

SecurityCompass



VotingWorks

ARLO DETAILED REPORT

Contents

- Document Information 3**
- Introduction 4**
 - SCOPE4
 - Targets and Environment..... 4
 - Out-of-Scope 4
 - ASSESSMENT LIMITATIONS5
- Testing Observations 6**
 - OVERVIEW6
 - AUTHENTICATION AND AUTHORIZATION7
 - SESSION MANAGEMENT8
 - DATA VALIDATION9
 - INFRASTRUCTURE TESTING12
 - SOURCE CODE REVIEW14
 - (MIS)CONFIGURATION REVIEW16
- Findings and Recommendations 16**
 - RISK CLASSIFICATION.....16
 - CURRENT VULNERABILITIES SUMMARY17
 - DETAILED VULNERABILITIES18
 - F-01 Improper Session Termination..... 18
 - F-02 Weak SSL/TLS Configurations..... 21
 - F-03 Audit Board Enumeration 23
 - F-04 Improper CSP Configuration 27
 - F-05 Lack of Audit Board Passphrase Collision Handling 29

Document Information

Document/Testing History

Date	Version	Description
November 13, 2020	1.0	Detailed Report

Introduction

VotingWorks engaged Security Compass to perform an open-box web application security assessment and a tool-assisted source code review. Testing was performed from October 26, 2020 to November 13, 2020. The goal of this engagement was to determine the overall security posture through the discovery of security vulnerabilities and a qualitative assessment of the associated risk levels. Remediation strategies are provided for each vulnerability to help VotingWorks mitigate or reduce identified risks.

SCOPE

TARGETS AND ENVIRONMENT

The scope of the assessment was limited to the environments and targets listed below:

Environment	Targets
Production	https://arlo.voting.works
Staging	https://vx-arlo-pentest.herokuapp.com/

OUT-OF-SCOPE

The following components and tests were **out-of-scope** for this review:

- ▶ Any applications and infrastructure external to the Arlo application. In cases where the Arlo application had inbound and/or outbound interfaces with another application, the interfaces and communications were considered in scope. All other external elements were excluded.
- ▶ Supporting policies, procedures, and processes
- ▶ Social engineering
- ▶ Software development life cycle
- ▶ Detailed hardware and non-Arlo software configuration
- ▶ Auth0 authentication mechanisms

ASSESSMENT LIMITATIONS

The ever-changing technology landscape and the increasing sophistication of attacks against networked systems are reasons for which no entity can truthfully claim to identify all security issues, nor guarantee the lifetime-security of an organization's network and applications. Note that this point-in-time assessment was based on a best-effort basis and was performed only on the environment provided by VotingWorks. Thus, changes to the environment may impact the applicability of results provided herein.

Security Compass cannot guarantee 100% coverage for any security assessment.

Testing Observations

The following section contains Security Compass's observations from the security assessment. These notes should be read as informational. Any vulnerabilities or significant risks that were discovered during the assessment are explained in more detail in the [Findings and Recommendations](#) section.

OVERVIEW

VotingWorks's Arlo application is a tool that is used to conduct post-election audits in the United States. The application has the following types of functionality:

- ▶ A page that allows Super Admins to create, list, and delete organizations and to log in as any Audit Administrator users who are registered to the application
- ▶ A summary page for Audit Administrators that lists all started, in-progress, and completed audits and provides options for new audits to be created
- ▶ An Audit Setup functionality that allows users to set the audit specifications and settings
- ▶ An Audit progress page that allows Audit Administrators to track the progress of active audits
- ▶ A Member Sign-In page for Audit Board users to input their personal information, such as party affiliation and full name
- ▶ A summary page for Audit Board users to review their audited ballots and batches so that they can start a new batch of ballots or submit the audit results to the jurisdiction
- ▶ Ballot Card Data Entry functionality that allows Audit Board users to audit each of their assigned ballots
- ▶ File-upload options that allow the following:
 - Super Administrators to upload a list of Jurisdictions Administrators
 - Jurisdiction Administrators to upload Ballot Manifest Files
- ▶ A File Download functionality that allows Jurisdiction Administrators to retrieve the following:
 - Aggregated Ballot Retrieval Lists
 - Placeholder Sheets
 - Ballot Labels
 - Audit Board Credentials

Users who interact with the Arlo application are external to the VotingWorks network. The Arlo application has both regular and administrative users who have access to the application.

AUTHENTICATION AND AUTHORIZATION

The application leverages authentication that is performed via Auth0's Universal Login login flow. Therefore, authentication (e.g. the login functionality, password resetting, security question handling) was deemed to be out of scope for this assessment. However, the assessment team did attempt to bypass this login flow (i.e. authenticate without the use of the out-of-scope functionality). No means to bypass authentication and no problems in the Universal Login flow were identified.

After authenticating with a username and a password, the user is redirected back to the application, where the callback endpoint verifies whether the authentication results that Auth0 returns are valid.

```
GET /auth/auditadmin/callback?code=Tl0ocImD90XCyX0_&state=
OdPoyJd288HXQxJPLix3Su6z1hhVN1 HTTP/1.1
Host: vx-arlo-pentest.herokuapp.com
```

Figure 1: The callback endpoint that verifies the authentication results that Auth0 returns

If the authentication is successful, a session cookie named "session" is assigned to the user. The cookie's value is a Base64-encoded signed token. The token's signature is properly verified by the server, and all attempts to alter the token's value failed. The cookie has both the HttpOnly and Secure flags set. Each request is validated to ensure that the user is authenticated before the request is processed by the server.

