AEGISBYTE | Headwind Remote

MOBILE APPLICATION SECURITY ASSESSMENT REPORT

REPORT DATA

www.aegisbyte.com
(866) HACKER-5
(571) 368-3950
(571) 368-3937

AEGISBYTE COMPANY DETAILS

Account Executive

Ashley Alcala
Account Executive
ashley.alcala@aegisbyte.com

Assessment Team

Felix Alcala
Lead Engineer
felix.alcala@aegisbyte.com

Jason Brewer
Lead Engineer
jason.brewer@aegisbyte.com

Technical Oversight

Jason Bernier
Lead Engineer
jason.bernier@aegisbyte.com

CLIENT DETAILS

Contact Information

Vsevolod Mayorov

Headwind Remote
Product Owner
mayorow@gmail.com

ASSESSMENT SCOPE SUMMARY

Engagement Timeframe

01 July 2023 - 10 July 2023

Engagement Scope

Headwind Remote Version 1.30 (Android)
Headwind Remote Version 1.35 (Android)

Project ID: Headwind Remote Mobile (Android) Applications

Report Date: July 12th, 2023

ENGAGEMENT OVERVIEW

Aegisbyte is a leading provider of specialized services in the field of mobile application penetration testing. Our expertise lies in employing advanced techniques and practices to thoroughly evaluate the security posture of mobile applications, with a focus on identifying areas that require improvement. At Aegisbyte, we go beyond conventional automated tests and prioritize manual assessments to uncover genuine attack vectors that could potentially exploit vulnerabilities in your application.

Backed by a team of seasoned professionals with extensive industry experience, Aegisbyte is at the forefront of the application security and penetration testing landscape. Our experts are subject matter authorities in their respective fields, ensuring that every resource deployed for your project possesses unparalleled expertise. With decades of combined knowledge and a commitment to excellence, Aegisbyte offers comprehensive and reliable solutions to safeguard your mobile applications from potential threats.

SERVICE DESCRIPTION

Mobile Application Security Testing is a crucial process that involves simulating real-world attacks using techniques similar to those employed by attackers. It aims to identify vulnerabilities and weaknesses in mobile applications, ensuring their resilience against potential threats. When seeking a comprehensive security assessment that surpasses the capabilities of a basic vulnerability scanner, it is essential to engage industry experts with extensive knowledge and experience in the field.

Extensive Application Assessment

With the increasing complexity and demand for mobile applications, they have become a primary target for attackers with malicious intent. Aegisbyte's Application Service offerings are designed to safeguard the mobile services of your enterprise applications against a wide range of security threats.

Application security issues not only represent the most prevalent type of vulnerability but also continue to evolve in terms of complexity. While the OWASP Top 10 is commonly used as a standard to identify application security flaws, it serves as merely a starting point. Numerous advanced vulnerabilities exist beyond the scope of this list. Automated vulnerability scanners and penetration testers that solely focus on the OWASP Top 10 may lag behind emerging threats, thereby leaving the application exposed to unknown risks.

At Aegisbyte, we surpass the limitations of the OWASP Top 10 by continuously pushing the boundaries of application security. Our approach entails exploring novel attack vectors and uncovering how unique architectures can be exploited, in addition to providing comprehensive solutions for addressing these vulnerabilities.

Manage Application Security Risk

In today's landscape, mobile applications have transcended their initial purpose of marketing tools. They now serve as interfaces to comprehensive technological ecosystems, incorporating various components such as critical databases, susceptible libraries and plugins, XML and LDAP capabilities, underlying operating systems, and client web browsers. As a result, the notion of a "mere mobile app" has become obsolete, posing significant challenges in terms of security protection.

CAMPAIGN OBJECTIVES

Vulnerability Identification

Aegisbyte consultants leverage the outcomes of the automated scan, combined with their extensive expertise and experience, to perform a meticulous manual security analysis of the client's applications. Our assessors systematically work to exploit vulnerabilities and acquire unauthorized remote access to the client's data and systems. The comprehensive results obtained from both the vulnerability scan and the manual testing are meticulously documented and presented in this report.

