Thank you for taking the time to review this sample report! More about our services can be found at: https://penconsultants.com/services.

Please let us know if you are interested in learning more, have questions, or are ready for us to serve your security testing needs.

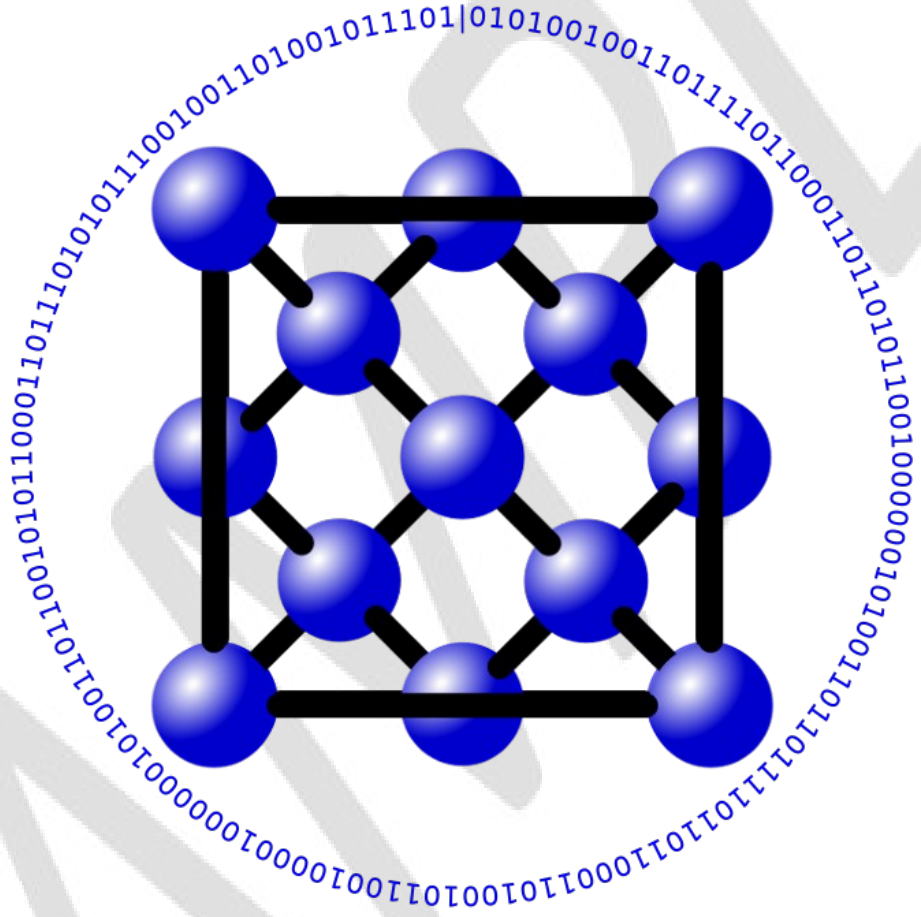
Robert Neel
Founder | Owner, PEN Consultants, LLC
PEN Consultants, LLC
Phone: 830-446-3411
Email: info@PENConsultants.com

DISCLAIMER: This is a sample of a Findings and Recommendations Report which is representative of what we deliver to our clients after testing.

- Examples shown are from various engagement types: web application penetration testing, network penetration testing, physical social engineering, phishing assessments, wireless assessments, etc. A single engagement would typically not cover such a breadth. As such, some sections may/may not be present for a given engagement.
- The amount of information given in this sample report is limited, as details would be specific to each client/environment. Some entire sections are blank, as those are unique per client. Likewise, the typical number of screenshots, example attacker code, specific recommendations, etc. are limited.
- When possible, we've used excerpts from actual delivered reports from the last nine years, removed the sensitive text with "[REDACTED]", or similar, so the reader can get a better sense of what actually gets delivered to a client.
- Given that these examples are from reports as far back as nine years ago, certain vulnerabilities and attack vectors mentioned may no longer be possible.
- Although there is no guaranteed minimum or maximum, on average, we run about 18 findings and recommendations per engagement/report (median 19, standard deviation 9), ranging from "informational" to "critical".

PEN Consultants

Information & Cybersecurity Testing Services



Security Testing

Findings and Recommendations Report

[CLIENT]

[DATE]

[CLIENT NAME],

Your organization requested PEN Consultants to perform security testing against one or more of your applications, systems, networks, or facilities to assess your current security posture. We are honored to have been trusted to serve your organization with your information and cybersecurity needs.

Please read through this document in its entirety, and return it to us at your earliest convenience with any questions or requested changes. Alternatively, a conference call can be setup to work through the document.

A detailed explanation of our services can be found at https://PENConsultants.com.

The general timeline for the engagement is as follows:

1. Phase 1 - Pre-testing
    1. Request by client, initial phone/email discussions, etc. (COMPLETE)
    2. Mutual Non-Disclosure Agreement (COMPLETE)
    3. Discuss and define scope, goals, objectives, etc. (COMPLETE)
    4. Preliminary Service Quote (COMPLETE)
    5. Scoping Questionnaire (COMPLETE)
    6. Service Contract and Statement Of Work (SOW), to ensure all parties understand what our service provides (COMPLETE)
    7. Kick-off document, with initial change requests needed prior to scheduling testing (COMPLETE)
2. Phase 2 - Testing
    1. Testing and evaluation, as defined in the SOW (COMPLETE)
    2. Once all objectives above are met, we'll coordinate with you to go into the "getting noisy" phase (if applicable/requested) (N/A)
3. Phase 3 – Reporting
    1. Findings and Recommendations Report describing vulnerabilities discovered, attack vectors confirmed, and steps to mitigate and/or detect (THIS DOCUMENT)
    2. Follow-up to review/ refine report
    3. Deliver the final report in fulfillment of the SOW, send invoice, and schedule exit interview
4. Phase 4 – Post-Testing
    1. Optional phases - can be purchased/added at anytime:
        1. Brief the report to IT support staff, leadership, or 3rd parties
        2. Assist IT support staff in implementing and verifying recommended mitigations
        3. Post remediation testing and updated reporting
        4. Begin Cybersecurity Unlimited service, based on SoW
        5. Schedule quarterly testing

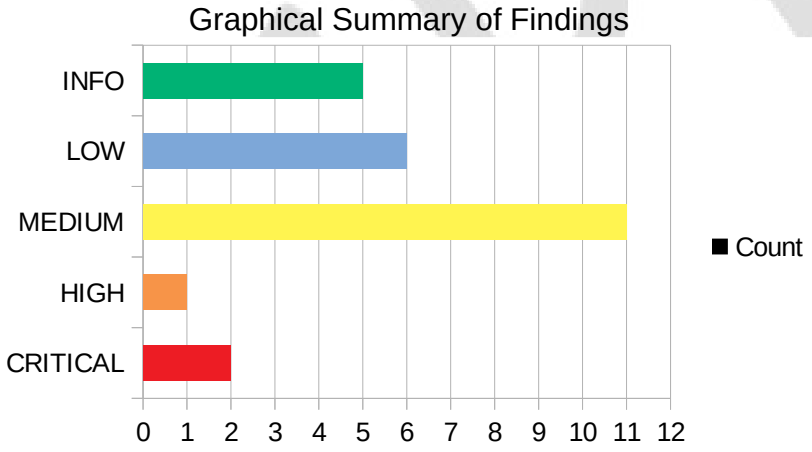
Table of Contents

Executive Summary

During the month of February 2018, PEN Consultants performed a Web Application Penetration Test of the [NAME OF WEB APP] and the supporting infrastructure ("web app"). The objective of the testing was to find gaps that would cause a violation of Confidentiality, Integrity, or Availability of data or systems.

It is evident the web application developers and IT staff understand and implement basic security standards. Hopefully this report will reassure [CLIENT NAME] leadership about their security stance, while at the same time, highlight areas of improvement.

It is impractical for any system or software to be 100% secure from compromise. What sets organizations apart is best-effort security from the get-go, followed by a desire to seek out both internal and external review for overlooked issues and a responsiveness to address them. [CLIENT NAME] excels in all of these areas. It is a privilege to work with such a great organization.



The overarching recommendations are:

- **Top Recommendation - Session Management and Access Control:** Strengthen the authentication processes to prevent undetected brute force user enumeration, password compromise, and account take over (ATO), improve session management to prevent hijacking, tighten access control of cloud resources to prevent unauthorized users from accessing confidential data, and overhaul the access controls between users, roles, and tenants to prevent an attacker from compromising data at-will.
- **User Input Validation:** Implement more robust user input sanitization, validation, and enforcement for user input to prevent back end compromise through SQL injection (SQLi), user compromise through XSS, ATO due to poor username and password policies.
- **Configuration Management:** Improve configuration management which will resolve many of the issues discovered, reduce the likelihood of attacks demonstrated in this report, help mitigate an infrastructure breach, and minimize the impact should a breach occur.
- **Additional & Continued Testing & Training:** Based on observations made during this engagement, PEN Consultants recommends continued testing, monthly developer training, internal and external network penetration testing, as well as testing of other web applications. More details can be found on page X.

Introduction

Commendations

During a typical penetration test, PEN Consultants performs tens of thousands of automated, semi-automated, and manual tests and attacks based on our proprietary testing methodology, which is a combination of industry leading standards (largely the OWASP testing guide), tools, and our personal experience and knowledge - all of which can be viewed in the testingGuide-TOC document on Box. Although we are not legally allowed to use all of the same techniques a real attacker would, the vast majority of the ones we were able to use were not successful in breaching in-scope systems from an external, unauthenticated source.

In addition to the tests that validated as secure, we would like to highlight the following items which, historically, are weak areas, but are areas in which CLIENT excelled.

- White Box Testing: Approximately only half of our clients opt for full white box testing on the first engagement. CLIENT granted us nearly full access and visibility; there was no request denied. We were treated and trusted as much as a senior IT employee with administrative accesses, to include access to a domain admin account, access to all alerting and logging, we were frequently asked if we would like additional access before we even had a chance to ask first, access to some external facing services that normally have restricted access through firewall policy, back end access to the AWS infrastructure (i.e readonly), etc. Not only did CLIENT benefit from a tenfold increase in testing thoroughness because of this, but it also speaks volumes about CLIENT IT staff and their strong pursuit for the utmost security. It can truly be said, CLIENT IT staff care deeply about the security of the network and data.
- Firewall: The firewalls used and configurations set made initial scanning impractical to perform at speed. Within seconds of beginning those scans, CLIENT firewalls automatically classified our IP as a threat and temporarily blocked our access. Although this did not detect our slower, more targeted, scanning, it is, nonetheless, an impressive capability which is rare to see and one that will protect against some of the more common attackers. Note: CLIENT approved our request to disable this feature for our IPs so that testing could continue at a fast, yet thorough, pace.
- CLIENT'S network segmentation and access control is among the best we have seen. Many systems that we tested on the internal network may have contained vulnerabilities, but the network access controls make these systems much more difficult to compromise. This is a sign of a strong security posture, as this demonstrates a defense-in-depth approach to security throughout the internal network.
- External, 3rd party service configurations limited access to sensitive resources such as Azure AD, and Microsoft Office 365, even though we gained access to those services

with compromised user accounts. This is another area in which we rarely find such restrictions.

- MFA on Endpoints: Even in large, fortune-500 companies, it is rare to see an MFA requirement on workstation and server access. Although there are certainly ways to get around this feature, it raises the level of security. We were very pleased to see this.
- SMB Shares: Most internal penetration tests reveal numerous SMB misconfigurations, allowing anyone to access sensitive information on file shares - shares containing service account passwords (ex. in scripts) and user passwords (ex. in documents). We found no immediate concerns in this commonly exploited area.
- Password Spray: A common attack technique PEN Consultants conducts on many engagements is known as a password spray - testing a small number of well-known passwords across all accounts. That attack is successful nearly every time it is tried. Unfortunately for us, but great news for CLIENT, we spent almost an entire week running a password spray attack with 4,800 passwords, compromising zero accounts. This demonstrates the strength of NOT force rotating user passwords, as well as apparent user education about creating strong passwords.
- Password Audit: Another common issue we see during engagements is a significant portion of the users have common and weak passwords, such as those seen in Appendix #X. Doing so leaves accounts vulnerable to a number of password based attacks (ex. password sprays). During a password audit of CLIENT's user and service account passwords, it was noted that only a few of the very weak passwords (Appendix #X) were being used, and only a few additional users had some form of a password from a top-10k common password lists. Even brute-forcing (trying every combination) for a couple of hours only uncovered a few dozen passwords which revealed the usage of moderately strong passwords. This speaks volumes about the quality of passwords the users create, as well as the IT staff who undoubtedly have trained them to use strong passwords.
- User Training: CLIENT users demonstrated the ability to recognize and report when an MFA push notification was sent when they were not actively logging in to systems. Our attempts to get users with compromised passwords to approve push notifications were denied.
- Monitoring: CLIENT IT staff demonstrated an exceptional ability in recognizing various malicious activity during testing. Although there were some concerning logging/alerting gaps realized (documented in this report), there were other attacks CLIENT IT staff detected and immediately notified us, most of which are commonly missed by our other clients - ex. MFA lockouts, internal port scanning and service enumeration, detection of the external password spray attack and the exporting of user password hashes from Active Directory, etc.

- Response: CLIENTNAME IT staff's quick response time and remediation to some of the more serious vulnerabilities we discovered and reported during testing was exceptional - ex. the critical SQL Injection vulnerability (see FR-0xx).
- Comparison: It is difficult to make a claim that one corporate network is more secure than another, as there are many variables that would need to be considered. With that said, it is common for us to find at least 50% more issues in an environment of this size, on average. It seems clear that CLIENT IT has better secure configuration management practices than many.
- The limited number of people who have access to server areas made it harder for us to gain access to them. Because of this, it was not possible for us to access three of the six server areas.
- Exponential timeout for the WEBAPP1 login process, which helps defend against brute force attacks.
    - IMAGE OF TIMING TABLE

Statement of Work ("SOW")

The SOW from the contract has been included in Appendix #1 of this report for reference.

Scope

The [public IP ranges|private IP ranges|domains|applications] tested:

- [list here]

Red teaming, social engineering, physical attacks, wireless evaluations, denial of service (DoS), data modification/destruction (ex. cryptovirology or deleting), etc. were all out of scope.

Scope was maintained as described in the contract/SOW with the following exceptions:

- There were no exceptions.
- [or, list exceptions here]

Date(s) of testing

- 29 Jan: Demo of the web app provided by [CLIENT]
- 14 Feb 2018: As can be seen in "rawTimeline" on the Box share, product familiarization and setup started on 14 Feb 2018 while installing the client side application and troubleshooting issues.
- 15 Feb: Started manual testing – port scans, web crawls, DNS enumeration, user enumeration, manual vulnerability discovery and exploitation, etc.
- 18 Feb: Compromised a target server and requested a second test environment be setup so additional testing could be performed without touching the out-of-scope prod data.