Name	Value	Domain	HttpOnly	Secure
session	eyJfYXV0aDBfc2Ff...	vx-arlo-pentest.herokuapp.com	true	true

Figure 2: The session token that is used by the application

The application was designed so that Audit Admins who belong to a given organization can view and modify only audits within that organization. Audit Admins are not supposed to be able to modify the audits of other organizations. During testing, this logic was consistent. The application displays an HTTP 403 Forbidden error to prevent users from accessing audits from other organizations.

```
errors:
  ▼ 0:
    errorType: "Forbidden"
    ▼ message: [REDACTED] securitycompass.com does not have access to organization e27da67d-4
               dd5d8d53cbad"
```

Figure 3: An error message that is returned in response to attempts to access other users' resources

The application has four user roles: Jurisdiction Admin, Audit Admin, Super Admin, and Audit Board. Multiple attempts were made to access the resources of other users of the same role and to access resources that are available only to a different role. These attempts failed, and the application returned a generic Forbidden error message.

Note that although the staging environment leveraged standard Auth0 users without multi-factor authentication (MFA) enabled, production users utilize Auth0 single sign-on (SSO) features via Google, where VotingWorks is actively using MFA controls.

SESSION MANAGEMENT

Standard session-management test cases were used to test the Arlo application. The assessment included attempts to identify the following issues:

- ▶ Session hijacking
- ▶ Session lockout
- ▶ Session puzzling
- ▶ Other session-related issues
- ▶ Cross-site request forgery

When a user logs in, the application assigns a new Base64-encoded cookie that is used as a session token. The generated session token is sufficiently random and cannot be predicted. The token is also properly signed, and its signature is verified by the server. However, the application does not invalidate the user's session when the user logs out, and the session is maintained for a long period. See F-01 for more information.

The authentication cookie is set with the "Secure" and "HttpOnly" flags by default. The Secure flag prevents the cookie from being transferred across unsecured channels where an attacker could

intercept the plaintext cookie and use it to hijack a user's session. The HttpOnly flag prevents JavaScript from retrieving the cookie's value.

DATA VALIDATION

Attempts were made to perform HTTP Header, URL, and POST-body tampering attacks on identified input parameters in the application. Attempts were also made to inject both raw and encoded cross-site scripting (XSS), SQL injection, XML external entity (XXE), poorly encoded, and poorly formed payloads throughout the service. These injections were made with the goal of tampering with the service's data handling and processing functions to trigger client-side or server-side code execution.

SQL-injection attempts were performed on various parameters in the application. No issues were identified, and all attacks were unsuccessful. The application handled these attempts by demonstrating no change in response.

The application handled the XSS payloads gracefully by encoding the output back to the user or performing proper input validation.

```
<li>
  <form method="post" action="./delete-election/871d0103-
    &#34;&gt;&lt;&script&gt;alert(1)&lt;/script&gt; [<a
    href="./jurisdictions?election_id=871d0103-d66d-4792-
  >jurisdictions</a
```

Figure 4: The output is encoded to prevent XSS attacks.

Payloads that were inserted at the database level were also successfully output encoded and did not trigger XSS.

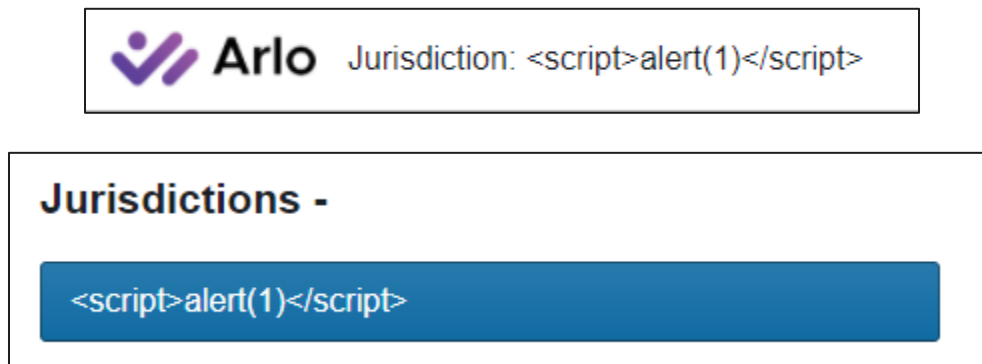


Figure 5: Payloads injected at the database level did not render or execute in the browser when viewed.

Parameters in several requests were tampered with to determine whether the application would display unauthorized content or an error message. The application handled these test cases gracefully, and no issues were identified.

The xkcdpass library is used to generate hard-to-guess but easy-to-read Audit Board passphrases in the event that an Audit Board member is unable to scan a QR code. Because these credentials are designed to be printed and scanned on-location, this generation method allows these members to easily input the values into their mobile or web browsers while maintaining a low level of credential predictability. The implementation that is in use by Arlo uses the default word list but decreases the length from six words to four.

```
118     audit_boards = [  
119         AuditBoard(  
120             id=str(uuid.uuid4()),  
121             name=json_audit_board["name"],  
122             jurisdiction_id=jurisdiction.id,  
123             round_id=round.id,  
124             passphrase=xp.generate_xkcdpassword(WORDS, numwords=4, delimiter="-"),  
125         )  
126     for json_audit_board in json_audit_boards  
127 ]
```

Figure 6: From the audit_boards.py source code file, the numwords parameter is overwritten to generate four-word phrases.