PROCESS AND METHODOLOGY

Aegisbyte employed a comprehensive methodology to conduct a security review of the mobile application(s). The process commenced with an extensive scanning and research phase, focusing on the architecture and environment of the application. Automated testing was performed to identify known vulnerabilities within the application. Subsequently, manual exploitation of vulnerabilities was carried out to uncover any potential security weaknesses present in the application.

1

Intelligence Gathering and Application Mapping

Intelligence Gathering: The environmental and architectural context of the app was analyzed to gain a general understanding of its context. Application Mapping: Based on the information obtained from previous phases, the app was mapped. This involved a combination of automated scanning and manual exploration. The mapping process provided a comprehensive understanding of the app, including its entry points, stored data, and main potential vulnerabilities. These vulnerabilities were then assessed and ranked according to the potential damage they could cause when exploited. This allowed the security tester to prioritize them accordingly. Additionally, this phase involved creating test cases that could be used during the test execution.

2

Automated and Manual Testing

Static Analysis: Reviewed the source code of the mobile app to ensure the proper implementation of security controls. Utilized a hybrid automatic/manual approach, where automated scans were employed to identify easily detectable vulnerabilities, while the assessor explored the code base with specific usage contexts in mind.

Manual Code Review: Performed a thorough manual analysis of the mobile app's source code to identify security vulnerabilities.

1. Employed various methods, ranging from basic keyword searches using the 'grep' command to a line-by-line examination of the source code.
2. Leveraged Integrated Development Environments (IDEs) with built-in code review functions, extended with additional tools.
3. Adopted a common approach to manual code analysis, which involved identifying key security vulnerability indicators by searching for specific APIs and keywords, such as database-related method calls like "executeStatement" or "executeQuery" code containing these strings served as a starting point for the manual analysis process.

3

Dynamic Analysis

Conducted dynamic analysis to test and evaluate apps during their real-time execution.

1. Identified security vulnerabilities and potential flaws in the program while it was running.
2. Performed dynamic analysis at both the mobile platform layer and against backend services and APIs.
3. Analyzed request and response patterns of the mobile app to assess security.
4. Checked for security mechanisms to ensure sufficient protection against common types of attacks, including data disclosure in transit, authentication and authorization issues, and server configuration errors.

4

Assessment Reporting

Once the engagement is complete, Aegisbyte delivers a detailed analysis and threat report which includes remediation steps. Our consultants set an industry standard for clear and concise reports, prioritizing the highest risk vulnerabilities first. The assessment includes the following:

- **Executive Summary**
- **Strategic Strengths and Weaknesses**
- **Identified Vulnerabilities and Risk Ratings**
- **Detailed Risk Remediation Steps**
- **Assets and Data Compromised During Assessment**

5

Remediation

As an optional addition to the standard assessment, Aegisbyte offers remediation retesting for all vulnerabilities listed in the report. At the conclusion of the remediation testing and request of the client, Aegisbyte will update the report with a new risk level determination and mark which vulnerabilities in the report were, in fact, remediated to include the new risk level.

SCOPING AND RULES OF ENGAGEMENT

While real attackers have no limits on Mobile Application Penetration engagements, we do not engage in penetration testing activities that violate any ethics or privacy concerns.

Constraints

No additional limitations were placed upon this engagement, as agreed upon with Headwind Remote.

Assessment Scope

1. Review the source code in the repository located at https://github.com/h-mdm/remote-control-android (a similar branch to the one specified).
2. Identify any instances of malware code, specifically focusing on:
3. Phishing code that masquerades as a trusted source, soliciting user authentication credentials or billing information, and subsequently transmitting the data to a third party.
4. Review and analyze code that intercepts the transmission of user credentials during transit.
5. Decompilation of the provided APK file, preferably using JADX, Apktool, or another reliable decompilation tool.
6. Conduct a thorough analysis of the decompiled code to detect any potential malware code.
7. Perform a clean build of the APK from the provided source code, ensuring that all third-party libraries originate from trusted sources.
8. Compare the decompiled code of the clean APK with the analyzed APK to identify the presence of any malware code.