- 20 Feb: Second test instance setup by Client. Full server exploitation and compromise of all test data.
- 21 Feb: Started the intense testing tasks – service enumeration, network vulnerability scans, brute force enumeration against various services, sqlmap, etc.
- 28 Feb: First user account compromised from password spray attack. Used access to collect 38 additional user IDs and launched a second password spray attack in parallel.
- 30 Feb: Pillage of the internal network, including gaining access to 6 employee accounts, RDP access to most of the internal servers, nearly every 3rd party service [CLIENT NAME] uses, and even administrative access to online banking (example.org).
- 02 Mar: Core testing complete
- 03 Mar: Began creating this report, while wrapping up additional testing tasks
- 08 Mar: Draft Findings and Recommendations Report complete and delivered to Client for review

Testers

The tester for this engagement was:

- Robert Neel
- Riley Neel

Testing Source IPs

The source IPs used during testing include, but may not have been limited to:

- 1.2.3.4

Labor Hours

PEN Consultants delivered an estimated 100 man-hours of time towards this testing:

- 10 hours: pre-testing - discussions, proposal/SoW, etc.
- 15 hours: product install/troubleshooting, product familiarization, open source research, initial testing, analysis, etc.
- 50 hours: core testing and analysis app, web, and infrastructure testing, analysis, etc.
- 25 hours (est): post-core testing - report creation, follow-up/additional testing, refining report with Client, etc.

Risk Rating Methodology

This report uses CVSS (Common Vulnerability Scoring System) as an attempt to rate the risk of each finding. The 3rd party hyperlink to the CVSS calculator is provided for each finding.

The overall rating given is based on the above mentioned calculation and these ranges:

Rating	Score Range
Info	0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 – 10.0

Details of the rating methodology can be seen here: https://www.first.org/cvss/specification-document.

Assumptions

- The testing was performed from various perspectives:
  - an external unauthenticated perspective
  - an authenticated, self-provisioned account
  - an external authenticated perspective (ex. compromised credentials)
  - internal unauthenticated access (ex. VPN access, physical network tap, etc.)
  - internal authenticated access (ex. server/workstation compromise, rouge employee or ATO, etc.)
  - internal authenticated privilege access (ex. sysadmin compromise)
- All findings and recommendations are based on the testing environment setup for the web application/API testing. The testing assumed an environment that matched the configuration of the production version.
- Testing was from the perspective of an attacker who performed due diligence in the reconnaissance phase (ex. prior reconnaissance visits) and took action to disable HVAC equipment (ex. disabling outdoor units) prior to their entry into the building - not simply someone who just walked in off the street.

Limitations

- Black box
  - Testing was close to full black box testing - few of our white box requests were provided (see Appendix #2). We were only given the required domain names/IPs/IP ranges.
  - All network, infrastructure, and web service scanning and testing was unauthenticated - testing credentials were not provided. As such, no checks requiring authentication could be run, and others could not be thoroughly performed - account brute force protections, thorough MFA testing, authenticated testing such as identity management, authentication, access control, session management, most input validation, business logic, client side attacks, etc.
  - No back-end access to the infrastructure was provided, so a subset of manual checks and tests could not be performed and others could not be thoroughly performed.
  - These limitations made testing less effective and thorough than full white box testing would have been, as well as limited our ability to understand misconfigurations that led

to given vulnerabilities and our ability to provide more detailed and specific recommendations. Unauthenticated, black box scanning and testing decreases testing thoroughness as much as tenfold.

- Interference / Access issues
  - Our testing IP was not granted a firewall bypass, so vulnerability scanning was negatively impacted. See FR-0xx for more details.
- The necessary mitigation efforts by Client against the discovered SQLi vulnerability early on in testing impacted our ability to perform post-exploitation analysis from that vantage point beyond what is shown in FR-0xx. Client determined on 13 Feb that further testing was not a priority at this time, therefore, steps were not taken to simulate continued access for testing purposes (ex. setting up a DB user for the testing).
- There were over 275 major features/pages, many with multiple sort, filter, search, and export options. This was a much larger scope than the declared 36. As such, only a subset of the typical scanners were used.
- Testing techniques and tactics were limited by applicable laws, permissions, and availability concerns, which limited our ability to exploit some vulnerabilities. It should be noted, an actual attacker does not operate under said limitations.

Post-Testing Tasks (for Client)

The following is a list of items Client should perform due to changes made by PEN Consultants and/or the level of access gained during testing. This is not a replacement for the more detailed recommendations included in the report, but it is merely a prioritized list:

- Destroy all tenants with the prefix “pentest*”. At most, there should only be three.
- Delete all files in the GCP bucket/path of: `[bucket name]/production/[GUID]/*`. There are likely hundreds of files that were generated during testing.
- If desired, the phishing email sent to all users could be removed from Inboxes
- Force a password reset for all Active Directory user accounts
- Disable or change the credentials issued for and during, testing.
- Change the VPN’s pre-shared key
- We added the ”pentestbasic” AD user to the “Remote Desktop Users” AD group during testing.
- Destroy the virtual machine used during testing.
- The database should be completely purged of all transaction data, as testing introduced many thousands of entries. Note: PEN Consultants was able to purge many of the entries and [CLIENT] (at our request) purged additional entries, but there are likely many more remaining.
- The testing environment should be completely refreshed from a backup/snapshot, as testing introduced many thousands of entries, including new user and group entries, that would be impractical to clean-up manually.
- Other than the above mentioned items, we are not aware of any other changes made to CLIENT’s systems or data that would require follow-up action.

Pretext

General Pretext

Of the five pretext options provided, “HVAC Tech” was chosen by the client for this engagement. The details of this pretext can be seen in Appendix #x.

Of the six white box vs black box options given for the chosen pretext, white box was selected for all – floor plans provided, site familiarization visit, interception of the service requests, client initiated temperature modification, etc. These white box options were selected due to time and budget constraints, primary purpose of the testing, and safety of IT equipment.

For more about white box and black box approaches, see https://penconsultants.com/graybox

Timeline

This is the general timeline PEN Consultants followed for each of the four sites. Note: The details of what took place at each site are included as Appendices.

1. Work order on phone app: Work order was sent to tester’s phone app before arrival to the site, and sometimes modified while on-site to match a new cover story. Example:
   IMAGE HERE
2. Arrive on-site & park: For situational awareness, tester sent text to [CLIENT NAME] POCs about being at site X. Video recording equipment was enabled (see Appendix #x for details), and tool bag, ladder, etc. was carried in.
3. Enter building: When arriving to the first site before hours, testers waited for an arriving employee to let them in. For all other sites, testers entered the front lobby door.
4. Greet first employee seen: Testers told employee a service request had been received for the HVAC unit in the “IT room” (exact terminology differed some from site to site) and showed the employee the work order, on the phone app, to make the story more believable.
5. Core Social Engineering
    1. In some cases, testers were escorted and taken to the server area.
    2. For some sites, testers had to use a “could we just stick our head in since we’re here and this is urgent” type comment.
    3. On one occasion testers had to call a fake dispatch (i.e. PEN Consultants surveillance team outside) and ask them to call “John”. They faked a call to “John”.
    4. On another occasion testers requested “John’s” number from the fake dispatch and then called him directly from within the bank. Testers told the [CLIENT NAME] employee “John” was on the phone and he was saying it’s okay to enter. [CLIENT NAME] employee did not actually talk with “John”.
    5. If needed, we were prepared to have “John” speak directly with the employee and give approval for entry, but that was never needed.
6. Server area: While in server area, testers performed a few tactics to divert attention or get escort to leave the area. Action-on-Objective (AOO) was to plug in a hardware device - small box, short CAT5 tail (see Appendix #x for details) - into an open network port. Once AOO was performed, testers “fixed” the HVAC unit.

7. Leave building: Testers exited the room with escort, signed out, and left building.

Findings and Recommendations

When possible, Free Open Source Software (FOSS) tools are shown in the examples, as opposed to paid solutions. This allows you to reproduce, for free, and also helps highlight how easy it would be for the most inexperienced attacker to discover and execute the same vulnerability/attack vector.

Summary – Testing Results

The details of what took place at each site are included as Appendices (Appendix #x - #y). The following table gives a high level summary of how each site performed. These checkpoints are based on [CLIENT NAME]'s Visitor and Vendor Policy (see Appendix #x) and the objectives of testing set out in the SOW (see Appendix #1).

	Location #1	Location #2	Location #3	Location #4
Visitor Log binder and badges available	✓	✓	✓	✓
Visitor sign-in required	✓	✓	✓✗	✓
ID requested and verified	✗	✗	✗	✗
Copy of ID retained with Visitor Log	✗	✗	✗	✗
Attempted to verify visit with appropriate internal contact before entry	✓	✗	✓	✓
Verified visit with appropriate internal contact before entry	✗	✗	✗	✓
Escorted/supervised into a non-public area	✓	✓	✓	N/A
Maintained escort/supervision in a non-public area	✗	✗	✓	N/A
Escorted/supervised into server area	✓	N/A	✗	N/A
Maintained escort/supervision while in server area	✗	N/A	✗	N/A
Client's network integrity maintained (AOO unsuccessful)	✗	✗	✗	✓

Summary of Findings

Sorted by Reference Number

Ref #	Description	Localization	Risk Rating
FR-001	Disable GraphQL Introspection	Web Server	Info
FR-002	Knowledge Bases (KBs)	Knowledge Bases	Medium
FR-003	Version Disclosure	Web Server	Low
FR-004	Missing Security Headers	Web Server	Medium
FR-005	Auto Provisioning	Web Server	Info
FR-006	Brute-Force User Enumeration	Web Server	Medium
FR-007	Username == Email Address	Web Server	Info
FR-008	Brute-Force Authentication	Web Server	High
FR-009	Password Policy	Web Server	Medium
FR-010	User Notifications	Web Server	Low
FR-011	Email Verification	Web Server	Low
FR-012	MFA	Web Server	Medium
FR-013	Privilege Escalation	Web Server	Critical
FR-014	Chat Transcripts	Web Server	Medium
FR-015	Session Termination	Web Server	Medium
FR-016	Session Timeout	Web Server	Medium
FR-017	Session Control	Web Server	Low
FR-018	HttpOnly Flag	Web Server	Low
FR-019	SQL Injection	Web Server	Critical
FR-020	Stored XSS	Web Server	Medium
FR-021	TLS Weaknesses	Web Server	Low
FR-022	CORS	Web Server	Medium
FR-023	External Service Interaction	Web Server	Medium
FR-024	User Experience	Web Server	Info
FR-025	VDP & Bug Bounty	Company Policy	Info

Sorted by Risk Rating

Ref #	Description	Localization	Risk Rating
FR-019	SQL Injection	Web Server	Critical
FR-013	Privilege Escalation	Web Server	Critical
FR-008	Brute-Force Authentication	Web Server	High
FR-012	MFA	Web Server	Medium
FR-014	Chat Transcripts	Web Server	Medium
FR-016	Session Timeout	Web Server	Medium
FR-004	Missing Security Headers	Web Server	Medium

Ref #	Description	Localization	Risk Rating
FR-006	Brute-Force User Enumeration	Web Server	Medium
FR-009	Password Policy	Web Server	Medium
FR-015	Session Termination	Web Server	Medium
FR-023	External Service Interaction	Web Server	Medium
FR-022	CORS	Web Server	Medium
FR-020	Stored XSS	Web Server	Medium
FR-002	Knowledge Bases (KBs)	Knowledge Bases	Medium
FR-003	Version Disclosure	Web Server	Low
FR-010	User Notifications	Web Server	Low
FR-017	Session Control	Web Server	Low
FR-018	HttpOnly Flag	Web Server	Low
FR-021	TLS Weaknesses	Web Server	Low
FR-011	Email Verification	Web Server	Low
FR-001	Disable GraphQL Introspection	Web Server	Info
FR-005	Auto Provisioning	Web Server	Info
FR-007	Username == Email Address	Web Server	Info
FR-024	User Experience	Web Server	Info
FR-025	VDP & Bug Bounty	Company Policy	Info

Summary – Recommendations

The overarching recommendations are:

- **Top Recommendation - Session Management and Access Control:** Strengthen the authentication processes to prevent undetected brute force user enumeration, password compromise, and account take over (ATO), improve session management to prevent hijacking, tighten access control of cloud resources to prevent unauthorized users from accessing confidential data, and overhaul the access controls between users, roles, and tenants to prevent an attacker from compromising data at-will.
  - Examples: Privilege escalation, unauthenticated access to cloud resources such as chat archives, session control and termination failures, lengthy session timeout, brute force user and password enumeration, absence of MFA, exposed knowledge bases with confidential data, etc.
  - References: FR-0xx, FR-0xx, FR-0xx, FR-0xx, etc.
- **User Input Validation:** Implement more robust user input sanitization, validation, and enforcement for user input to prevent back end compromise through SQL injection (SQLi), user compromise through XSS, ATO due to poor username and password policies.
  - Examples: Input sanitization issues leading to XSS and SQLi vulnerabilities, weak username and password policies, failure to verify email addresses and other user profile changes, etc.
  - References: FR-0xx, FR-0xx, FR-0xx, FR-0xx, etc.

- **Configuration Management:** Improve configuration management which will resolve many of the issues discovered, reduce the likelihood of attacks demonstrated in this report, help mitigate an infrastructure breach, and minimize the impact should a breach occur.
  - Examples: Exposure of the GraphQL schema, multiple issues exposing users and their sessions to attack, such as missing security headers, cookie and TLS weaknesses, misconfigured CORS headers, etc.
  - References: FR-0xx, FR-0xx, FR-0xx, FR-0xx, etc.
- **Additional & Continued Testing & Training:** Based on observations made during this engagement, PEN Consultants recommends continued testing, monthly developer training, internal and external network penetration testing, as well as testing of other web applications. More details:
  - Continued testing of newly developed web app features and major releases on a quarterly basis for at least the next year. We recommend a white box testing approach, to include an audit of the Google Cloud/GCP environment, during future testing to ensure thoroughness.
  - Monthly developer training, through our Cybersecurity Unlimited retainer service – https://penconsultants.com/cybersecurityUnlimited, in order to help prevent common mistakes, such as those seen in the OWASP Top-10.
  - Internal and external network penetration testing against [CLIENT]'s corporate network, to ensure the corporate network is not susceptible to attack.
  - Consider social engineering assessments in conjunction with the network penetration testing, as this is the most common way attackers breach corporate networks.
  - If not already performed, web application penetration testing of all in-house developed web apps.
-

FR-001: Log Verbosity

Category: Logging
Mitre ATT&CK: Defense Evasion
Testing Standard: OWASP OTG-CONFIG-002
Localization: Application
Total Calculated Risk: 2.3 - Low
CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:L/I:N/A:N

Details

Log verbosity of [PRODUCT NAME] is quite detailed (in TRACE mode). The log verbosity was detailed enough for an attacker to [DETAILS HERE]. Add to this the fact that (by default) this log file is readable by all users, this is pretty trivial to pull off.

Best case scenario, an attacker with admin privileges enables a higher log verbosity and catalogs the things needed to [DETAILS HERE]. Worst case, a sysadmin forgets to reset log verbosity to a lower level and even a non-admin user/attacker can use this information to [DETAILS HERE].

Example…

[SCREENSHOT HERE]

With that knowledge, we were able to [DETAILS HERE].