Passphrases were manually replaced at the database level to simulate a collision scenario. If a duplicate passphrase is created, the application will likely not fail gracefully. The public.audit_board table enforces a unique value constraint on the “passphrase” column, but no code-level exception handling is present to manage this scenario. More information can be found in F-02.

```
ERROR: duplicate key value violates unique constraint "audit_board_passphrase_key" DETAIL: Key (passphrase)=  
(enable-jester-grandkid-reclaim) already exists. .
```

Figure 7: If an Audit Board passphrase collision occurs, the key will not be written due to the constraints on the public.audit_board table.

Because multiple workflows accept comma-separated value (CSV) files as inputs, instances of exploitable injection scenarios were explored. Although a Jurisdiction Administrator can entice an Audit Administrator to download a malicious CSV (e.g. by uploading a malformed file, prompting the view seen in Figure 8), the built-in controls in modern versions of programs such as Microsoft Excel significantly decrease the likelihood of a user being unknowingly targeted by such an attack.

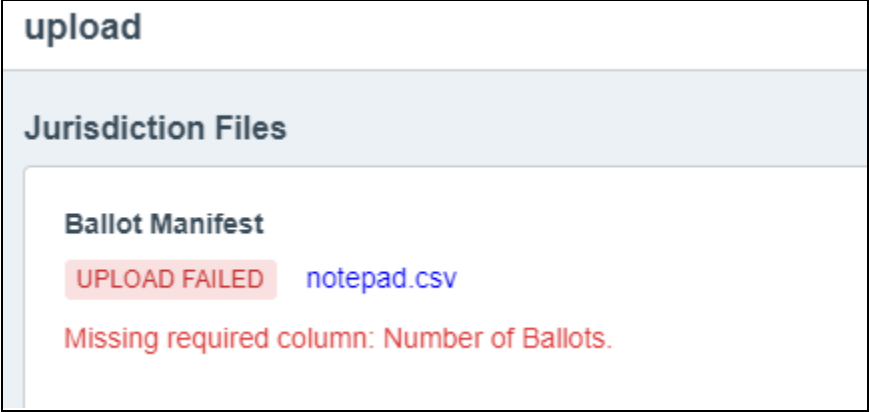


Figure 8: The “Ballot Manifest Upload Failed” view within the Audit Administrator’s audit progress page

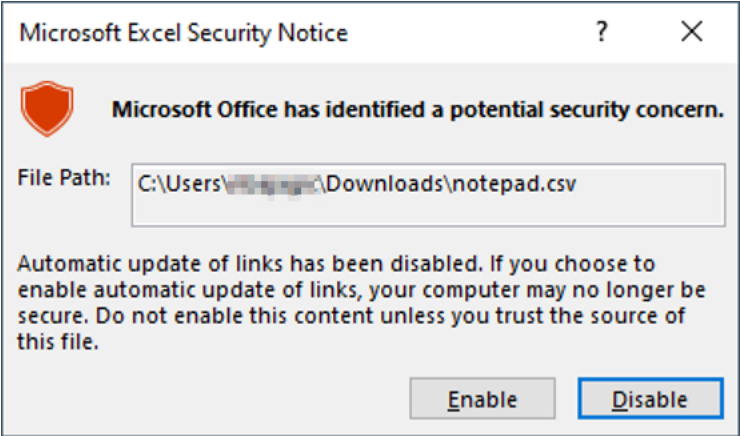


Figure 9: The Security Notice prompt that appears when the malicious file is opened.

Subsequent warnings inform users that the links to external data in the workbook require the user to enable specific settings. The warning is as follows: “This workbook contains links to external data sources that use DDE (Dynamic Data Exchange) that may be unsafe and have been disabled. See File > Options > Trust Center for DDE configuration options.”

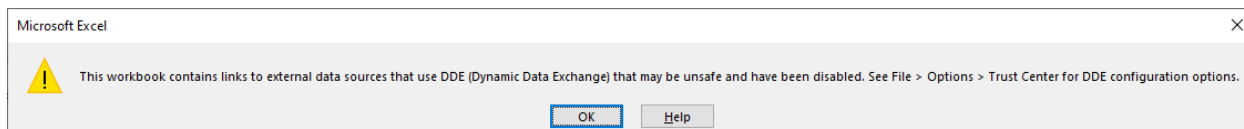


Figure 10: The warning that indicates that the settings for enabling links to external data sources via Dynamic Data Exchanges must be explicitly enabled by the user

INFRASTRUCTURE TESTING

Due to the targeted nature of this assessment, port scans were completed on all hosts even if they were unresponsive to the initial host-discovery scans. A variety of scanning techniques were used to find services listening on TCP and UDP ports. The scans included full connection scans, TCP SYN scans, UDP data packet scans, and TCP source port scans. In most cases, TCP SYN scans were run against the entire range of 65,535 TCP ports in an attempt to find any uncommon services, backdoors, or common services running on non-standard ports. No services beyond HTTPS (port 443) and HTTP (port 80), which redirected to HTTPS (port 443), were found.