The objective of a mobile app penetration test is to evaluate the security of an application and its surrounding infrastructure. It looks to uncover potential weaknesses that can be exploited by malicious actors and identify any logic flaws or vulnerabilities in the code. The tester will attempt to exploit these flaws to gain access to sensitive data and/or privileged functions, so as to gain an understanding of the application's security posture. The tester will also assess other aspects, such as user interface (UI) design and usability, as well as its compliance with industry standards.

Aegisbyte shall provide all personnel and items necessary to perform the functional and technical support described in this SOW, except those items specified as Customer (Headwind Remote) furnished equipment/property. Aegisbyte shall perform all tasks identified in this SOW.

Mobile Assessment Scope

1. To evaluate the security posture of a mobile application against established industry standards and published best practices;
2. To identify security weaknesses and document all relevant findings reported during the testing process;
3. To develop recommendations for any identified vulnerabilities for remediation;
4. To verify that the security controls, features, and architectures are in place to protect user data;
5. To ensure that all aspects of the mobile application (source code, platform, libraries) are secure.

Headwind Remote Mobile Client

Application(s)

Android - Headwind Remote Version 1.30
https://headwind-remote.com/files/hremote-1.30-donate.aab
SHA1: 3950c28d1af47bf6809e2a52c995f43cf7c8cb26

https://headwind-remote.com/files/hremote-1.30-donate.apk
SHA1: f528fe11b7dae67e0a52e5cc256a5646137cb840

Android - Headwind Remote Version 1.35
https://headwind-remote.com/files/hremote-1.35-premium.apk
SHA1: e71af46a7a0a08d8af5a111f4b530b6891c50522

Description

The Headwind Remote system is delivered as a container of Docker Compose. This simplifies the installation and makes Headwind Remote server compatible with every Linux distribution.

1. Screen sharing and delivery of gestures to devices is powered by the Janus media server.
2. Web service is based on Nginx. The HTTPS certificate for the domain name where the Headwind Remote application is installed is provided by LetsEncrypt.
3. The administrator's application for the remote control of Android devices is a web application. It can be opened in any web browser supporting WebRTC, for example, Chrome. So the web portion is cross-platform.
4. The mobile agent could be installed on an Android device. Mobile platforms other than Android are not supported by Headwind Remote.

The screencast is being broadcasted as an RTP stream based on the UDP bearer protocol. The video streaming data are sent by a mobile agent to a Janus inbound socket. Janus transforms this data to a secure WebRTC stream and sends it to the administrator's web application.

The mobile agent casts the Android device screen by using the MediaProjection API. This API is a part of Android (5.0 and above) and doesn't require any firmware customization or rooting. The video is encoded by low-level built-in encoders with low CPU costs.

By using this method to mirror the screen of an Android phone, the assessor was able to achieve a secure and low-latency screencast even in slow mobile networks (3G and above).

Headwind Remote utilizes solely the native Android API for screen sharing and gesture emulation on mobile devices.

Note: The mobile agent is compatible with Android 7 and higher on any device.

Headwind Remote can be used for remote control of the following:

- Smartphones and tablets
- Embedded Android devices
- POS kiosks and terminals
- Smart TVs and TV boxes
- Handheld terminals and barcode scanners
- Any devices running a custom AOSP build

Note: The agent functions as a standard Android application and can be downloaded from Google Play.

The server component is provided as a Docker-Compose container and is compatible with any Linux distribution. The installer is specifically optimized for Ubuntu Linux (16.04 or later, with 20.04 being recommended), and CentOS 8 is also supported. For other Linux distributions, modifications to the installer configuration are required (please contact us for further details).

Prerequisites

To install the Headwind Remote server, a domain must be set up (a subdomain is sufficient).

The server requires root access, and it is advisable to use Ubuntu Linux 20.04 LTS.