Recommendation

Consider a capability for a customer to optionally set a max limit on the verbosity, within the binary itself (hard coded max). This would make it harder for an attacker to see this very low-level detail by a simple configuration change on the (untrusted) endpoint.

Some security-conscious customers maintain two sets of binaries for highly secure applications. One set includes things like debugging symbols. The second set does not include symbols and has logging disabled (or minimized to error only). The “debug” version of the software is used only when troubleshooting is needed.

FR-002: Elasticsearch

Category: Configuration Management
Mitre ATT&CK: Collection
Testing Standard: OWASP OTG-CONFIG-002
Localization: Graylog Server
Total Calculated Risk: 7.6 - High
CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:N

Details
Elasticsearch, running on 100..x.y.z, is vulnerable to CVE-2019-7611 and to Unrestricted Access Information Disclosure.

CVE-2019-7611
The Elasticsearch version running (5.6.13) does not have 'xpack.security.dls_fls.enabled' set to 'true' in the 'elasticsearch.yml' config file, which causes certain permission checks to be skipped for certain user actions and can lead to an attacker gaining access to restricted indexes. [TENABLE-01]

Unrestricted Access Information Disclosure
Elasticsearch, running on 100.x.y.z:9200, requires no authentication to access log data being received from the entire [CLIENT] platform.

Example: curl -X GET 'http://100.x.y.z:9200/_cat/indices?v'
IMAGE HERE

With this access, simple search queries such as curl -X GET 'http://100.x.y.z:9200/graylog_72/_search?q=password' were performed to locate various credentials, which allowed us to gain access to other systems.

IMAGE HERE

Impact Example #1 – Vault Service

Vault credentials
IMAGE HERE

Used credentials to authenticate with Vault
IMAGE HERE

Impact Example #2 – Maria DB

DB credentials
IMAGE HERE

Used credentials to authenticate with DB
IMAGE HERE

Recommendation

- CVE-2019-7611

    - At minimum, set ‘xpack.security.dls_fls.enabled’ to ‘true’ in ‘elasticsearch.yml’.
    - Consider upgrading to the latest version of Elasticsearch, which addresses this vulnerability as well as other enhancements.
    - More information:
        - https://nvd.nist.gov/vuln/detail/CVE-2019-7611
        - https://www.elastic.co/guide/en/elasticsearch/reference/current/security-settings.html
- Unrestricted Access Information Disclosure
    - Enable and require authentication to access the logs stored in Elasticsearch
    - More information: https://www.elastic.co/blog/demystifying-authentication-and-authorization-in-elasticsearch

FR-003: Email Server Misconfiguration

Category: Identity Management
Mitre ATT&CK: Discovery
Testing Standard: OWASP OTG-IDENT-004
Localization: Mail Server
Total Calculated Risk: 3.7 - Low
CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

Details

A vulnerability exists in your email server that allows an attacker to enumerate your email accounts. This is a notable flaw for the following reasons:

- Active accounts can be identified to send spam to your employees, or worse, send phishing emails, in an attempt to gain access to your systems.
- "This information obtained can be used by an attacker to gain a list of users and to attack, for example, through a brute force attack." [OWASP-02]

Verified Attack #1 - Single Email Address + Custom Exploit

Utilized the following PEN Consultants developed exploit tool, https://gitlab.com/J35u5633k/emailScripts_public/blob/master/checkIfValidEmailAddress.py

Example of an invalid email address:

IMAGE HERE

Example of a valid email address:

IMAGE HERE

Verified Attack #2 - Common Last Names + Custom Exploit

In this attack, we downloaded the top-100 first names and fed it into a similar PEN Consultants developed exploit tool, https://gitlab.com/J35u5633k/emailScripts_public/blob/master/checkIfValidEmailAddress_bulk.py.

IMAGE HERE

This attack sent 100 email addresses/requests to the email server. It uncovered 9 valid email addresses. Had we used a larger last names list (ex. top-1000), and increased the speed of the script, we would have discovered additional addresses. Likewise, we could use a top-1000 last names list and enumerate all 26 first initials to recover the newer FLast@example.com formatted email addresses also.

IMAGE HERE

Verified Attack #3 – Partially open mail relay

[CLIENT]'s email service has a "partially open mail relay". This means an unauthenticated, external attacker is able to send email from one example.com email address to another example.com email address, from external, without authenticating, using the mail service at example.com.

Below is one example. Notice [CLIENT]'s mail service never verifies our identity.

IMAGE HERE

Notice how "John's" signature is automatically appended to our email. This is extremely convenient for the attacker since we do need to perform any reconnaissance to determine what his real signature looks like. Exchange is kind enough to append it for us, which further legitimatizes the appearance of the email.

Some security professionals state this is merely a spoofing attack. A basic spoofing attack would be, for example, an attacker impersonating GMail.com, Yahoo.com, etc. to your mail server, and your server blindly accepting that as valid, without verifying. At minimum, your server fails to do this, as we claimed to be example.com, but were in fact external. If there is any domain in which example.com should be able to properly identify, that domain should be itself. It is for this reason that PEN Consultants, as well as a subset of other industry leaders, argue this is more than just basic spoofing.

This vulnerability was exploited during the social engineering/phishing assessment and is likely one of the leading factors in the high number of user click-through rates.

In addition to being able to send phishing email, this vulnerability could be leveraged to perform a Man-in-The-Middle (MiTM) attack. Example:

- Malicious Actor (MA) uses employee A's account (ex. an HR employee) to email employee B a request for a form to change paycheck deductions. MA sets the "reply to" address to an account he owns (ex. a Gmail address).
- MA receives the reply from the employee B, which has the form attached with requested information.
- MA now has the employee's social security number, banking information, etc. (whatever was requested on the form).

Recommendation

Verifying a sender is not always an easy process - it is error prone and can lead to legitimate email being placed in spam or dropped altogether. However, it is trivial for your mail server to validate one of your own users, as it is the authoritative source for validation - it can request the user's username and password. Our recommendations are as follows:

- Your email server should be configured to give the same response, regardless of whether the email address is valid or not. There is likely no operational need for it to operate as it is now.
  - If your email provider is unable to remediate this vulnerability, seek a new provider.
  - More on this topic can be read here: https://penconsultants.com/bruteEmail

- Your email spam solution could likely benefit from additional configuration changes and/or monitoring to prevent, or at least detect, these massive brute-force attacks. We brute-forced 26,000 email address in a very short period of time (over 330 per minute), from a single source IP. This activity should have stood out from normal activity.
- Internal corporate usernames should always be different from email addresses and not easy to correlate. Example: John Doe may have an email address of JDoe@domain.com, while his internal username is TW2342.
- Use the various industry recommended methods to validate the sender, such as reverse DNS lookups, SPF, DKIM, and DMARC checks, etc., and ensure they are enforced.
- Force authentication for senders claiming to be a member of your domain.
- Work with Microsoft to address the following:
  - Your email service should be configured to give the same response, regardless of whether the email address is valid or not. There is likely no operational need for it to operate as it is now.
  - Force authentication for senders claiming to be a member of your domain.
  - If they are unable to resolve this, choose a new email provider. Microsoft historically has a weak track record on basic email security principles, while simultaneously having one of the highest false-positive rates of the major mail providers (i.e. marking good email as spam). They have been getting better as of late, but provably, they are not quite there yet.
- Consider including this service in a future testing engagement.
- Require staff to take Information Assurance (IA) training on a regular (annual) basis. The staff needs to have a healthy skepticism of anything coming in via email that requests confidential information.
- More on this topic can be read here: https://penconsultants.com/bruteEmail

FR-004: Brute-Force User Enumeration

Category: Identity Management
Mitre ATT&CK: Discovery
Testing Standard: OWASP OTG-IDENT-004
Localization: API
Total Calculated Risk: 3.7 - Low
CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

Details

A vulnerability exists in the login process that allows an attacker to easily brute force for valid user accounts.

As a test, we performed close to 100 login attempts using a mix of known valid and invalid user accounts (and a bad password) in order to baseline the response time:

IMAGE – TABLE SHOWING RESPONSE TIMES

As can be seen in the diagram:

- Invalid user account timing: 558 ms (avg) with a standard deviation of just 5 ms
- Valid user account timing: 340 ms (avg) with a standard deviation of 260 ms

These timing metrics make it trivial to brute force for valid user accounts, which is a prerequisite for launching a password attack (separate finding). By simply checking if the timing is between 273 and 295 ms, we can predict valid/invalid usernames with 98% accuracy with a single pass. A second pass gives near 100% accuracy (our prediction was never incorrect with two passes).

"This information obtained can be used by an attacker to gain a list of users and to attack, for example, through a brute force attack," or to launch an account DoS attack. [https://wiki.owasp.org/index.php/Testing_for_User_Enumeration_and_Guessable_User_Account_(OWASP-AT-002)]

Recommendation

Ensure the response time for login attempts are either the same between valid and invalid accounts or sufficiently random.

- We are not aware of a vendor solution to this vulnerability, but we recommend making contact with Microsoft to confirm. If they cannot provide a solution to this issue, consider a more secure vendor's solution.
- These are our boilerplate recommendations. Feel free to pass this along to the vendor:
- Solution A (best): Ensure the response times are exactly the same regardless of account validity:
    1. Record time1
    2. Accept user input

3. Process request on the back-end
4. Record time2
5. Wait until time2 - time1 = 3 seconds (for example)
6. Return page response (ex. HTTP 200-ok)

- Solution B (sufficient): Introduce a randomized time delay. Example for one.example.com:
  - Introduce a 0.4 – 2.4 second random delay for invalid accounts
  - Introduce a 0.7 – 2.7 second random delay for valid accounts
- No matter which solution is used, it will be important to periodically evaluate actual processing time and ensure the mitigation remains effective.

FR-005: MFA

Category: Identity Management
Mitre ATT&CK: Credential Access
Testing Standard: OWASP OWASP-AT-009
Localization: Application
Total Calculated Risk: 6.5 - Medium
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

Details
There seems to be a lack of Multi-Factor Authentication (MFA). This makes [web app] vulnerable to many forms of password and DoS attacks, as seen in other findings in this report.

There are security questions prior to the password screen, but those are no longer industry acceptable practices. In addition to this configuration being weak at protecting account access, it also makes the account vulnerable to DoS (see FR-031).

Security questions are referred to as MFA in the app, but that is not MFA per NIST. NIST states, MFA refers to the use of more than one of the following:

- Something you know (e.g., a password).
- Something you have (e.g., an ID badge or a cryptographic key).
- Something you are (e.g., a fingerprint or other biometric data).
- Source: [NIST-01]

A security question challenge is “something you know” just as the password and username are, therefore, this is still considered single factor authentication, not multi-factor authentication (MFA).

NIST states: *“Knowledge-based authentication (KBA), sometimes referred to as ‘security questions’, is no longer recognized as an acceptable authenticator by SP 800-63. This was formerly permitted and referred to as a ‘pre-registered knowledge token’ in SP 800-63-2 and earlier editions. The ease with which an attacker can discover the answers to many KBA questions, and relatively small number of possible choices for many of them, cause KBA to have an unacceptably high risk of successful use by an attacker.”* [NIST-02]

Single factor authentication (ex. username/password) is considered a high risk and is advised against by an increasing number of industries and standards (ex. OWASP, PCI DSS, etc.) [OWASP-05]. MFA is a must for all remotely accessible services, but is even more critical for VPN and administrative accesses. Attackers frequently target these services using stolen credentials and/or password attacks (ex. brute-forcing, password spraying, etc.). Compromised credentials are a leading cause of system and data breaches. MFA is one of the most effective protections in an overall strategy to prevent this attack vector. [VDBR-01] [CENTRIFY-01]

Note: Brute force and other password type attacks were out-of-scope for this device. Historically, some of these attacks have a high probability of success.

<u>A note about SMS based MFA</u>

SMS based MFA (i.e. sending a one time code) is one of the weakest MFA solutions. With that said, it is still over 75% effective against even the most advanced and targeted of attacks, while remaining one of the most simple, affordable and user friendly solutions. Although users and admins should be encouraged to use a more secure solution, PEN Consultants still recommends maintaining this as an option for your clients.

More information and research on that topic can be found here:
https://penconsultants.com/MFAFUD

Recommendation

- Implement MFA to reduce the risk of compromised credential access. The most effective mitigation is to implement MFA and verify the MFA token before the password (in the back-end) is checked. This has a dual benefit of protecting against brute-force attacks and other forms of password attacks (ex. password spraying), as well as preventing a DoS attack against known accounts.
- MFA needs to be enabled and forced for admins and employees. We would encourage the use of at least SMS based MFA for even your clients. It is understandable if there is business justification for not utilizing a more secure MFA solution (ex. Google Authenticator) because of user experience. However, nearly all major sites employ SMS based verification, at minimum. Because of that, your users should be comfortable using it on your system as well.
- For SMS based MFA, verify the SMS OTC/OTP token before the user is given a pass/fail. To do so, collect all three pieces of information (username+password+mfa) before providing a pass/fail result to the user (or creating even so much as a noticeable timing difference). To prevent a user from receiving a flood of SMS messages (ex. during a brute force attack), verify the username+password (on the back-end only) before sending out the SMS.
  - Example: Once the username+password is entered and the user is taken to the MFA page, present a message such as, *"If you did not receive a SMS message with the OTC, then your username or password is likely incorrect"*.
  - NIST's recommendations should be followed to include never locking out an account based on failed passwords & implement proper rate-limiting mechanisms.
  - Side note: SMS based MFA is a little tricky since the app needs to verify the username+password prior to sending out the SMS. A token based MFA solution (ex. Google Authenticator) simplifies this greatly as the initial page would include a field for all three pieces of information (username+password+mfa) and not even check the password unless the MFA token is verified first.
- Ensure the MFA OTC/OTP token is temporarily invalidated, once verified, to prevent it from being brute forced, and, subsequently used. Note: Do not permanently invalidate the MFA token, as a 6-digit OTC will eventually repeat with time.
- Consider allowing multiple MFA methods, to include a token based method. Examples: Yubico YubiKey, Symantec VIP, Google Authenticator, etc.

- More information about MFA attacks and protections can be found here:
  - https://penconsultants.com/MFAFUD
  - https://penconsultants.com/MFAAttacks
- Get rid of the security questions, per NIST.

FR-006: Session Termination

Category: Session Management
Mitre ATT&CK: Exploit Public-Facing Application
Testing Standard: OWASP OTG-SESS-006
Localization: Web Server
Total Calculated Risk: 5.7 - Medium
CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:N/A:N

Details

The logout function is user side only (user side storage of token cleared). The token is not invalidated on the server side. This greatly increases the "time period an attacker can launch attacks over active sessions and hijack them" [227].

As a test, we authenticated into each user, saved off the token, clicked "log off", and then inserted one of the previous tokens, at will, to jump back and forth between users. Additionally, multiple valid session tokens for a single user were collected and we jumped back and forth among sessions. As an example, these session tokens, seen in Chrome's local storage log, were all simultaneously active, even though "log off" was clicked each time.

[SCREENSHOT HERE]

An additional problem discovered was that closing the browser does not delete the client side session values. It was determined that session values and tokens are being stored in Local Storage as opposed to the more secure Session Storage or cookies.

The likelihood of this being abused greatly increases in conjunction with the next finding (FR-0xx).

