password1_on_our/. While according to our audit it was not on any system this is still considered bad practice in case the team missed some machines with the old passwords.

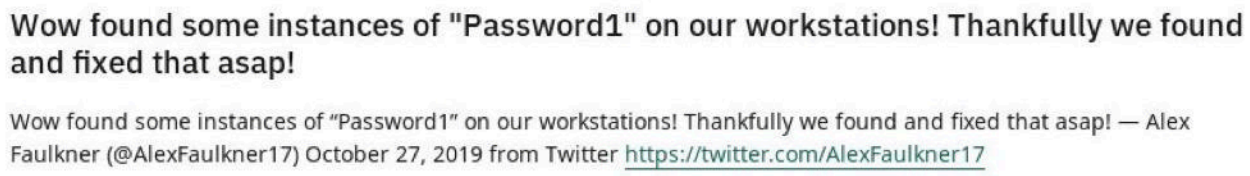

Figure. 38 Oversharing Post