The following TCP/IP ports must be enabled on the firewall or forwarded if using NAT:

80/TCP: Used exclusively by certbot for certificate renewal.

443/TCP: Used by nginx to display the web user interface.

8989/TCP: WSS protocol utilized by WebRTC for video playback.

8089/TCP: Used by the web application to communicate with the Janus server through the REST API.

10000-10500/UDP: UDP is employed for the RTP screencast, with the port dynamically selected by Janus.

Sensitive Assets and Processes

User information, access to the users (end-user's phone).

EXECUTIVE SUMMARY OF FINDINGS

Aegisbyte, a renowned cybersecurity firm, undertook a comprehensive Mobile Application penetration test on behalf of Headwind Remote. The primary objective of this test was to evaluate the security posture of the Headwind Remote Android application(s) and offer proactive assistance in identifying vulnerabilities, assessing their severity, and suggesting appropriate remediation measures.

Aegisbyte meticulously analyzed the security aspects of the Headwind Remote Android application(s) and concluded that there exists a **moderate** level of risk concerning potential compromise by malicious attackers. This determination was based on the identification of specific vulnerabilities, which are extensively documented in the subsequent sections of this report.

To ensure a robust security posture, Aegisbyte presents a comprehensive breakdown of the findings from the assessment. Additionally, the report provides detailed recommendations for remediation, offering actionable steps to address the identified vulnerabilities. These valuable insights can be found in the later sections of this report.

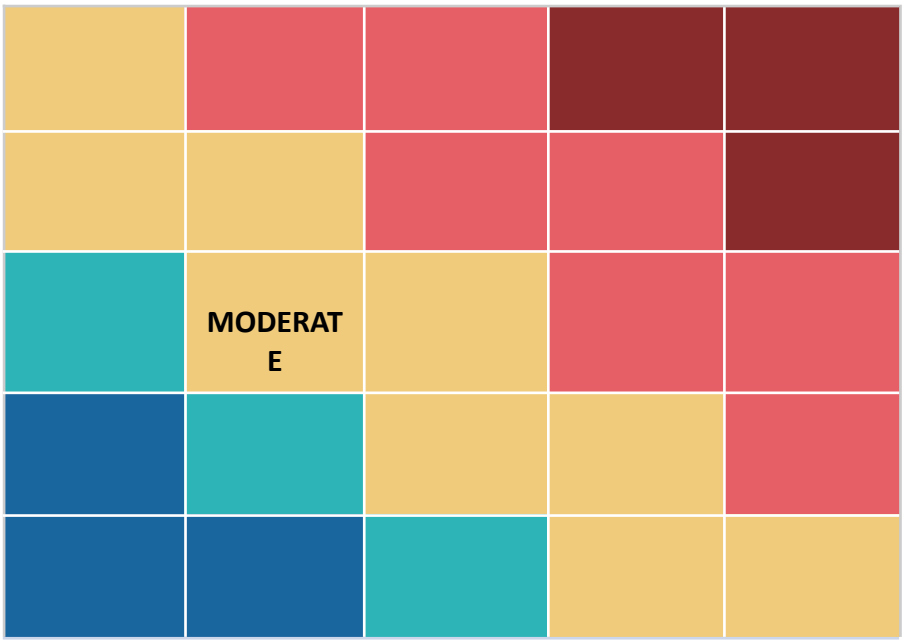
MOBILE APPLICATION RISK RATING

Aegisbyte employs a robust methodology for assessing the risk associated with mobile applications. Our risk assessment process centers around two key factors: Exploitation Likelihood and Potential Impact.

Exploitation likelihood refers to the ease with which vulnerabilities in a mobile application can be exploited by malicious actors. Through meticulous analysis and testing, Aegisbyte evaluates the application's code, architecture, and implementation to determine the likelihood of successful exploitation.

Potential Impact, on the other hand, gauges the potential business impact that could arise from the exploitation of identified vulnerabilities within the mobile application. Aegisbyte conducts a thorough examination of the application's functionalities, data handling mechanisms, and integration points to ascertain the magnitude of the potential damage to the overall environment.