Recommendation

- "Do not store session identifiers in Local Storage...use the object sessionStorage instead of localStorage. SessionStorage object is available only to that window/tab until the window is closed". Alternatively, "cookies can mitigate this risk using the httpOnly flag". [https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html]
- Ensure the log off function invalidates the session token on the server side. Per OWASP, "the web application must invalidate the session at least on the server side". [https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#session-expiration]

FR-007: IWA Scope

Category: Authentication
Mitre ATT&CK: Exploit Public-Facing Application
Testing Standard: OWASP OTG-AUTHN-010
Localization: Server
Total Calculated Risk: 9.0 - Critical
CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H

Details

We were able to brute force a local Windows account through the server's web interface, including, but not limited to, the local administrator account.

Integrated Windows Authentication (IWA) has inherent flaws that greatly increase the risk of server compromise unless purposeful actions are taken to mitigate. By default, IIS hosted sites (with authentication enabled) allow any local account to authenticate over web apps/IWA, even if that account has no permissions within the application itself.

Here is an example of the screen one gets once the correct password for a local account (that does not have app permissions) is guessed…

[SCREENSHOT HERE]

Once we compromised certain local accounts via IWA (the screenshot above), we were then able to RDP using the brute forced credentials.

Recommendation

- At minimum, restrict the web application to only those users intended to use web app: https://support.microsoft.com/en-us/help/815151/how-to-restrict-specific-users-from-gaining-access-to-specified-web-re. Make sure those accounts can access nothing else on the system (ex. rdp).
- You will most likely want to enable the built-in Windows brute force policies regardless of the other options below.
  - Note: We enabled it (temporarily) and then caused a denial service attack against multiple accounts (including local admin accounts). So, this alone, is not a complete solution.
  - More information about brute force attacks can be found here: https://www.owasp.org/index.php/Top_10-2017_A2-Broken_Authentication.
- Integrated Windows Authentication (IWA) is an easier vendor provided solution, but is not always the best choice for secure web apps, especially those operating over the internet (as opposed to an intranet). [CLIENT NAME] may want to consider supplemental code/processes to make IWA more secure (additional layers, configuration checks, etc). Replacing the entire authentication mechanism with another solution entirely is another option.
- As mentioned above, MFA (if implemented correctly) would prevent DoS/account lockout through account brute forcing.

- Ensure password policies are enabled and securely configured to ensure it is infeasible to guess passwords. More information about that topic can be found at: https://penconsultants.com/passwordPolicy

FR-008: Account Lockout

Category: Authentication
Mitre ATT&CK: Exploit Public-Facing Application
Testing Standard: OWASP OTG-AUTHN-003
Localization: Web Server
Total Calculated Risk: 3.7 - Low
CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L

Details

The web app successfully mitigates against a brute-force attack. Unfortunately, it does so in a manner that creates an account Denial of Service (DoS) vulnerability.

We were able to easily DoS accounts through the login process since there is no captcha or MFA to prevent a scripted attack. Intentional bad passwords led to an account DoS in 5 attempts. The lockout being permanent makes this problem a little more serious. For example, an attacker can lockout all admin accounts first, then others, thus making it more difficult for an admin to keep accounts unlocked during an attack.

Additionally, no user or admin notification is recorded or sent for an account lockout.

Recommendation

- Modify the password policy to be more inline with NIST and PEN Consultants' recommendations, specifically, those that prevent account DoS attacks. See "Implement one or more rate limiting solutions" below.
- Implement one or more rate limiting solutions.
  - See FR-0xx (AD Password Policy) recommendations for details.
- Add other brute-force protections, such as a WAF, which may help by temporarily blacklisting offending source IPs, for example, during externally launched attacks.
- Ensure each user, in addition to admins, has access to robust user accessible notifications and logs related to authentication and profile change activity in the GUI.
- MFA
  - See FR-0xx (MFA) for details

FR-009: Application Password Policy

Category: Identity Management
Mitre ATT&CK: Credential Access
Testing Standard: OWASP OTG-AUTHN-007
Localization: example.org
Total Calculated Risk: 5.3 - Medium
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N

Details

The web application's password policy is not inline with NIST standards. The most significant issues are (1) the inability for a user to use a password > 12 characters, (2) the ability to use special characters outside the 9 listed, and (3) an account DoS vulnerability.

Password policy grade: 64%

The web app's current stated password policy:

IMAGE HERE

Based on NIST's policy, found at https://pages.nist.gov/800-63-3/sp800-63b.html#sec5, the web app's password policy passes/fails at the following:

- Pass: No truncation
- Pass: Do not force password changes arbitrarily (e.g. periodically)
- Pass: Must allow the paste function (ex. password managers)
- Partial: Must compare against commonly used, expected, or compromised passwords and prevent use.
  - Pass: dictionary words
  - Pass: repetitive or sequential characters as the password (exclusively)
  - Partial: previous breach lists
    - Note: Even extremely common passwords, such as "password", were allowed in addition to other common passwords such as Password1, Summer2020, God12345, Welcome1, Secret123, etc. See Appendix #3 for more details.
  - Fail: context-specific words (name of the service, username, etc.)
  - Note: As seen in FR-019 (SQLi), 18% of the user password hashes we collected were cracked with a basic dictionary attack. This demonstrates clearly the fact that, if users are not prohibited from using weak passwords, they will.
- Pass*: Allow >=64 characters
  - Note: 73+ characters causes unexpected behavior (see note below)
- Fail: Allow all printed ASCII
  - Note: X does allow all printable ASCII, but Y only allows the following 4 special characters: "@_.-"
- Fail: Allow unicode
  - This could cause problems with non-US/English based users

  - Note: Using a unicode password causes unexpected behavior (see note below)
- Fail: Display a password strength meter
- Ineffective: Rate-limiting mechanism: captcha, exponential increasing wait, MFA, etc.
  - Note: X's brute force prevention is dangerous. Once an account locks, the web app keeps it locked for 24-hours with no user self-service unlock feature. It requires X staff intervention (i.e. a support phone call). More on this DoS vulnerability below.
  - This is expanded on in FR-0xx
- Fail: Do not require a mixture of different character types or forbid consecutively repeated characters
  - Per the app: Passwords "must contain an upper-case letter, lower-case letter, and number"
- Fail: Offer an option to display the secret (default hide, allow show) on login screens

Rate-limiting mechanism

A rate limitation mechanism's primary purpose is to prevent traditional brute force password attacks. Although the existing solution, account lockout, is one solution to help mitigate these attacks, it is no longer preferred as it creates an account DoS vulnerability.

When considering the user name enumeration vulnerability in FR-0xx, and how trivial it would be to script a known-bad password attack against the web app's login page, one can see how easy it would be to cause a DoS across many accounts. Best case scenario, this would cause call volumes to spike and likely some brand reputation issues until the attack could be stopped. It is important to note, stopping the attack would likely require changes to all affected usernames. Simply blocking the attacker's IP, for example, would cause the attacker to change the source of the attack, or even distribute the attack across many thousands of sources (ex. leasing a bot net).

** Outside of the testing accounts, PEN Consultants was unable to safely test the above mentioned attack, but we are highly confident in it being successful.*

Recommendation

- Correct the bug in XYZ that is causing the instability when using passwords of length >20 or unicode characters.
  - An immediate solution is to determine why a generalized password change error/failure does not propagate up to the user and allow them to correct the issue. The long term solution is to support unicode (ex. International clients) and maybe longer passwords. Note: The allowable length is greater than the standard now.
- Modify the password policy to be more inline with NIST recommendations.
  - As mentioned above, the three greatest concerns are not supporting passwords >12 characters in length, allowing special characters outside of the 9 supported, and preventing account DoS attacks.
  - Get rid of the password rotation (i.e. set max password age to "0")
  - Set the minimum password length >12 characters, instead of 8

    - Consider changing the lockout to 5 failed passwords in 10 minutes, or 10 failed passwords in 60 minutes, or both
    - In addition to, or instead of the above, implement auto unlock
    - Restrict the use of “commonly used, expected, or compromised passwords”
    - Read more here:
        - https://penconsultants.com/passwordPolicy
        - https://penconsultants.com/forcedPasswordRotation
- Implement one or more rate limiting solutions
    - Properly implement captcha: At minimum, consider the use of captcha on the authentication page to mitigate automated/scripted attacks. Captcha will not prevent manual account lockout attacks, but it will mitigate mass scripted attacks.
    - The preferred method is exponential rate limiting for each failed attempt; keep the state of user login attempts and exponentially increase timeout for each error: 1 second, 2, 4, 8, 16, etc. for ANY source (not simply source IP based). This makes brute-forcing impractical, while not causing too much burden on a user who fails a few times to enter the correct password.
        - Alternatively: Lock out account+srcIP tuples instead of locking out the account for all source IPs. This would block an attacker, but not affect the legitimate user.
- Audit passwords
    - Periodically dump all user hashes from AD and run John The Ripper against them using a large dictionary file (ex. a million or more words) to verify your password complexity policies are working. Many of your users’ passwords are so weak they would fail against multiple types of password attacks, so they should proactively be identified.
        - Resource to help with this: https://www.dionach.com/blog/active-directory-password-auditing-part-1-dumping-the-hashes/
    - Note: You should crack these in a “safe” manner, meaning (1) you are not exposed to the password and (2) the cracked password is not stored. Your process should simply give a list of whose account passwords were cracked and NOT display/store the actual password.
- MFA
    - The authentication process needs properly configured MFA to prevent this and other brute-force type attacks (see FR-0xx).

FR-010: AD Password Policy

Category: Identity Management
Mitre ATT&CK: Credential Access
Testing Standard: OWASP OTG-AUTHN-007
Localization: Active Directory
Total Calculated Risk: 5.3 - Medium
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N

Details

Password Rotation Policy

100% of the 17 Active Directory account passwords compromised used an obvious form of incrementing* for each password change, making the password rotation policy not only completely ineffective, but in some cases, trivial to compromise.

* More details on what incrementing is can be found here:
https://penconsultants.com/forcedPasswordRotation

These are the actual passwords discovered, to give you an idea. Note: Each has had a portion of the password redacted or changed slightly as to not display the actual password:

- Seasonal passwords
  - January1
  - December2018
  - Winter51
- “Secret”
  - Secret2018
  - Secret13
  - Secret21
- Names of children, spouses, or (presumably) pets
  - [name]10
  - [name]2011
  - [name]13*
  - [name]20
  - [name]14
  - [name]15!!!
  - [name]7
  - [name]35
- Other
  - [redacted]83
  - [redacted]09
  - 80[redacted]
    - ^ The only user to place their increment at the beginning of the password.

As can be seen from that list, the first two categories are trivial to comprise through attacks such as password sprays, as they are common passwords that people use when they are forced to rotate their password every 60-90 days. The third category (“Names”) could be compromised

fairly easy with a few minutes of research – collecting the names of the person's spouse and kids. Although the last category would not be as easy to guess, the incrementing that users perform with the same base password makes the password rotation policy meaningless from a security standpoint.

Password Lockout Policy

The current Active Directory settings successfully mitigate against a traditional brute-force attack. Unfortunately, it does so in a manner that creates an account Denial of Service (DoS) vulnerability, while offering little protections against a password spray attack (see FR-0xx for details).

During testing, PEN Consultants requested and was granted permission to carry out a password spray attack (see FR-0xx for details). One of the safe guards we took was to ensure we were well under half of the passwords/hr lockout rate to reduce the likelihood of account lockouts. This ended up being a 30 minute sleep, or pause, placed in our script between every password attempt. Despite introducing this extreme delay, we still locked out all accounts being targeted at one point.

Not only was it easy to DoS the internal AD accounts through the external facing OWA, we did it while purposely trying to avoid it. Since there is no captcha or MFA to prevent this attack, it takes only a few seconds (5-7 seconds) to send three intentional bad passwords per account to trigger an account DoS across the entire domain. The lockout being permanent (i.e. requiring manual unlock) makes this problem a little more serious. For example, an attacker can lockout all admin accounts first, then others, thus making it more difficult for an admin to keep accounts unlocked during an attack until all external facing services are shut down.

Recommendation

- See "Password Policy" above for more recommendations

FR-011: Reflected Cross Site Scripting (XSS)

Category: Input/Data Validation
Mitre ATT&CK: Exploit Public-Facing Application
Testing Standard: OTG-INPVAL-001
Localization: Web Server
Total Calculated Risk: 4.3 - Medium
CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N

Details

A Reflected XSS vulnerability was discovered in multiple locations which could be used by an attacker to introduce malicious functionality that may affect the user's device or account.

"Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application. The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes." [865]

Example 1: `https://testing.example.com/Monitor.aspx?Source=what"><script>alert("XSS")</script>ever`

IMAGE HERE

Produces:

IMAGE HERE

Example 2: `EXAMPLE HERE`

Produces:

IMAGE HERE

See Appendix #x for additional examples.

Recommendation

- The best mitigation is to validate the user input format against what is expected; all other input should be forbidden. For example, if only numerical input is expected, an input containing anything other than numbers should be forbidden. If a strictly defined format is expected, validate it and do not allow anything that does not conform. Regular expressions (regexp) are a great solution for this.
  - Example of an expected request:
    - "shareImage" input should be sent through a URL validation filter/library or custom regular expression.

  - Example of a regexp to validate: `^https:\/\/WEBAPP1\.imgsvr\.net\/prod\/[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{12}\/shareImage\/[0-9]{13}\.(png|jpg|etc)$`
- All input should be encoded prior to storage to ensure the browser, or any content consumer, does not attempt to render HTML code and scripts.
- In addition to the above, black lists, WAF protections, etc. should be used to block known malicious input. It is important to note, these protections are not to be considered "primary" protections; rather, they should be seen as an extra layer of protection. These are a poor layer of protection compared to resolving the underlying web app injection flaw, as any number of things could cause the protections to fail or be bypassed.
- See the following resources for more information:
  - https://wiki.owasp.org/index.php/Cross-site_Scripting_(XSS)
  - https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
  - https://portswigger.net/web-security/cross-site-scripting
-

FR-012: SQL Injection

Category: Input Validation
Mitre ATT&CK: Exploit Public-Facing Application
Testing Standard: OWASP OTG-INPVAL-005
Localization: API
Total Calculated Risk: 10.0 - Critical
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N

Details
[web app] is vulnerable to SQL Injection (SQLi) through multiple user input fields. PEN Consultants was able to dump the entire database (DB) through SQLi.

"SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior. In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack." [PORTSWIGGER-03]

The following is a timeline view of our discovery and exploitation of the vulnerability.

During the initial familiarization phase of testing, we noticed what appeared to be raw SQL being sent as a variable through the subCon GraphQL query.

IMAGE

NOTE: The number of fields in this query was reduced to minimize the data returned and the limit set to "1" as to only receive one subscriber back in response.

The above query returned one subscriber.

IMAGE

Next, we added a "--" (SQL comment characters) after the "created_at DESC".

IMAGE

That returned a list of all subscribers.

Based on these results, it appeared as if the SQL comment character was affecting the back-end query (i.e. the offset/limit portion was ignored). At this point, we speculated that a SQLi injection was probable, but we need more information about the back-end schema.