The data that supports the Arlo application is stored in a PostgreSQL database that runs on an Amazon AWS EC2 host. The version of the database in use is PostgreSQL 12.4 for Ubuntu 16.04.16. It does not have any open, public vulnerabilities.

```
dd530ij5qorod2=> SELECT version();
version
-----
PostgreSQL 12.4 (Ubuntu 12.4-1.pgdg16.04+1) on x86_64-pc-linux-gnu, compiled
by gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609, 64-bit
```

Figure 11: The query for version information reveals an up-to-date database instance.

The default “postgres” named database is not available. Instead, the main database name is sufficiently random and hard to guess. The same attributes apply to the username and password that are required to connect to the database. Note that the “superadmin” user, “postgres”, was not provided for testing. However, the authentication methods that are configured for this user require a connection from a specific address and the provision of valid Kerberos materials. Therefore, brute-force attacks (including those performed with dictionary and word-list attacks) are not likely to be successful.

```

$ psql -h 50.16.219.58 -U postgres -d dd530ij5qorod2
Password for user postgres:
psql: error: could not connect to server: could not initiate GSSAPI security
context: Unspecified GSS failure.  Minor code may provide more information
could not initiate GSSAPI security context: Server
postgres/50.16.219.58@SECURITYCOMPASS.COM not found in Kerberos database
FATAL: password authentication failed for user "postgres"
FATAL: no pg_hba.conf entry for host "174.142.184.236", user "postgres",
database "dd530ij5qorod2", SSL off

```

Figure 12: Attempts to access the postgres user on the target database reveal the presence of additional authentication controls.

```

$ psql -h 50.16.219.58 -U u4llae62v5fn4r -d dd530ij5qorod2
Password for user u4llae62v5fn4r:
psql (12.4 (Debian 12.4-1))
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits:
256, compression: off)

```

Figure 13: Successful connections are encrypted via TLS 1.2 using ECDHE-RSA-AES256-GCM-SHA384 ciphers.

The main database user has permissions to read from and write data to the necessary tables in the “public” schema. However, no additional permissions or roles have been assigned that would allow for privilege escalation to perform sensitive tasks such as reading from or writing to system files or otherwise interacting with the underlying system. For example, neither the “pg_read_server_files” or the “pg_execute_server_program” roles permissions are assigned or can be self-assigned.

Login Role - u4llae62v5fn4r

General Definition Privileges **Membership** Parameters Security SQL

Roles

- × pg_read_all_settings
- × pg_read_all_stats
- × pg_signal_backend

Roles shown with a check mark have the WITH ADMIN OPTION set.

Figure 14: The primary database user’s permissions, as configured within the dd530ij5qorod2 database

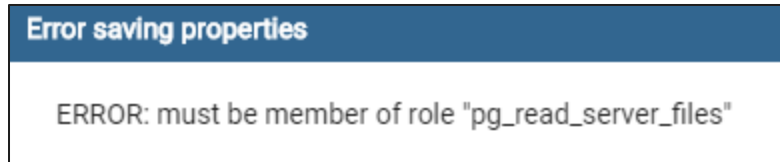


Figure 15: The primary database user is not permitted to read server files.

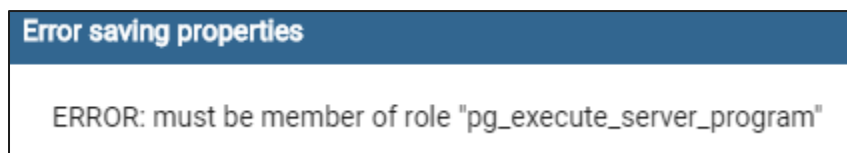


Figure 16: The primary database user is not permitted to execute server programs.

Although the permissions above (Figure 14) do not present an immediate risk of privilege escalation, they do allow for the successful retrieval of configuration data, `pg_stat_*` views, and the ability to signal other back ends (cancel the query or terminate the back-end process). These actions should be granted only to trusted users, which the primary user is designed to be.

The Arlo application is deployed within a default Heroku configuration, so the in-use services (such as EC2 instances, Elastic Block Storage, and load balancers) are using at-rest encryption options as offered by AWS. The PostgreSQL database, however, does not employ any encryption. Employing encryption is a best practice, but sufficient defense-in-depth controls are in place to reduce the likelihood of data leakage through external attacks.

SOURCE CODE REVIEW

The goal of this part of the assessment was to manually review critical code sections to determine the overall security posture of the application, verify the vulnerabilities that were reported by Checkmarx, and perform a qualitative assessment of the associated risk levels.

Checkmarx reported a total of 18 issues: three high-risk issues, 13 medium-risk issues, and two low-risk issues. These issues were reviewed, and the assessment team determined that they were all false positives or not applicable to the application.