By combining the assessments of exploitation likelihood and potential Impact, Aegisbyte provides a comprehensive risk profile for mobile applications, enabling organizations to make informed decisions regarding their security measures and prioritize mitigation efforts accordingly. Our expertise in evaluating and quantifying these risks ensures that our clients have a precise understanding of the security posture of their mobile applications, thereby empowering them to proactively safeguard their digital assets.



OVERALL RISK RATING: **MODERATE**

Summary of Strengths

As Aegisbyte was entrusted with the responsibility of identifying issues and vulnerabilities within the prevailing environment, it is essential to acknowledge and highlight any positive findings that emerge. Recognizing the strengths of the current environment serves to fortify security best practices and offers valuable insights for developing a comprehensive defensive posture. During the assessment of the Headwind Remote Android application(s), the following traits were identified as noteworthy strengths:

1. Implementation of robust user authentication: The Headwind Remote Android application(s) demonstrated a strong implementation of user authentication mechanisms. This indicates that appropriate measures have been taken to ensure the authentication process is secure, reliable, and resistant to unauthorized access attempts.

2. Integration of stack-smashing protection: The applications examined exhibited the incorporation of stack-smashing protection techniques. This defensive measure safeguards against stack-based buffer overflows, a common exploit technique used to compromise the security of applications. By employing stack-smashing protection, the Headwind Remote Android application(s) demonstrate a proactive approach in preventing such vulnerabilities and potential attacks.

Summary of Weaknesses

Aegisbyte, an esteemed cybersecurity firm, undertook an extensive examination of the Headwind Remote system, unearthing and thoroughly scrutinizing numerous vulnerabilities. These vulnerabilities have been meticulously classified into overarching weaknesses present in the current environment. Furthermore, Aegisbyte is committed to offering comprehensive guidance on the appropriate remedial actions that should be taken to fortify the enterprise's security posture.

The vulnerabilities identified by Aegisbyte include instances of sensitive information disclosure through various methods. Notably, clear text HTTP request values were discovered within the strings.xml files, potentially exposing critical data. Additionally, the application(s) lacked root (Android) detection mechanisms, leaving them susceptible to exploitation by malicious actors. Moreover, on the Android platform, the absence of SSL certificate pinning facilitated the interception of the secret key through a "man-in-the-middle" attack during our testing process. These findings underscore the pressing need for immediate attention and mitigation efforts to safeguard the enterprise's sensitive assets.

Strategic Recommendations

Not all security vulnerabilities are solely of a technical nature and cannot be fully addressed by security personnel alone. Companies often need to focus on identifying the underlying security issues and resolving them at their fundamental level. These strategic measures involve making changes to the operational policies of the organization. Aegisbyte recommends the following strategic steps to enhance the security posture of the company:

1. Implementation of SSL certificate pinning on both the iOS and Android applications: Ensure that the applications only trust specific SSL certificates, thus mitigating the risk of unauthorized or fraudulent certificates.
2. Incorporation of checks for jailbroken (iOS) or rooted (Android) devices into each application: Implement mechanisms to detect if the device has undergone jailbreaking (iOS) or rooting (Android), as these actions can compromise the security of the applications. Apply appropriate security measures to prevent unauthorized access or tampering on such devices.
3. Transmitting and displaying data without persistent storage: Develop mechanisms to handle data in a way that avoids storing it in long-term memory or on disk. However, it is crucial to remain vigilant against analog leaks, such as accidental saving of data screenshots to disk, and take necessary measures to prevent such leaks.
4. Exclusive storage of sensitive data in RAM (Random Access Memory): Store sensitive data solely in the device's RAM, ensuring that it is not stored in non-volatile memory or on disk. This reduces the risk of data exposure in the event of a security breach or unauthorized access.
5. Secure encryption of data: Apply robust encryption algorithms to protect sensitive data. Additionally, employ a master key that is encrypted with a passphrase, which must be provided when the application starts, to further strengthen the encryption scheme.
6. Utilization of secure protocols and cipher suites for encrypted communication: Implement industry-standard secure protocols and cipher suites for establishing encrypted channels between the applications and their backend services. Avoid creating custom encryption protocols or algorithms, as they may introduce unknown vulnerabilities.