IMAGE

Next, we determined the number of columns in the table/view being referenced. We started with a request to sort on column “1” (the first column).

IMAGE

That was successful - we saw the subscribers returned. Next, we sorted on column 20 and got the same result, indicating there were at least 20 columns. We repeated this process to determine there were 25 columns in the table/view being referenced. If we attempted to sort on column 26, or anything larger, we received an error.

IMAGE

After determining the number of columns, we attempted to find the column names. We started by replacing the original “date” with “email”.

IMAGE

That was successful - a list of subscribers was returned. To ensure things were working as expected, we tried a known invalid column name.

IMAGE

That gave us the same error as before - when we attempted to sort on a column >25 - indicating that it was an invalid column name. By brute forcing (AKA field stuffing) this value, we determined all of the valid column names.

IMAGE

Note: The ~4k responses indicate a valid column name, while the ~1.3k responses indicate an invalid name

By repeating this attack with a much larger customized list - 35k words based on keywords from the web app - we collected all 25 back-end table/view column names.

LIST OF COLUMN NAMES

Note: Given that the field names are NOT case sensitive, the case may be different than seen in the back-end.

Similarly, we were able to determine the table name of each of the above column names by brute forcing it as seen here:

IMAGE

The next step was to build a baseline SQL statement to perform what is known as a blind boolean based SQL injection. The idea is to build a query that gives a known response when it evaluates to True and a different response when it evaluates to False.

Through some trial and error, we came up with the following query which returns a given subscriber (subscriber A) when the query evaluates to True. It does this by specifying a different sort column to the query based on the outcome of an expression we provide. When the expression evaluates to True, the list of subscribers is sorted by email address, which returned subscriber A since "a" is the first in the sorted list.

IMAGES

When we changed this SQL statement to be False (see query below) the subscribers are now sorted by language (it could have been just about any other field) instead of email, and, therefore, the subscriber A is no longer listed. In this case, subscriber B is listed.

IMAGES

With this behavior, we could substitute any valid SQL statement in the place of the "1=1" and have a known response for True and False. Thus, the reason this is known as a Blind Boolean Based SQLi. The reason it's "blind" is because this limits our ability to simply return a string value with a select statement (for example), bit instead, we must rely on a known response for a True vs False expression.

Although it is a slow process, this vulnerability allows us to guess a stored value, one character at a time, until we have retrieved the entire value. As an example, one of the first enumerations performed was to determine the database names. In MySQL/Maria DBs, those DB names are found in a special table called "INFORMATION_SCHEMA". By using a SQLi query like the following, we can determine if the first character of a given DB name is greater than the ASCII value of 78 ("N"):

IMAGE

Depending on whether that comes back as true or false, we select another ASCII number (i.e. another letter) and issue a second query to determine if it is true or false. Based on this, we now have a range of characters the first letter could be. By issuing repeated requests, we narrow it down to just one possible character. Once we have identified the first character, we repeat this process for the second character, and so on until the full DB name is determined. On average, each letter takes 7 or 8 requests to guess using this method. A 20 character DB name would take ~150 requests.

This process, if performed manually, is painfully slow and tedious. In order to speed up exploitation, we switched to a well known and trusted SQLi exploitation tool called sqlmap. This allowed us to quickly carry out other SQLi attacks that would have otherwise taken several days, or even weeks or months. Additionally, this is the tool many attackers use against applications vulnerable to SQLi.

Sqlmap can use a variety of techniques to exploit SQLi vulnerabilities. The web app, as mentioned above, was found to be vulnerable to "boolean-based blind", which can be read about here: https://github.com/sqlmapproject/sqlmap/wiki/Techniques.

Although sqlmap can often automatically detect the vulnerability using built-in capability, because of the nature of this SQLi vulnerability, we had to create a custom payload to direct sqlmap to the appropriate location and tell it how to interpret the responses.

```
<test>
     <title>PENConsultants</title>
     <stype>1</stype>
     <level>1</level>
     <risk>1</risk>
     <clause>3</clause>
     <where>3</where>
     <vector>(case when ([INFERENCE]) then email else language end) ASC OFFSET 0 LIMIT 1;
--</vector>
     <request>
         <payload>(case when (1=1) then email else language end) ASC OFFSET 0 LIMIT 1; --
</payload>
     </request>
     <response>
         <comparison>(case when (1=2) then email else language end) ASC OFFSET 0 LIMIT 1;
--</comparison>
     </response>
</test>
```

Notes about the payload above:

- "vector" provides the template query that is to be made
- "[INFERENCE]" is a place holder for all SQL statements that will be issued by sqlmap
- The "payload" and "comparison" strings help sqlmap understand what a True and False response should look like.

Once that basic payload was created, the first action performed with sqlmap was to enumerate all database names: python3 sqlmap.py -r sqli.txt -p -proxy http://127.0.0.1:8080 --technique=S –dbs

IMAGE – DATABASE LIST

Note: only the sqlmap and command (ex. "--dbs") will be shown from here on

Next, we determine the current username: sqlmap--current-user

IMAGE – CURRENT USER

Next, we collected the DB usernames: sqlmap –users

IMAGE – USER LIST

Next, we collected a list of tables: sqlmap -D [DB name] --tables

IMAGE – TABLE LIST

Now that we know the table names, we determined the column names of each table: sqlmap -D [DB Name] -T [table name] --columns

IMAGE – COLUMN LIST

Before we dumped the data in the table, we used the following command to determine the number of rows: sqlmap -D [DB name] -T [table name] --count

IMAGE – COUNT

Finally, we exported the data in the data column of the "[table name]" table: sqlmap --dump -D [DB name] -T [table name] -C [column name]

IMAGE – DATA DUMP

Dumping ID and email of 20 rows from subscribers table: `sqlmap –D public –T subscribers –C id,email --dump --start 100 --stop 120`

IMAGE – DATA DUMP

Dumping ID and password_digest of 20 rows from legacy_users table: `sqlmap –D public –T legacy_users –C id,password_digest --dump --start 100 --stop 120`

IMAGE – DATA DUMP

Kudos for using bcrypt/blowfish in the back end! One of the best choices.

By repeating this process for other tables, we were successful at dumping all relevant data in the back end DB.

In addition to the features shown above, sqlmap also allows us to drop into a pseudo DB shell or OS shell:

IMAGE – SHELL

As can be seen, this SQLi vulnerability allowed for a full compromise of the back end databases associated with X.

Cracked Password Hashes

In addition to compromising the DB itself, we were able to crack 18% of the small sample of hashes collected (above) using a top-100k passwords list. Given the high percentage of users who use the same password across multiple accounts, it is likely we could have compromised

other 3rd party accounts belonging to those users as well (ex. email, bank accounts, etc.). These results highlight the fact that ~18% of your users use extremely basic passwords that are susceptible to not only dictionary attacks against the password hashes, but also to online password attacks (ex. password spraying).

Note: It should go without saying, we did not store cracked passwords, nor did we attempt usage of the passwords in any way.

Recommendation

- Note: The web app does appear to be performing input sanitization fairly well.
  - Of all the other inputs tested, the "direction" field is the only one we were able to exploit. That's not to say there are not others that are vulnerable.
  - A full evaluation should be performed with the assumption that all input is potentially exploitable.
  - Since this was black box testing, we are unable to determine if current methods of input sanitization are using a black list or white list approach (or both), and, therefore, we are unable to determine what failed to catch our attack.
  - Based on source code review and testing, many endpoints use SQL prepared statements. This, however, does not mean that there are not undiscovered vulnerable endpoints and parameters.
- As mentioned in FR-0xx, isolate each database by creating unique database credentials for each client and set DB permissions appropriately. That would have limited our ability to gain access to all of the DBs.
- Keep software updated
  - SEE VULNERABLE 3RD PARTY SOFTWARE RECOMMENDATIONS
- Reevaluate Vulnerability Management policies
  - SEE VULNERABLE 3RD PARTY SOFTWARE RECOMMENDATIONS
- Do not send raw DB error messages back to the user. Properly handle each anticipated exception and give the user a generalized, but informative, error message that helps in understanding what went wrong without exposing back-end syntax. If a specific exception is not accounted for in code, the default exception would give a general "error", and would not give away raw back-end error messages. Note: This does not mitigate timing based attacks, but is usually a quick/easy patch until more thorough mitigation can be put in place.
- The best mitigation is to validate the user input format against what is expected; all other input should be forbidden. For example, if only numerical input is expected, an input containing anything other than numbers should be forbidden. If a strictly defined format is expected, validate it and do not allow anything that does not conform. Regular expressions (regexp) are a great solution for this.
  - Example of an expected request:

    - "order" should only ever have a [valid column name followed by "asc" or "desc" | number] as the value.
- Implement SQLi protections, and input validation in general, using common utilities/libraries in the language used for the web app.
- In addition to the above, black lists, WAF protections, etc. should be used to block known malicious input. It is important to note, these protections are not to be considered "primary" protections; rather, they should be seen as an extra layer of protection. These are a poor layer of protection compared to resolving the underlying web app injection flaw, as any number of things could cause the protections to fail or be bypassed.
- More information on protection against SQLi, and on input validation in general, can be found at:
    - https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
    - https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html
- The high number of password hashes cracked highlight the need to employ a password strength policy - to prevent users from using such weak passwords. See FR-0xx for more details.

FR-013: Missing Security Headers

Category: Configuration Management
Mitre ATT&CK: Exploit Public-Facing Application
Testing Standard: OWASP Security Headers
Localization: example.org
Total Calculated Risk: 4.6 - Medium
CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:N

Details

The majority of OWASP recommended security headers are missing. These HTTP response headers should always be present in web server responses to prevent attacks. [https://wiki.owasp.org/index.php/Top_10-2017_A6-Security_Misconfiguration] That guidance can be see at: https://wiki.owasp.org/index.php/OWASP_Secure_Headers_Project#tab=Headers

The following sections will describe the impact of each missing header.

HSTS - prevents MiTM and downgrade attacks

A missing or misconfigured HTTP Strict Transport Security (HSTS) header on web servers can lead to Man-in-The-Middle (MitM) attacks. HSTS "will prevent any communications from being sent over HTTP to the specified domain and will instead send all communications over HTTPS." [https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html][https://www.owasp.org/index.php/Top_10-2017_A3-Sensitive_Data_Exposure]

Here is an example of a properly configured HSTS header:

```
$ curl -k -sD - -o /dev/null https://PENConsultants.com | grep -i strict
strict-transport-security: max-age=31536000; includeSubDomains; preload
```

Here is an example from the CLIENT web server that does not have an HSTS header:

IMAGE – same thing as above, but showing in-scope domain and missing header (i.e. blank line after command)

Note: The max-age should be set to at least 31536000 seconds (1 year). Having a low max-age will cause the cached time to expire so quickly that the header will, for all practical purposes, provide no protections. [https://hstspreload.org/] [https://blog.qualys.com/securitylabs/2016/03/28/the-importance-of-a-proper-http-strict-transport-security-implementation-on-your-web-server]

Additionally, the preload directive should be included in the header and the parent domain (example.org) submitted to the preload list for inclusion. An active HSTS header offers little

protection for first time visitors to domains that are not included on this list, as the browser is not yet aware of the mandate for HTTPS and an attacker with MiTM access will simply strip the header from any response.

IMAGE OF HSTS PRELOAD SITE ERRORS

The following are additional, out-of-scope, examples noted during testing:

- LIST HERE, IF ANY

X-Frame-Options or CSP - prevents clickjacking

Clickjacking is “when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page...and [capture] keystrokes. Thus, the attacker is 'hijacking' clicks [and input] meant for their page and routing them to another page, most likely owned by another application, domain, or both". [https://owasp.org/www-community/Clickjacking]

As an example, PEN Consultants created an attacker website that dynamically embedded the X page within our website. As can be seen in the following screenshots, a user is unlikely to notice the (invisible) iframe we used to embed the legitimate page in ours.

IMAGE(S) SHOWING ATTACK BUTTON AND INPUT ABOVE WEB APP’S

In the above example, the overlayed buttons and input fields have purposefully been made to look different and be of a different size for demonstration purposes. In a real attack, these would match the precise format and location of the legitimate buttons and input fields as to capture what the victim was entering and remain undetected.

By attacking a user with a crafted page such as this, the data the user assumes they are entering into the legitimate web app will actually be sent to the attacker instead. The attacker could pass along the collected values to the legitimate web app so the user is not tipped off to the interception of the data, or the attacker could simply redirect the user to the legitimate page and the user would assume the initial submission failed.

By including the response header `X-Frame-Options: DENY`, the browser is instructed to not allow embedding of one page within another, as demonstrated above.

In addition to this header, you may also want to include the more modern Content-Security-Policy (CSP) header. Example: Including the CSP header with a directive such as `frame-ancestors 'none'` will also prevent iframes. OWASP recommends that both headers be utilized as not all web clients support CSP.

Note: The attacker code used for the above PoC has been included in Appendix #2.

X-XSS-Protection - helps prevent Cross Site Scripting

"Cross-site scripting (AKA XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. XSS vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all of the application's functionality and data". "The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes."
[https://portswigger.net/web-security/cross-site-scripting]

The `X-XSS-Protection` header was historically used by browsers to enable protections that help prevent well known XSS attacks. In recent months, many major browsers have dropped support for this feature as it was never as effective as the industry had hoped. The core problem was the XSS protection engines running in the browsers were typically easy to bypass. However, it is still a good practice to enable it, as there were no negative effects to add the header, and it could stop some attacks.

Even though most major browsers have dropped support, some older web clients (ex. IE, Safari, etc.) still support this header. As of 2020, OWASP still recommends the inclusion of this header in all server responses. [https://owasp.org/www-project-secure-headers/]

In present day, the Content-Security-Policy (CSP) header is used to help mitigate XSS attacks (among other things). As an example, OWASP recommends a header similar to the following to help mitigate some of the risks associated with XSS:

```
Content-Security-Policy: script-src 'self'
```

The CSP header must be carefully implemented, as it can break legitimate functionality if that functionality is not properly white listed. More about setting up CSP can be found here: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

Note: OWASP recommends that both headers be utilized as not all web clients support CSP.

X-Content-Type-Options - prevents MIME (or content type) sniffing attacks

Web clients, at times, inspect the response from a server and attempt to determine the content type of the response in order to know how to process it or render it for the web client (ex. a browser). Most commonly, this "MIME Sniffing" is performed when the server fails to send a

Content-Type header that specifies the content type. Unfortunately, many web clients still perform MIME sniffing even if the web server declares it, and at times, may even override what the server has declared in an attempt for user friendliness (i.e. ensuring the content is properly rendered for the end-user).

This behavior can have serious security consequences when the response data contains user input, be it stored or reflected. Example: A user may upload an "image" that is actually a JavaScript file. In addition to the user input sanitization concerns with this scenario, JavaScript will be executed if the web server does not declare the content type to the browser when sending back the "image", or if the browser overrides the declared type (especially since the difference between a JS file and an image is obvious).

This same attack scenario can utilize not only other file types, but also basic user input. Example: The name field of a user profile, description, comment, or anything else the user can control is sent back in a JSON response. If the user (or attacker) inserts JavaScript (for example) into one of those fields, the JSON response containing that malicious code could be interpreted by the web client as JavaScript, even if the server has declared it to be JSON.