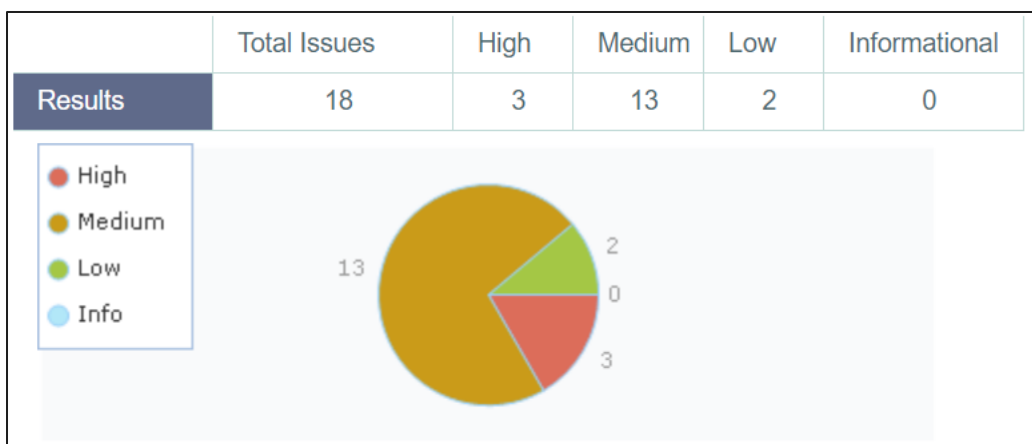


Figure 17: The issues reported by Checkmarx

The three high-risk issues are instances of a vulnerability called OS_Access_Violation. The tool's findings suggest that the file operation methods use user-provided input from three sources without proper validation. However, the assessment team determined that the scenarios in which these issues would be exploited are not present based on the available circumstances in which these methods are used.

```

d = input("Where would you like nOAuth installed? [default: .]")

if not d:
    d = "."

d = os.path.expanduser(d)

if not os.path.exists(d):
    os.makedirs(d)

os.chdir(d)

```

Figure 18: The "d" variable is used without proper validation.

The assessment team determined that the other issues either were false positives (e.g. missing HSTS header, missing clickjacking protection, using deprecated React function) or did not pose any risk to the application.

The missing security headers that were noted by Checkmarx (X-Frame-Options and HSTS) are configured on the web server, not on the application level. The deprecated React function, `ReactDOM.render()`, is deprecated if it is used to hydrate a server-rendered container, which is not the case for Arlo.

(MIS)CONFIGURATION REVIEW

The following libraries were determined to be the main components in use for the Arlo web application:

- ▶ React 16.9.0
- ▶ styled-components 4.3.2
- ▶ Flask 1.0.1
- ▶ Werkzeug 1.0.1
- ▶ Python 3.8.6

Each library is either the latest available version or otherwise a currently supported version. In each case, no public vulnerabilities have been reported.

Open-source scripts such as testssl.sh were used to review the production host, arlo.voting.works, for TLS configuration issues. The testssl.sh script checks for insecure versions of TLS and supported insecure cipher suites. The host is configured to accept encrypted connections using insecure TLS versions, 1.0 and 1.1. For more information, see F-02.

The application was also reviewed for missing or misconfigured security headers. These security headers provide an additional layer of defense-in-depth and would increase the security posture of the application.

Generally, the application is using appropriate security headers: Content Security Policy (CSP), HTTP Strict Transport Security (HSTS), and X-Frame-Options. However, the CSP header is not properly configured to prevent or mitigate DOM-based injection attacks. See F-03 for more information.

Findings and Recommendations

The remainder of this report describes the vulnerabilities that Security Compass identified as part of the assessment, their impact, and recommendations for resolving the vulnerabilities.

RISK CLASSIFICATION

The risk ratings below are established using the CVSS (Common Vulnerability Scoring System) but are subject to changes as deemed appropriate by VotingWorks and Security Compass.

CURRENT VULNERABILITIES SUMMARY

ID	Vulnerability	Risk
F-01	Improper Session Termination	Low
F-02	Weak SSL/TLS Configurations	Low
F-03	Audit Board Enumeration	Low
F-04	Lack of Audit Board Passphrase Collision Handling	Info
F-05	Improper CSP Configuration	Info

DETAILED VULNERABILITIES

F-01 Improper Session Termination

LOW RISK

DESCRIPTION

The application's session token is not invalidated on the server when the user clicks the "Log out" button. Instead, the server merely assigns a new cookie with the "_user" value set to null. Therefore, after a user has clicked the logout button, the session cookie that the user was using remains valid and may be used to access the application normally.

Additionally, the session token does not expire, so the window in which an attacker can compromise a valid session is increased, and the likelihood of an attacker who has compromised a session retaining their access is also increased.

Consequently, an attacker who steals a legitimate user's session token (e.g. as part of a cross-site scripting attack) may retain indefinite access to the application while posing as the victim, even if the user logs out of the application.