To mitigate this, all major web clients support a response header from the server that instructs the client to disable this MIME Sniffing feature. All web server responses should include `X-Content-Type-Options: nosniff`, which will instruct the web client to disable this feature.

Another important aspect on this topic is to set the Content-Type header on all server responses. Not declaring the content type in responses, while simultaneously instructing the browser to not sniff for the content type, can have negative consequences. Note: All responses were observed to contain the Content-Type header during testing.

Referrer-Policy - prevents disclosure to 3rd parties

Server responses are missing the "Referrer-Policy" HTTP response header, which instructs the browser on how to handle the "Referer" and "Origin" headers when requesting resources from 3rd parties. This allows the 3rd party to discover website portals that you may not want to broadcast. Although not observed during testing, in some cases, this can also expose sensitive parameters to the same 3rd party.

Example request: https://platform.CLIENT.com/

IMAGE OF RESPONSE HEADER SHOWING MISSING OR REFERRER-POLICY MISCONFIG HEADER

That requested page contains an external reference to a CSS resource at Google:

IMAGE SHOWING REFERENCE TO 3RD PARTY URL IN CODE (ex. stylesheet)

When the browser requests that 3rd party JS resource, it leaks the full URL, and any potential URL parameters, from the page along with the request. In this example, it is disclosing a user ID to the 3rd party:

IMAGE SHOWING SENSITIVE DATA BEING SENT TO 3RD PARTY VIA REFERER OR ORIGIN HEADERS

Another concern is the general fact that the web app require the browser to fetch resources from a 3rd party website as opposed to them being hosted on the CLIENT servers. This has various performance and security implications. The performance and security of your website is dependent on the performance and security of the 3rd party's site. In this case, Google is likely one of the most secure and high performance 3rd party options, but it is still an unneeded risk.

It is best practice to host all resources on your domain to ensure security and performance scope is maintained. [https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/loading-third-party-javascript]

Expect-CT - prevents rogue TLS certificates

In 2013, the Certificate Transparency (CT) standard was created. CT is a publicly accessible, append-only, cryptographically assured record (Merkle tree) of all TLS certificates issued by a Certificate Authority (CA). This offered a level of monitoring for attacks after a string of CA compromises, between 2000 and 2015, which allowed attackers to comprise encrypted traffic to/from many high profile web sites – Facebook, Google, government websites, etc. Since 2018, all major CAs have complied with CT standards. Note: Several CAs were posting CT records prior to 2018. The goal was to create the ability for TLS certificate owners to monitor all certificates issued across all CAs, that participate in the standard, and become alerted to rogue certificates being issued by a compromised CA.

As of 2018, about half of the major web clients (ex. web browsers) support the Expect-CT header, which is a directive to the web client to confirm the certificate being used by the web server is listed in one of these public CT logs before establishing a secure connection. If the CT record is not found, then it can be assured that the certificate did not originate from a reputable CA and that the certificate/web server owner would be unable to verify it. In other words, an attacker is likely performing a man-in-the-middle (MiTM) attack and the "secure" connection to the server is not actually secure.

As of 2020, other major web clients continue to add support for this header, and some even enforce CT checks by default, regardless if the server sends the header. Even though this header is supported in only a subset of the major web clients, it is still advisable to include it in all HTTP response headers.

Another weakness of this security measure is that not all TLS certificate owners monitor the CT records, and, therefore, are never alerted to these rogue certificates being issued. Because of this, it is imperative to not only implement the Expect-CT header, but to also monitor public CT records for unexpected certificates being issued for your domain(s).

Cache-Control - prevents unauthenticated data access

The absence of cache related headers causes a web client (and intercepting proxies) to semi-permanently store sensitive content retrieved from the web app to the local disk. This makes it trivial for an attacker, or another user on a shared computer, to gain access to sensitive information without the "need to know the username and password of the user to steal the information". [https://owasp.org/www-community/OWASP_Application_Security_FAQ]

Specifically, the following OWASP recommended HTTP headers are missing:

- Cache-Control: no-store
- Expires: 0
- Pragma: no-cache
  - Especially if HTTP/1.0 is used (rare)

Although all responses should contain cache directives, the caching of responses containing financial transaction data was most concerning. Here is an example of cached data accessible on disk that was accessible without authentication long after the session ended.

IMAGE

Recommendation

Ensure the following headers and directives are set in all HTTP responses from the server in order to mitigate the risks and attacks shown above:

- HSTS
  - `Strict-Transport-Security: max-age=31536000; includeSubDomains; preload`
  - Check HSTS using https://hstspreload.org/ and correct errors. Once error free, submit the domain for "preload" to ensure new users are protected from MiTM attacks.
  - For more details, see:
    - https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

        - https://portswigger.net/kb/issues/01000300_strict-transport-security-not-enforced
- Clickjacking
    - `X-Frame-Options: DENY`
    - Note: At one time, “frame busting” code, usually JavaScript, was used to help prevent this type of attack (ex. parent.location = self.location). Those techniques are no longer considered trustworthy as they are trivial to bypass. In light of the browser support for the above two mentioned headers, this technique only introduces additional development work while providing no additional security.
    - If possible, also enable the CSP header with appropriate directives to prevent this behavior.
        - `Content-Security-Policy: frame-ancestors 'none';`
        - Note: You will likely want to include both of these headers as to support web clients that do not recognize the CSP header.
    - For more details, see:
        - https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html
- XSS
    - `X-XSS-Protection: 1; mode=block`
    - If possible, also enable the CSP header with appropriate directives to prevent untrusted scripts from running in the context of your web app.
        - More about setting up CSP can be found here:
            - https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
            - https://blog.sendsafely.com/using-content-security-policy-to-prevent-cross-site-scripting-xss
            - resource on configuring the header in IIS: https://scotthelme.co.uk/hardening-your-http-response-headers/#x-xss-protection
    - For more details, see:
        - https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection
- MIME Sniffing
    - `X-Content-Type-Options: nosniff`
    - For more details, see:
        - https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options
- Disclosure to 3rd parties
    - `Referrer-Policy: no-referrer`
    - Include the “Referrer-Policy” header set to “no-referrer” or “same-origin”. References:
        - https://wiki.owasp.org/index.php/OWASP_Secure_Headers_Project#rp
        - https://aws.amazon.com/blogs/networking-and-content-delivery/adding-http-security-headers-using-lambdaedge-and-amazon-cloudfront/

- Consider hosting all 3rd party resources on the app's domain/web server, where possible.

- Rogue TLS Certificates
  - `Expect-CT: max-age=86400, enforce, report-uri="https://example.org/report"`
  - Monitor the public CT records for unexpected certificates being issued. Example: Hourly checks with automated alerting when a new certificate is observed.
    - Option A: Depending on the services you use for creating certificates or hosting your web app, one or more of your current providers may include this monitoring and alerting service. It will typically be called something such as "Certificate Transparency Monitoring" and is often included for no additional cost.
    - Option B: Sign-up for a monitoring service
      - Free: https://support.cloudflare.com/hc/en-us/articles/360031379012-Understanding-Certificate-Transparency-Monitoring
      - Free: https://developers.facebook.com/tools/ct/
        - Oddly enough, this has proven to be one of the best free options available. They will even send alerts for sub-string matches of a monitored domain. For example, here is a recent alert we received on our domain, penconsultants.com, that was found within the following domains:

Domains	Subject	Issuer
penconsultants.com.eu.cas.ms penconsultants.com.admin-us.cas.ms penconsultants.com.admin-mcas.ms penconsultants.com.admin-us3.cas.ms penconsultants.com.admin-eu2.cas.ms penconsultants.com.us2.cas.ms penconsultants.com.admin-eu.cas.ms penconsultants.com.eu2.cas.ms penconsultants.com.us.cas.ms penconsultants.com.admin-us2.cas.ms penconsultants.com.mcas.ms *.mcas.ms penconsultants.com.us3.cas.ms	CN=*.mcas.ms	C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, OU=Microsoft IT, CN=Microsoft IT TLS CA 2

      - Free (reports sent from browsers, not CT logs): https://report-uri.com/products/certificate_transparency_monitoring
      - Paid: https://sslmate.com/certspotter
    - Option C: Download or create a simple script that monitors and alerts
      - https://github.com/SSLMate/certspotter
  - For more details, see:
    - https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expect-CT
    - https://wiki.owasp.org/index.php/OWASP_Secure_Headers_Project#ect

- Cache directives
  - All three headers:
    - `Cache-Control: no-store`
    - `Expires: 0`

        - `Pragma: no-cache`
            - Especially if HTTP/1.0 is used (rare)
    - For more details, see:
        - https://wiki.owasp.org/index.php/Testing_for_Browser_cache_weakness_(OTG-AUTHN-006)
        - https://aws.amazon.com/premiumsupport/knowledge-center/prevent-cloudfront-from-caching-files/
- General
    - Ensure all future development maintains the recommended, and properly configured, HTTP response headers.

FR-014: Upload of Malicious Files

Category: Business Logic
Mitre ATT&CK: Exploit Public-Facing Application
Testing Standard: OWASP OTG-BUSLOGIC-001
Localization: Web Application
Total Calculated Risk: 7.1 - High
CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:L

Details
There appears to be little to no server side validation or restrictions in place for file uploads through any of the file upload functionality (ex. profile avatar, business related documents, etc.), nor AV scanning of files, which could allow an attacker to compromise employee, client, and other user systems in some scenarios.

The file upload content type restrictions for member message attachments are easily bypassed by simply changing the file extension. As an example test, we took calc.exe, changed the extension to calc.txt, successfully uploaded it, and then downloaded it. As can be seen, the uploaded file was stored as-is, as a binary executable.

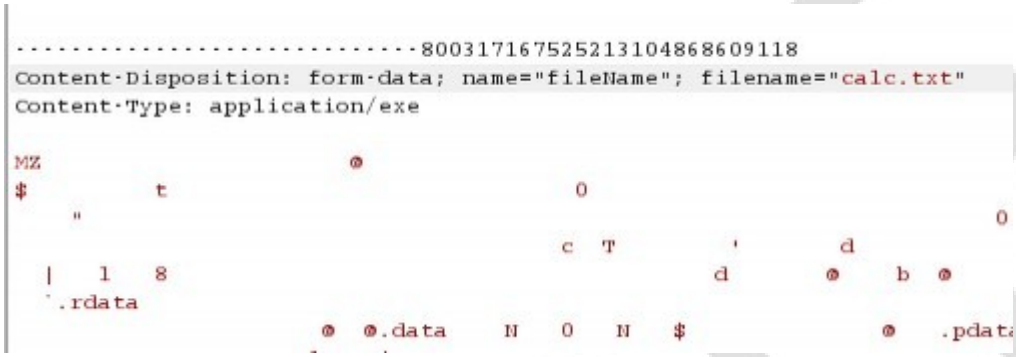
```
root@kali017:~/Downloads# file calc.txt
calc.txt: PE32+ executable (GUI) x86-64, for MS Windows
root@kali017:~/Downloads# xxd calc.txt |head
00000000: 4d5a 9000 0300 0000 0400 0000 ffff 0000  MZ..............
00000010: b800 0000 0000 0000 4000 0000 0000 0000  ........@.......
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 f000 0000  ................
00000040: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468  ........!..L.!Th
00000050: 6973 2070 726f 6772 616d 2063 616e 6e6f  is program canno
00000060: 7420 6265 2072 756e 2069 6e20 444f 5320  t be run in DOS
00000070: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000  mode....$.......
00000080: e0ca 74e6 a4ab 1ab5 a4ab 1ab5 a4ab 1ab5  ..t.............
00000090: add3 8fb5 a6ab 1ab5 add3 9eb5 8aab 1ab5  ................
```

To confirm the file was unmodified from the original, a checksum was taken.

```
10e4a1d2132ccb5c6759f038cdb6f3c9  calc.exe
10e4a1d2132ccb5c6759f038cdb6f3c9  calc.txt
```

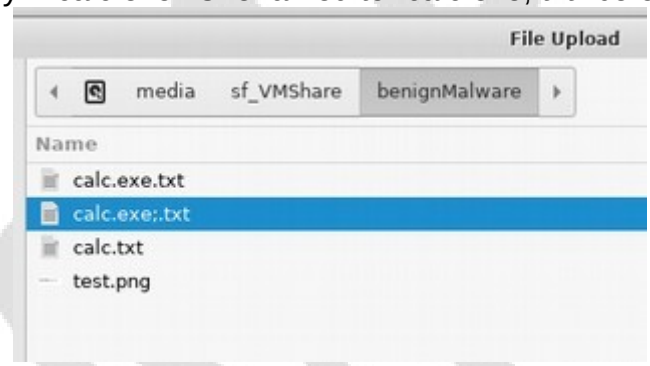
Through additional testing, it appears the content type sent with the upload is not checked by the web app. As a test, we uploaded a file content type of application/octet-stream (i.e. a Windows Exe), but claimed it was text/plain. The server accepted it. We performed several other mismatched file types and given content types as well. No error was ever given, and the file was always stored properly.

The fact that content types, such as application/exe, were allowed at all indicates the web app is not checking these. White listing on the web app side appears to be based on extension alone.



Another method was identified that allows for an attacker to easily bypass file upload restrictions without the use of special tools, and do it in such a way that maintains the malicious extension (ex: .exe).

In this example, we were able to upload and host an executable with a .exe extension. The first step was to name the executable as “[filename].exe;.txt” on disk. As before, we use “calc.exe” to simulate a malicious binary. “calc.exe” is renamed to “calc.exe;.txt” before uploading.



Next, we uploaded the file and attached it to a message.

IMAGE HERE

By clicking on the following link in the message center, the “;.txt” is ignored and a properly name “calc.exe” is downloaded.

IMAGE HERE

This technique could be very dangerous to employees who may not recognize the danger and "run" the executable, thus giving an attacker access to your network. As mentioned in FR-0xx, this technique could also be leveraged to host malware from your web app and target employees and members as well.

As part of continued testing on this functionality, benign malware testing samples were uploaded to determine if the files would be stored and made available as opposed to quarantined by an anti-virus/malware solution. For testing, malware test files obtained from EICAR (https://www.eicar.org/?page_id=3950) were uploaded as message attachments. All security vendors are supposed to identify these as malicious. Testing demonstrated these were NOT quarantined when uploaded and continue to remain available. This indicates that no anti-virus/malware scanning is being performed against file uploads, or the current solution is not functioning.

IMAGE HERE

IMAGE HERE

Recommendation

- The (suspected) white list approach to file extensions is great. However, as a second verification step, the app should check the content type being passed. And finally, as a third verification, the app should check the "magic bytes" of the file content to verify permissible content.
- Ensure content type matches the given extension. For example, don't allow an image content type when the extension is ".txt".
- Locate and correct the faulty code that allowed the ".exe;.txt" extension bypass to work and fix it.
- Consider flattening all documents to an image or PDF using an automated process so no "active code" makes it to employees. There are several allowed content types/extensions that could still contain malicious code. Examples: Microsoft Word documents could contain malicious macros, maliciously crafted PDFs, etc. By automatically converting all

documents to an image (for example), it will ensure malicious macros (for example) never make it to an employees' workstation through member messaging.

- Traditional anti-virus/anti-malware solutions have a low probability of catching modern malware. However, they do still catch malware and should, therefore, be used. Ensure all uploaded documents are scanned for malicious code and content before being stored and presented to users or employees.

FR-015: Phishing Assessment

Category: Attack
Mitre ATT&CK: Initial Access
Testing Standard: Social Engineering
Localization: Users
Total Calculated Risk: 8.2 - High
CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:L/A:N

Details

On 06 May 2018 @ 0810, PEN Consultants carried out an email based phishing assessment against [CLIENT NAME]. As seen in a previous finding (FR-0xx - Brute-Force User Enumeration), we were able to generate the list of targets on our own. In all, 86 unsuspecting users were targeted, along with our trusted insider.

The phishing campaign impersonated the President and CEO, John Doe:

IMAGE OF EMAIL

The link, once clicked, requested the user to enter their name and email address into a form on our malicious web server:

IMAGE OF LANDING PAGE

After that, it prompted the user to enter their corporate username and password using an HTML/CSS based popup that closely resembles the built-in Windows 10 password dialog:

IMAGE OF POP UP

This type of phishing, although unsophisticated and only semi-targeted, requires a minimal amount of research by the attacker. Given that it impersonates a senior officer in the company, it can greatly increase the click-through-rate (CTR). Additionally, we registered a domain name that resembled examplle.org - notice the extra “l” in the URL in the first screenshot.

These techniques, in conjunction with the vulnerability discussed in FR-0xx (Partially Open Mail Relay), led to the following metrics:

- 87 - users who received the email
- 6 (7%) - Outlook downloaded the tracking image
  - It’s interesting to note that most Outlook clients did NOT download the tracking image as we know with certainty the number of users who read the email was at least 26%. The 7% that did have their Outlook activate this external tracking image was either due to a permanent setting (ex. “always trust external images”) or the user clicked on “show images” for this one email (this seems less likely).
- 23 (26%) - CTR - Clicked the link and visited our external page
  - Had PEN Consultants being trying to actively exploit users visiting the site (ex. browser exploits), there’s a potential this many workstations could have been compromised.
- 17 (20%) - Entered their name and email address
  - Although this information is not necessarily that sensitive, it demonstrates that users are willing to enter at least some data into an improperly vetted system.

- 11 (13%) - Entered their username and password
  - Using this information, PEN Consultants was able to gain access to these users' accounts (ex. OWA, VPN, etc.).
- Note: Raw results can be seen in Appendix #x

Although not in scope for the testing, it is important to note the following social engineering techniques used in many of the recent publicly disclosed data breaches:

- more sophisticated phish emails - ex. leverage known vulnerabilities (vulnerabilities in the browser, adobe reader, java, etc.)
- phish are tailored to the individual user, not just the organization - i.e. spear phishing
- attempts to install malware when the user navigates to the page - RAT, keylogger, etc.
- takes many forms - SMS, phone, social media, media drops in the parking lot, etc. in addition to email
- etc.

As can be seen, phishing, and social engineering in general, are an extremely powerful vector attackers can leverage to breach your network and data. According to the 2018 Verizon Data Breach Report, 74% of data breaches start with an attacker sending a phish email to compromise one or more systems.

Notes:

- Good Spam Filter:
  - The day before, at approximately 10am, PEN Consultants performed several tests against an [CLIENT NAME] test email account that our trusted insider gave us access to. Within minutes, [vendor name] servers began scanning our phishing page as we had not yet implemented network ACLs to block non-CLIENT access. By 13:15, [vendor name] had categorized our IP as suspicious, causing our emails to be sent to the Junk folder.
  - Our trusted insider white listed the IP just before the phishing assessment in order to allow the emails to come through the Inbox. Otherwise, we would have had to change IPs and ensure our network ACLs prevented access by [vendor name] servers.
  - Note: OWA still showed the emails in the Inbox even after being classified as suspicious.
- Observant and pro-active Admin:
  - At 0834, 24 minutes after the phish emails went out, a system administrator, who was unaware of the testing, sent out a warning to all users to not click the link as it was a malicious email.
  - At 0857, our trusted insider replied-all to the email confirming it was an exercise.
    - The email exchange can be seen in Appendix #x.
  - All but the last two recorded clicks were before 0855 - within the first 45 minutes. Two additional users still clicked the link around 1300 (4 hours later). One of those users entered their data.
- Our overall impressions:
  - The recipients did pretty well considering this type of testing had not been performed before.

- In one respect, the metric "26% CTR" is meaningless as it only takes one user/workstation compromise to breach a corporate network. In another respect, it does give you a general sense of how observant your users are and how well they identify/avoid phish email.
- According to multiple sources (knowbe4.com, csoonline.com, etc.), the industry average CTR for first time testing is 27-31% with 15-17% of those users entering their password, so, [CLIENT NAME] is within a normal range.

Recommendation

Attempting to achieve a 0% user click-through-rate (CTR) for every test or campaign is not practical. It is unlikely for an organization to maintain a 0% CTR during a semi-targeted phishing campaign, and nearly impossible for a highly-targeted spear phish (ex. establishing a dialog with the victim before requesting an action). Due to new employees who haven't received your full training, the sophisticated nature of some phish, accidental clicks, or just someone having a "bad day", you can count on a subset of users falling for a phish during a targeted campaign.

Here are some recommendations that can help reduce the CTR, but more importantly, the risks associated with phishing:

- Create a group policy to block remote images by default in Outlook (at minimum).
- Consider a group policy that prevents the users from ever enabling remote images Note: This may impact business and/or cause an administration white listing burden.
- Determine why OWA was still sending suspicious emails to the Inbox as opposed to Junk like Outlook was. This is likely a simple configuration issue.
- Continually educate your users on the dangers of, and how to spot, phishing through security awareness training - both CBTs (computer based training) and in-person
  - The responsibility for users who fall for phish is often a shared one – insufficient training provided by the organization and careless users.
- Consider consequences for users who fall for phish in the future, be it real or testing:
  - out-of-cycle compliance training
  - meeting with a manager
  - temporary removal of accesses (ex. web access) and/or placement into a less risky job function
  - Other consequences some companies choose to use (PEN Consultants does not endorse these): monetary penalties and/or termination
- Consider rewards for users who quickly report phish, be it real or testing:
  - rewards would only be for the first X who report and/or those who report within X minutes, as time is of the essence with a phishing campaign
  - gift card, spot bonus, lunch with manager, public recognition, etc.
- Impart the perspective to your users that they are an integral part of the security of your organization, not a weak link that is likely to cause a breach...even though both are likely true.
  - You have heard it said that it only takes one person falling for a phish to cause a data breach. The inverse is true as well - it only takes one user's report of a phishing attack to foil an attacker and stop a breach. Although your users are often the "weakest link" in terms of security, they are also your front-line defense.

- Ensure the related recommendations in this report are addressed - namely “Brute-Force User Enumeration”, “Partially Open Mail Relay”, and “MFA”.
- If it does not already exist, create a robust reporting and tracking process for phish emails your users receive:
  - reporting button/plug-in for users to report suspicious emails
  - daily review process (or 3rd party) to examine reported phish
  - immediate notification process to warn users when an attack/campaign is detected
  - a playbook of [semi-]automated action steps to clean-up/remove phish email from users’ mailboxes when a campaign is detected
  - etc.
- Frequently test your users with at least one monthly social engineering campaign
  - This can be made into a fun contest with the above mentioned rewards
  - Track the results...especially repeat offenders who may need to be given extra training and coaching
  - Be sure to change it up every month: level of sophistication, phishing and spear phishing, payloads and actions, the form: SMS, phone, email, in-person, fax, mail, media drop, social media, etc.
  - DIY: Use free frameworks such as https://getgophish.com/ to perform testing and track results
  - Managed service: Alternatively, we can offer an affordable, standalone social engineering service or perform the testing as part of our Cybersecurity Unlimited service - https://penconsultants.com/cybersecurityUnlimited
- Great resources for additional information:
  - https://info.wombatsecurity.com/hubfs/Wombat_Proofpoint_2019%20State%20of%20the%20Phish%20Report_Final.pdf
  - https://apwg.org/trendsreports/

FR-016: Media Drop

Category: Gaining Access
Mitre ATT&CK: Initial Access
Localization: Human Behavior
Total Calculated Risk: 7.1 - High
CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:L

Details

The employees at [REDACTED] are highly susceptible to malware infection and loss of sensitive data due to their tendency to trust unknown foreign media.

On 24 Apr 2017, a media drop attack was performed. A media drop is when an adversary delivers CDs, flash drives, or other portable media to a victim via mail, in person, or simply dropping them on the ground. The results of our test were devastatingly successful.

Step #1 – We created a webpage that looked exactly like the [REDACTED] login page.

Step #2 - That page was copied to a blank CD, which was labeled "[CLIENT NAME] [REDACTED] Backup."

[PICTURE OF CDs]

Step #3 – Three CDs were dropped on the ground in three locations at 0530 on [DATE].

[TABLE OF DROP LOCATIONS]

Step #4 – We monitored which CDs were inserted into computers and which ones received passwords.

Based on the user agent strings, at least two separate computers were used to view two of the three CDs. Had this been an actual adversary, these computers could likely also be infected with malware, giving remote access, key loggers, etc. The only thing this test did was steal passwords.

[TABLE OF REDACTED PASSWORDS]

Recommendation

- Never, under any circumstance, put unknown, foreign media in a computer. Unless you are 100% sure of the manufacture, source, and purpose of the media, do not put it in your computer. Social Engineering is a common attack vector used by attackers to gain access to a network.
- Ensure no user is running with administrative privileges. Restricting your daily use account to a limited user account is more effective at preventing attacks that attempt to infect your system with malware than just about any other preventive measure, including antivirus.
- If you do go as far as to put media in your computer, certainly do not input any passwords.

- Ensure all employees take Information Assurance training immediately and periodically.
  This is discussed in more detail in the General Overall Recommendations Section.

FR-017: Unsupervised Access

Category: Attack
Mitre ATT&CK: Initial Access
Testing Standard: Social Engineering
Localization: Users
Total Calculated Risk: 7.3 - High
CVSS:3.1/AV:P/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:H

Details

We were unsupervised (escort left us) for lengthy periods of time at three sites, and briefly unsupervised at a fourth site. See the appendices associated with “W”, “X”, “Y”, and “Z” for details. At two of the four sites, we were left unsupervised while in the server area. One of those sites was in [CLIENT NAME]’s primary data room at HQ where we were left unsupervised for a total of 18+ minutes over 5 occasions. Had it been in scope, we likely would have had time to remove a number of data drives and place them in our tool bag in addition to the hardware implant we installed.

Recommendation

- Given that [CLIENT NAME] already has a written policy about remaining in supervision of visitors at all times, this is primarily an employee educational opportunity. For a more complex solution to this problem, read more here: https://penconsultants.com/pairedBadges
- For the technical controls that would help reduce the risk of this type of attack, see FR-0xx.

Attack Scenarios

Overview

PEN Consultants was able to compromise most CLIENTNAME core systems and services utilizing vulnerabilities that have been highlighted above in this report. The main purpose of this section is to help CLIENTNAME understand how even info and low level vulnerabilities aided in the takeover of these core systems/services/data, and how mitigating those lower level vulnerabilities could have prevented the demonstrated attacks, or other attacks, just as much as mitigating higher level vulnerabilities.

These findings have been grouped to demonstrate multiple unique attack chains and are labeled with the corresponding FR# from the report. Each attack chain is further grouped into three attack phases:

1. RE: Recon and Enumeration – information gathering
2. IA: Initial Access – gaining a foothold into the system or data
3. LM: Lateral Movement and Breach – compromise key systems, services, and data

Note:

- Some phases were “out-of-scope”. These were not performed for one or more reasons: not included in the testing scope, specifically forbidden by Client in the SOW, impractical to perform during our short testing window, or illegal to perform. All techniques listed are common and have been seen in recent breaches, as a real attacker does not operate under the typical restrictions that a testing company must work under.
- Some of these findings may require an authenticated session or internal access to the network. This access could come in several forms: a legitimate user becoming an insider threat, an external unauthorized attacker gaining access to a legitimate user’s account (i.e. ATO / account take over), piggybacking on an existing connection via their access to a comprised user’s computer system (i.e. remote access to the system), etc. Bottom line, although this attacker requirement would reduce the risk, it would not reduce it substantially. As an example, toggle between “none” and “low” under “privileges required” to see how much an example risk score calculation would change: https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:N. The reduction of risk due to the need to be authenticated is conversely increased some by the risks posed from FR-0xx and FR-0xx.

Attack #1 – SQLi

- RE: Authenticated, non-privileged, attacker discovers the SQLi vulnerability.
- IA: Attacker exploits the vulnerability using free and easy to use exploitation tools, such as sqlmap.
- LM: Through the access gained, attacker compromises all data from the back end database, may inject malicious code into service content in order to infect users’ endpoints (users & admins), could potentially use it to also gain access to other 3rd party resources (user email accounts, bank accounts, etc.), etc.
- References: FR-0xx, FR-0xx, FR-0xx

Attack #2 – Privilege Escalation

- RE: Whether the attacker is an external unauthorized user, or an insider threat, privilege escalation is a common objective of an attacker. An authenticated, non-privileged, attacker sees the “non-admin” in their user controlled session information, or uses any number of free and easy to use tools or browser plugins to see the API calls being made by the scripts active in the browser and the fact that the active user ID is being sent as a user changeable value to the API.
- IA: As demonstrated in the respective findings, exploitation is trivial in most cases. The attacker changes their role in their stored session information or changes the active user ID in the API calls and replays them.
- LM: Once a level of privilege escalation is achieved, that privileged access will be used to look for additional access as the attack progresses and additional privileges are obtained. Once the attacker gains access to all privileged functions and data, they can carry out any number of attacks, to include, cloning users’ MFA, password attacks to recover weak users passwords, denying other users access, changing other users’ passwords, creating new user accounts, deleting/modifying data, etc., much without the other users noticing the active attack.

- References: FR-0xx, FR-0xx, FR-0xx

Attack #3 – User Attacks

- RE: Misconfigured CORS headers and XSS are two of the most common items attackers look for. In fact, misconfigured CORS headers is a vulnerability that is typically found at a mass automated scale by attackers and 3rd parties selling that vulnerability identification to attackers.
- IA: In both cases, the attacker would need to entice the victim to navigate to a particular site, either through a phishing link, or a watering hole attack. Getting the user to navigate to a tenant would likely be easy, but injecting the script would require a little more work. The setup to exploit the CORS vulnerability is trivial, but may be a little harder to get the user to click on a phishing link or require favorable probability in getting the user to visit a 3rd party site with the attacker code.
- LM: With a CORS attack, the attacker would be exploiting the [web app] account/system through the user; whereas, with XSS, the attack could be just about anything client side scripting is capable of exploiting - the user's system, attacking 3rd party systems, etc.
- References: FR-0xx, FR-0xx

Attack #4 – MiTM

- RE: Attacker easily discovers the various crypto weaknesses in the web app and the potential for various MiTM and downgrade attacks, detects sensitive information from the disk cache of a shared/public computer system, authentication tokens in proxy logs or browser history, tokens in the referrer string in their web access logs, etc.
- IA: Attacker establishes a MiTM presence with one or more users – malicious proxy settings, evil public WiFi location, physically taping ISP line, ARP poisoning, etc., or, in some cases, the information obtained can just be opened directly (ex. PDFs and MP3s on disk), or the URL (with auth token), copied in the browser address bar.
- LM: Once a MiTM presence is established, if needed, attacker can easily trick a user into communicating over non-ssl (ex. sslStrip), or, with more effort, perform SSL downgrade attacks to cause the client to encrypt with a weak TLS cipher, in order to capture not only data, but user credentials. Depending on what is found, it is unlikely there would be much lateral movement from this attack other than potential identify theft and fraud related to the information that was exposed.
- References: FR-0xx, FR-0xx

Conclusion

PEN Consultants is honored to have been trusted with providing this testing service, and we hope you have found it valuable and actionable in keeping your systems/data secure.