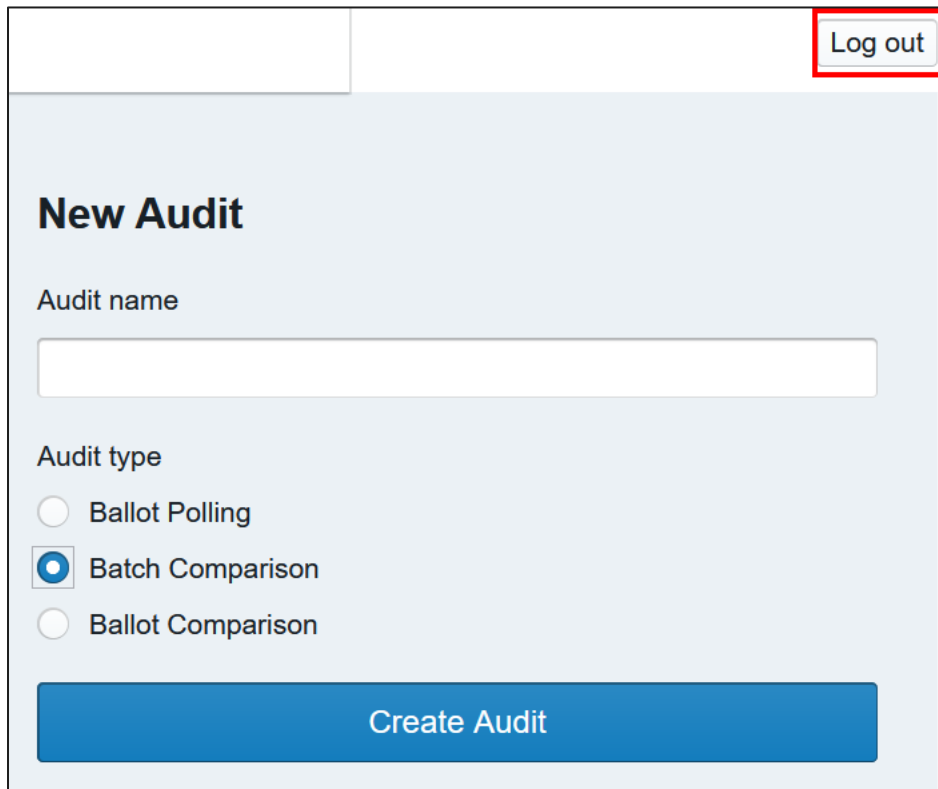
SUPPORTING EVIDENCE

The use of an HTTP proxy such as Burp Suite is required to reproduce the steps described below. Please see the following: http://portswigger.net/burp/help/suite_gettingstarted.html

The following steps can be used to illustrate the vulnerability:

1. While capturing traffic with Burp Suite, log in to the Arlo application as an Audit Admin

2. Click the “Log out” button



The screenshot shows a web interface for creating a new audit. At the top right, there is a "Log out" button highlighted with a red rectangle. Below it, the heading "New Audit" is displayed. Underneath, there is a label "Audit name" followed by a text input field. Below the input field is the label "Audit type" with three radio button options: "Ballot Polling", "Batch Comparison" (which is selected), and "Ballot Comparison". At the bottom of the form is a blue button labeled "Create Audit".

Figure 19: The “Log out” button

3. Observe that the user has been logged out of the application
4. In the HTTP proxy, find the request that was sent to the “/api/me” endpoint while the user was authenticated. Replay this request.
5. In the response, observe that JSON that contains information about the logged-in user and the available audits is returned, even though the user has logged out of the application

IMPACT

- ▶ An attacker who successfully re-uses the session token of a previously authenticated user would gain complete access to that user’s account, with all the same privileges. Doing so would have the same results as a session-hijacking attack.

LIKELIHOOD

- ▶ The likelihood of a successful session hijacking is quite low. To obtain a user’s session cookies, an attacker would have to capture the user’s requests to Arlo while the user is authenticated. Alternatively, an attacker could perform an attack such as cross-site scripting to obtain valid session tokens for the user.

RECOMMENDATIONS

- ▶ To mitigate the risk that this vulnerability poses, ensure that the application invalidates the user's sessions and clears their associated tokens from the server-side cache when a user clicks the logout button.

F-02 Weak SSL/TLS Configurations

LOW RISK

DESCRIPTION

The Arlo host supports insecure cipher suites and protocols. The presence of these SSL-based weaknesses may lead to sensitive information disclosure because they either increase the likelihood that an attacker will be able to decrypt traffic between two hosts or facilitate man-in-the-middle attacks that would allow an attacker to impersonate the target host to a user. To exploit these vulnerabilities, an attacker would need to be on the same physical or logical network as either a user or the target host to be able to intercept or capture encrypted communications and reach the target host.

TLS Protocol Versions 1.0 and 1.1

The host is configured to accept encrypted connections using TLS versions prior to TLSv1.2. The TLS versions prior to TLSv1.2 are vulnerable to various attacks due to known weaknesses in protocol design. An attacker could abuse these protocols to execute an on-path attack and steal sensitive information. The Browser Exploit Against the SSL/TLS (BEAST) vulnerability exists in TLS version 1.0 when a block cipher is in use, which can allow an attacker to obtain plaintext information.

SUPPORTING EVIDENCE

The steps below require the use of the testssl.sh utility. Testssl.sh can be obtained at <http://testssl.sh/>

The following steps can be used to illustrate this vulnerability on a Linux or Windows machine:

1. Open the terminal prompt and enter the following command:
 - `./testssl.sh ip_address:port`

2. Observe that a list of all the supported ciphers and the server certificate information are displayed

```
Start 2020-11-12 21:05:02 -->> 104.22.53.163:443 (arlo.voting.works) <<--
Further IP addresses: 104.22.52.163 172.67.7.90 2606:4700:10::ac43:75a 2606:4700:10::6816:35a3
                    2606:4700:10::6816:34a3
rDNS (104.22.53.163): --
Service detected:    HTTP

Testing protocols via sockets except NPN+ALPN

SSLv2      not offered (OK)
SSLv3      not offered (OK)
TLS 1      offered (deprecated)
TLS 1.1    offered (deprecated)
TLS 1.2    offered (OK)
TLS 1.3    offered (OK): final
NPN/SPDY   h2, http/1.1 (advertised)
ALPN/HTTP2 h2, http/1.1 (offered)
```

Figure 20: TLS versions 1.0 and 1.1 are offered by the Arlo host.

IMPACT

A successful attack could allow an attacker to decrypt part or all of a sensitive piece of information such as a session cookie or specific target text. The attacker could then use stolen session cookies or sensitive information to access the application while impersonating a valid user or as targeting information for further attacks.

LIKELIHOOD

The likelihood of the vulnerabilities being exploited is low due to the number of resources and repeated requests that would be required. Furthermore, the attacks would require that the attacker be able to intercept encrypted traffic between the target server and a legitimate user.

RECOMMENDATIONS

Reconfigure the affected hosts to allow only strong TLS protocol versions (i.e. TLS 1.2 and above). See the following for more information about recommended configurations:

https://wiki.mozilla.org/Security/Server_Side_TLS#Modern_compatibility

F-03 Audit Board Enumeration

LOW RISK

DESCRIPTION

The Arlo application manages Audit Board member access via pseudo-randomly generated, multi-word passphrases with a hard-coded delimiter. A resulting passphrase is appended to the /auditboard/ URL path as a unique route to a specific audit. Generation is performed using XKCD's xkcdpass Python module, which ensures that Audit Board members can easily read and understand the URL that is associated with the Audit Board that they belong to. The readability of the URL is important in situations where an Audit Board member is unable to scan the QR code (which is generated to visually represent the URL) and must instead manually type the URL into a web browser's address bar. As the sole means of authentication to an Audit Board, Arlo's implementation of the xkcdpass module may present an increased risk of unauthorized access.

SUPPORTING EVIDENCE

The steps below require the use of a Python module called xkcdpass, which can be retrieved from <https://pypi.org/project/xkcdpass/>

The following steps can be used to illustrate the vulnerability:

1. Navigate to the following source code file in the Arlo GitHub repository and observe the mechanism that is used to generate audit board passphrases:
arlo/server/api/audit_boards.py

```
118     audit_boards = [  
119         AuditBoard(  
120             id=str(uuid.uuid4()),  
121             name=json_audit_board["name"],  
122             jurisdiction_id=jurisdiction.id,  
123             round_id=round.id,  
124             passphrase=xp.generate_xkcdpassword(WORDS, numwords=4, delimiter="-"),  
125         )
```

Figure 21: audit_boards.py generates passphrases using the xkcd passphrase library.

2. Navigate to the following source code file and observe the way in which the passphrases generated by `audit_boards.py` are used to redirect Audit Board members to the appropriate audit context:
`arlo/server/auth/routes.py`

```
197 @auth.route("/auditboard/<passphrase>", methods=["GET"])
198 def auditboard_passphrase(passphrase):
199     auditboard = AuditBoard.query.filter_by(passphrase=passphrase).one()
200     set_loggedin_user(UserType.AUDIT_BOARD, auditboard.id)
201     return redirect(
202         f"/election/{auditboard.jurisdiction.election.id}/audit-board/{auditboard.id}"
203     )
```

Figure 22: `route.py` defines Audit Board routes within the Arlo web application using passphrases that are generated by `audit_boards.py`.

3. Using the `xkcdpass` module from the command line, generate a passphrase that is consistent with the settings outlined in `audit_boards.py`

```
root@SCLT00145:/mnt/d/VotingWorks$ xkcdpass -n 4 -d'-'
playset-baffle-catching-triangle
```

Figure 23: A four-word passphrase is generated using the `xkcdpass` Python module.

4. Attempt to access an audit board using the generated phrase



The screenshot shows a web browser address bar with the URL `arlo.voting.works/auditboard/playset-baffle-catching-triangle`. Below the address bar, a white error message box is displayed with the following JSON content: `{"errors": [{"errorType": "Internal Server Error", "message": "No row was found for one()"}]}`

Figure 24: An Internal Server Error message indicates that no corresponding audit board was found for the given passphrase.

- Using a passphrase that is known to correspond to an active audit, repeat step 4. Note that during testing, a non-production host was used for this step to minimize the disturbance of production assets.

The screenshot shows the Arlo Audit Board interface. At the top left is the Arlo logo, and at the top right is a 'Log out' button. The main heading is 'Audit Board #2: Ballot Cards to Audit'. Below the heading is a paragraph of instructions: 'The following ballots have been assigned to your audit board for this round of the audit. Once these ballots have been located and retrieved from storage, click "Start Auditing" to begin recording the votes you see marked on the paper ballots. When you are finished auditing these ballots, click "Auditing Complete - Submit Results" to submit the results. **Note that you will not be able to make changes after results are submitted.**' Below the text are two buttons: 'Start Auditing' and 'Auditing Complete - Submit Results'. At the bottom is a table with the following data:

Container	Tabulator	Batch	Ballot Position	Status
Box 1 - Tabulator 101 - Prec	2	N/A	Not Audited	N/A
Box 1 - Tabulator 101 - Prec	4	N/A	Not Audited	N/A

Figure 25: A successful request may present an attacker with an in-progress audit.