We strive for 100% satisfaction, and we will make every reasonable effort to completely satisfy! Please share any concerns, feedback, or suggestions you may have.

Our business success is not only dependent on satisfied clients, but referrals as well. We would be appreciative of a written or video testimonial for public release and/or direct referrals. Examples of past testimonials can be found on our website: https://penconsultants.com/testimonials.

In addition to changes in your environment, the threat landscape is constantly changing. As such, it is vital for you to continue this type of testing and find new weaknesses that may arise. We appreciate you choosing PEN Consultants to serve you with these testing needs, and we would be honored to serve you again in the future.

References

- CENTRIFY-01: https://blog.centrify.com/verizon-compromised-credentials/
- DIGICERT-01: https://www.digicert.com/blog/depreciating-tls-1-0-and-1-1/
- GITHUB-01: https://raw.githubusercontent.com/berzerk0/Probable-Wordlists/master/Real-Passwords/Top304Thousand-probable-v2.txt
- ISI-01: https://resources.infosecinstitute.com/dangers-web-management
- NSCS-01: https://www.ncsc.gov.uk/blog-post/protect-your-management-interfaces
- OWASP-01: https://wiki.owasp.org/index.php/Fingerprint_Web_Server_(OTG-INFO-002)
- OWASP-02: https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html
- OWASP-03: https://www.owasp.org/index.php/Top_10-2017_A3-Sensitive_Data_Exposure
- OWASP-04: https://www.owasp.org/index.php/Testing_for_User_Enumeration_and_Guessable_User_Account_(OWASP-AT-002)
- OWASP-05: https://www.owasp.org/index.php/Top_10-2017_A2-Broken_Authentication
- PCI-01: https://blog.pcisecuritystandards.org/migrating-from-ssl-and-early-tls
- PORTSWIGGER-01: https://portswigger.net/kb/issues/01000300_strict-transport-security-not-enforced
- QUALYS-01: https://blog.qualys.com/ssllabs/2018/11/19/grade-change-for-tls-1-0-and-tls-1-1-protocols
- TENABLE-01: https://www.tenable.com/
- VDBR-01: https://enterprise.verizon.com/resources/reports/DBIR_2018_Report.pdf
- WIKIPEDIA-01: https://en.wikipedia.org/wiki/Privilege_escalation
- WIKIPEDIA-02: https://en.wikipedia.org/wiki/DMARC

Acronyms

- ATO: Account Take Over, https://en.wikipedia.org/wiki/Credit_card_fraud#Account_takeover
- CVSS: Common Vulnerability Scoring System, https://www.first.org/cvss/

- DB: DataBase, https://en.wikipedia.org/wiki/Database
- FOSS: Free and Open-Source Software, https://en.wikipedia.org/wiki/Free_and_open-source_software
- FTP: File Transfer Protocol, https://en.wikipedia.org/wiki/File_Transfer_Protocol
- HSTS: HTTP Strict Transport Security, https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security
- HTTP(s): HyperText Transfer Protocol (Secure), https://en.wikipedia.org/wiki/HTTPS
- IP: Internet Protocol (address), https://en.wikipedia.org/wiki/IP_address
- MFA: Multi-Factor Authentication, https://en.wikipedia.org/wiki/Multi-factor_authentication
- MiTM: Man-in-the-Middle, https://en.wikipedia.org/wiki/Man-in-the-middle_attack
- OWASP: Open Web Application Security Project, https://www.owasp.org/index.php/Main_Page
- PCI DSS: Payment Card Industry Data Security Standard, https://en.wikipedia.org/wiki/Payment_Card_Industry_Data_Security_Standard
- SOW: Statement of Work, https://en.wikipedia.org/wiki/Statement_of_work
- SQL: Structured Query Language, https://en.wikipedia.org/wiki/SQL
- SQLi: SQL Injection, https://en.wikipedia.org/wiki/SQL_injection
- SSL: Secure Sockets Layer, https://en.wikipedia.org/wiki/Transport_Layer_Security
- TCP: Transmission Control Protocol, https://en.wikipedia.org/wiki/Transmission_Control_Protocol
- TLS: Transport Layer Security, https://en.wikipedia.org/wiki/Transport_Layer_Security
- UDP: User Datagram Protocol, https://en.wikipedia.org/wiki/User_Datagram_Protocol
- URI: Uniform Resource Identifier, https://en.wikipedia.org/wiki/Uniform_Resource_Identifier
- URL: Uniform Resource Locator, https://en.wikipedia.org/wiki/URL
- WAF: Web Application Firewall, https://en.wikipedia.org/wiki/Web_application_firewall
- XSS: Cross-Site Scripting, https://en.wikipedia.org/wiki/Cross-site_scripting

Legalities

- All third-party links are for convenience. They are not controlled by PEN Consultants. We do not endorse or support the content on these links. There is no guarantee of accuracy, availability, or trustworthiness. View and use them at your own risk. With that said, we have made a good faith effort to ensure all links are reputable, trustworthy web resources to supplement this document.
- As with the external referenced resources, any 3rd party vendor or product mentioned is not an endorsement nor any indication of confidence in the risks of using such a product or service.
- These recommendations, based on our findings, will help reduce the likelihood of a future attack/breach, but they will NOT eliminate it.

Appendix #1 – Statement of Work ("SOW")

A copy of the SOW from the contract:

[SOW HERE]

Appendix #2 – Pretext

A copy of the HVAC Tech pretext provided prior to and in the contract:

Pretext: HVAC technician - with full supporting evidence (i.e. the IT closet is actually heating up and the system needs to be fixed ASAP)

NOTICE: Although one or more of the testers are certified and experienced HVAC technicians, this is strict impersonation of both the technician and of the "service" provided. Tester(s) will not be performing a legitimate service call and may not be able to offer any advice or opinions as to the state of your HVAC system(s).

Phases of the Attack

- Phase 1: Disable the air conditioner serving the IT closet the night before.
  - This ensures the IT closet really does warm up and makes the pretext exponentially more believable.
- Phase 2: Wait for the phone call to the HVAC service company and intercept using the method chosen (#6 of Client Deliverables).
  - This prevents the real HVAC company from dispatching their tech and ensures Consultant is able to send our tech(s).
- Phase 3: Impersonate a HVAC technician
  - Tester will claim to be a HVAC technician who is there to fix the faulty cooling system serving the IT closet and will show a text message from a (fake) dispatcher giving a service order for that location.
  - Regardless of the location of the air handler, the tester will insist on visiting the IT closet and spending time there.
- Phase 4: Tap into the corporate network and/or other actions
  - This will be attempted whether the tester is being escorted or not - ensures employee is both escorting AND monitoring activity closely.
  - Insert am electronic hardware device into an open network jack.
- Phase 5: EndEx
  - The tester will eventually fix the broken air conditioner that services the IT closet and then leave the building.
  - Client retrieves devices at a later time and returns them to Consultant.

Client Deliverables:

1) A "Physical Social Engineering Approval Letter" signed by a C-level executive (or equivalent) and two alternates - standard letter will be provided with the contract
2) Building floor plan showing the location of the IT closet
3) The name of the HVAC service company
4) Schedule a time with Consultant prior to testing so we can install, or verify the installation of existing, temperature sensors in the IT closet - allows Consultant to monitor for dangerously high temperatures and prevent equipment damage.
   - This doubles as an opportunity for Consultant to become more familiar with the building without having to spend additional recon time.
   - Alternatively, Client monitors temperature directly and gives Consultant status updates.
5) Provide the location of the condenser (outside) that serves the IT closet, or area(s) that are deemed to be in scope.

6) Provide a means to intercept the communication channel between the target and HVAC company - Consultant will need to intercept the service call so that the real company does not show up first.
   - Option A: Client's maintenance employee (assuming there is one) can intercept and forward the call to Consultant or just give Consultant the message.
   - Option B: Convince the HVAC company's dispatcher to intercept and forward the call - the level of effort to do this could be costly unless Client has a good relationship with the HVAC company.
7) Let Consultant's certified HVAC technician in during the evening hours to safely disable the air conditioning system serving the IT closet. This can be done in conjunction with Client Deliverable #4.

Appendix #3 – Surveillance Equipment Used

Three body-worn surveillance cameras recorded the entire time we were inside any [CLIENT NAME] location. A button camera, bag camera, and pen camera were used. All cameras had a resolution of 1920x1080 pixels and included audio recording.

In accordance with Federal and [CLIENT STATE] state laws (i.e. "one-party consent"), only the testers who wore the surveillance cameras may view the unedited recordings unless the client can demonstrate employee consent to video and/or audio recording (ex. a company policy). Additionally, some [CLIENT NAME] members were captured during the course of testing and would require a separate demonstration of consent before another party could be allowed to view the unedited recordings containing said members. For more information, see [CLIENT STATE] State Ann. 123.45 or seek legal council.

Button Camera (fake button circled in red)

Pen Camera (circled in red)

Bag camera (circled in red)

Appendix #4 – Actual Timeline

Note: The names of the employees have been intentionally obfuscated. Names and/or pictures are available upon request.

- 25 Sep @ 0735: Arrived at employee parking lot.
- 0741: Left truck and walked to building as an employee (Emp-01) was being dropped off.

- 0742: Met Emp-01 at the door and asked about service request. Showed Emp-01 service request on our phone. Emp-01 let us into the building, via the employee entrance.
- 0744: Emp-01 passed us off to Emp-02.
- 0745: Emp-02 passed us to Emp-03 who had brought the visitor log book and had us fill in our name, company name, and time entered. ID was not asked for, nor checked. After this, we were asked to wait for next employee who would take us down.
- 0748: Emp-04 met us in the waiting area, confirmed what we were there for, and escorted us down to server room.
- 0750: Entered the server room with Emp-04. We signed the server room's visitor log.
- 0751: Emp-04 showed us where things were.
- 0752: Emp-04 asked if we were given a visitor fob so we could go in and out, as needed. We told him no, and he left the server room to get the fob.
  - 0752 (and 39 seconds) the door closes behind Emp-004.
  - 0752 (and 41 seconds) Tester-02 runs over to one of the network switches he had scoped out a few minutes earlier.
  - 0752 (and 45 seconds) Tester-02 inserts hardware implant into switch.
  - 0752 (and 51 seconds) Tester-02 returns to the HVAC unit.

IMAGE HERE

IMAGE HERE

- 0754: Emp-04 returned to the server room.
- 0755: Emp-04 leaves the room again.
- 0756: Emp-04 returns to the room.
- 0757: Emp-04 leaves the room again.
- 0758: Emp-04 returns to the room.
- 0801: Emp-04 leaves the room again.
- 0806: Emp-04 returns to the room.
- 0807: Emp-04 leaves the room again.
  - We knew Emp-04 would be gone for several minutes, as Emp-04 was meeting and signing in another vendor. Had it been in scope, we likely would have had time to remove a number of data drives and place them in our tool bag.
- 0816: Emp-04 returns to the room.
- 0817: We sign-out on the server room visitor log sheet.
- 0818: We leave the room with Emp-04.
- 0819: We leave our badges with Emp-05, but are not escorted out, leaving us with the ability to roam freely in the employee area of the credit union.
- 0821: We exited the building.

Appendix #5 – Common Passwords

The following is a list of common passwords used during password spray attacks, based on breach research. This is a boilerplate list to give an example of the format of passwords used and may not be updated to today's year/season/month/etc. or to the locality of your users (ex. state name, sports team name, etc.). More on that topic can be found at:

- https://penconsultants.com/passwordPolicy
- https://penconsultants.com/forcedPasswordRotation

EXAMPLE: Winter2019, Fall2019, Winter2019!, Fall2019!, Winter19!, Fall19!, Winter19, Welcome1, Password1, Password123, Password123!, P@ssword1, P@ssw0rd1, Passw0rd, Welcome123, Welcome123!, January2020, December2019, November2019, October2019, January2020!, December2019!, November2019!, October2019!, January123, December123, November123, October123, January1, December1, November1, October1, January1234, December1234, November1234, October1234, January12345, December12345, November12345, October12345, January12, December12, November12, October12, Winter12, Winter1234, Fall1234, Password10, Password11, Password12, Password13, Password14, Password15, Password16, Password17, Password18, Password19, Password00, Password01, Password02, Password03, Password04, Password05, Password06, Password07, Password08, Password09, Pa$$w0rd, Password2019, Password2020, Password20, God12345, Abc12345, Master19, Dragon19, Monkey19, Shadow19, Qwerty19, Iloveyou19, Thankyou19, Welcome19, Baseball19, Football19, Letmein19, Mustang19, Access19, Superman19, Batman19, Qwertyuiop19, Jesus19, Ninja19, Master20, Dragon20, Monkey20, Shadow20, Qwerty20, Iloveyou20, Thankyou20, Welcome20, Baseball20, Football20, Letmein20, Mustang20, Access20, Superman20, Batman20, Qwertyuiop20, Jesus20, Ninja20, Master2019, Dragon2019, Monkey2019, Shadow2019, Qwerty2019, Iloveyou2019, Thankyou2019, Welcome2019, Baseball2019, Football2019, Letmein2019, Mustang2019, Access2019, Superman2019, Batman2019, Qwertyuiop2019, Jesus2019, Ninja2019, Master2020, Dragon2020, Monkey2020, Shadow2020, Qwerty2020, Iloveyou2020, Thankyou2020, Welcome2020, Baseball2020, Football2020, Letmein2020, Mustang2020, Access2020, Superman2020, Batman2020, Qwertyuiop2020, Jesus2020, Ninja2020, Master1, Dragon1, Monkey1, Shadow1, Qwerty1, Iloveyou1, Thankyou1, Welcome1, Baseball1, Football1, Letmein1, Mustang1, Access1, Superman1, Batman1, Qwertyuiop1, Jesus1, Ninja1, Master12, Dragon12, Monkey12, Shadow12, Qwerty12, Iloveyou12, Thankyou12, Welcome12, Baseball12, Football12, Letmein12, Mustang12, Access12, Superman12, Batman12, Qwertyuiop12, Jesus12, Ninja12, Texas2019, Texas2020, Texas2019!, Texas2020!

Document Properties and Version Control

Tester(s)	
Author(s)	
Reviewed by	
Approved by	

Version	Date	Name	Comments