IMPACT

A successful attack may allow an attacker to gain access to and interfere with in-progress audit information, including submitting fraudulent ballot results. These actions may erode client trust in the Arlo application.

LIKELIHOOD

The likelihood of a successful attack depends on the following:

- ▶ The default number of words used by the xkcdpass generator is six, but Arlo overrides this configuration to use four words. This number materially reduces the entropy of the resulting passphrase, which increases the margin of success for an attacker. Use of the default word list (eff-long) combined with the known delimiter format may further increase success rates.
- ▶ The availability of in-progress audits to which an attacker may gain access. If no audits are pending completion, the potential for malicious interaction is reduced. Because the lifecycle of an audit is typically less than a week, an attacker would have a limited window in which to guess the valid passphrase.
- ▶ Additional controls such as web application firewalls (WAFs). Arlo is configured to restrict access to the audit board routes based on the requester's country of origin via their IP address. Only US-based IP addresses are permitted. Additionally, Cloudflare is configured to log data related to and alert based on abnormal events such as high-frequency requests that are consistent with automated enumeration attacks and attempts to access non-existent routes.

RECOMMENDATIONS

Reconfigure the passphrase-generation logic to produce passphrases that contains a greater number of words. Doing so will further increase the entropy of the resulting passphrase, which increases the possible enumerable key space and reduces the chance of an attacker guessing the passphrase of a valid, in-progress audit.

Informational Findings

During the assessment, Security Compass discovered issues or errors that currently do not pose a risk to the organization or its data. However, implementing the following recommendations as best practices would provide defense-in-depth security and thereby improve the overall security posture of the application.

F-04 Improper CSP Configuration

INFORMATIONAL FINDING

DESCRIPTION

The Arlo application has a content security policy (CSP) that is not properly configured to prevent or mitigate DOM-based injection attacks. The CSP is configured with the “frame ancestors” directive, but it does not have the “script-src” directive to define which domains can load JavaScript.

SUPPORTING EVIDENCE

The following steps can be used to illustrate the vulnerability:

1. Navigate to Google’s CSP evaluator at <https://csp-evaluator.withgoogle.com/>
2. Input the application’s URL
3. Review the output. Notice the severe errors for “script-src” and “object-src”.

✓	default-src		▼
!	script-src		▲
?	'self'	'self' can be problematic if you host JSONP, Angular or user uploaded files.	
!	'unsafe-inline'	'unsafe-inline' allows the execution of unsafe in-page scripts and event handlers.	
✓	style-src		▼
?	object-src [missing]	Can you restrict object-src to 'none'?	▼
i	require-trusted-types-for [missing]	Consider requiring Trusted Types for scripts to lock down DOM XSS injection sinks. You can do this by adding "require-trusted-types-for 'script'" to your policy.	▼

Figure 26: Missing CSP headers

RECOMMENDATIONS

Define a content security policy that granularly specifies where the application expects to get data or resources from and what is allowed to execute within the application. Review the recommendations of the CSP evaluator website. Specifically, review and implement the script-src directive.

See the following for more information about the definition and use of a CSP:

- ▶ <https://developer.mozilla.org/en-US/docs/Web/Security/CSP>
- ▶ https://www.owasp.org/index.php/Content_Security_Policy

F-05 Lack of Audit Board Passphrase Collision Handling

INFORMATIONAL FINDING

DESCRIPTION

Audit Board members access their audits through a randomly generated, four-word, dash-delimited passphrase that is generated using XKCD's `xkcdpass` library. This functionality is defined in the `audit_boards.py` source code file. The "create_audit_boards" function, from which this library is called, does not include exception handling logic to detect and prevent attempts to write a duplicate passphrase to the underlying PostgreSQL database. Because the target column in which the passphrases are stored is configured to accept only unique values, a collision may result in an unhandled exception at the application level.

```
114 def create_audit_boards(election: Election, jurisdiction: Jurisdiction, round: Round):
115     json_audit_boards = request.get_json()
116     validate_audit_boards(json_audit_boards, election, jurisdiction, round)
117
118     audit_boards = [
119         AuditBoard(
120             id=str(uuid.uuid4()),
121             name=json_audit_board["name"],
122             jurisdiction_id=jurisdiction.id,
123             round_id=round.id,
124             passphrase=xp.generate_xkcdpassword(WORDS, numwords=4, delimiter="-"),
125         )
126         for json_audit_board in json_audit_boards
127     ]
128     db_session.add_all(audit_boards)
129
130     if election.audit_type == AuditType.BATCH_COMPARISON:
131         assign_sampled_batches(jurisdiction, round, audit_boards)
132     else:
133         assign_sampled_ballots(jurisdiction, round, audit_boards)
134
135     db_session.commit()
136
137     return jsonify(status="ok")
```

Figure 27: Audit Board is committed without error-handling logic to mitigate write attempts for existing passphrase values.

RECOMMENDATIONS

Ensure that appropriate exception handling is included when data is committed to the connected database.