SUMMARY VULNERABILITY OVERVIEW

The rapid advancement of technology continually introduces new security risks, and mobile computing is no exception. Mobile apps pose distinct security concerns that differ from those of traditional desktop software. Although modern mobile operating systems are often considered more secure than their desktop counterparts, overlooking security considerations during mobile app development can lead to various issues. Key considerations encompass aspects such as data storage, inter-app communication, proper utilization of cryptographic APIs, and secure network communication.

To address these concerns and establish a framework for mobile app security, the Open Web Application Security Project (OWASP) Mobile Application Security Verification Standard (MASVS) offers a comprehensive model. MASVS defines a set of generic security requirements specifically tailored for mobile apps. This standard serves as a valuable resource for architects, developers, testers, security professionals, and consumers alike, aiding in the definition and comprehension of the essential qualities of a secure mobile app.

In conjunction with MASVS, the OWASP Mobile Application Security Testing Guide (MASTG) aligns with the core security requirements outlined in MASVS. Depending on the specific context, MASTG can be utilized independently or combined with MASVS to achieve diverse objectives in terms of mobile app security assessment and testing.

Aegisbyte conducted a Mobile Application Penetration Test for Headwind Remote from July 1, 2023, to July 10, 2023. This assessment involved the use of commercial and proprietary tools to initially map and gather information about the application(s), as well as the utilization of custom tools and scripts to identify unique vulnerabilities. The manual analysis conducted by our assessors aimed to exploit the discovered vulnerabilities and evaluate the presence of key security flaws, including those outlined in the OWASP Top 10 Vulnerabilities list. The identification of the following vulnerabilities as high-risk was based on various factors such as the criticality of the assets involved, the likelihood of a threat, and the severity of the vulnerabilities.

VULNERABILITY RISK DEFINITION AND CRITERIA

The risk ratings assigned to each vulnerability are determined by averaging several aspects of the exploit and the environment, including reputation, difficulty, and criticality.

CRITICAL	Critical vulnerabilities pose a serious threat to an organization's security, and should be fixed immediately. They may provide a total compromise of the target environment, or similar critical impacts.
HIGH	High risk vulnerabilities provide a serious risk to the company environment and should be corrected promptly. These issues can significantly affect the organization's security posture.
MODERATE	Medium severity vulnerabilities represent a moderate risk to the environment. They may require additional context before remediation but should be remediated after critical and high risks.
LOW	Low severity vulnerabilities provide minimal risk to the target environment, and are often theoretical in nature. Remediation of low risks is often a lower priority than other security hardening techniques.
INFORMATIONAL	Informational vulnerabilities have little-or-no impact to the target scope by themselves. They are included however, as they may be a risk when combined with other circumstances or technologies not currently in place. Remediation of informational items is not necessary.

VULNERABILITY SUMMARY TABLE

The following vulnerabilities were found within each risk level. It is important to know that total vulnerabilities are not a factor in determining risk level. Risk level depends upon the severity of the vulnerabilities found.

0	1	4	0	3
Critical	High	Medium	Low	Info

Vulnerability ID - Name And Remediation	Risk Level
H1 - USE OF A ONE WAY HASH WITHOUT A SALT	HIGH
M1 - ATTRIBUTE “usesCleartextTraffic” SET	MEDIUM
M2 - CLEAR TEXT HTTP REQUEST	MEDIUM
M3 - FOUND KEYWORDS (APPLICATION LOGS)	MEDIUM
M4 - FOUND KEYWORDS (MEMORY DUMP)	MEDIUM
I1 - ANDROID SOURCE OBFUSCATED	INFO
I2 - ANDROID MITM POSSIBLE	INFO
I3 - OVER PRIVILEGED PERMISSIONS	INFO

VULNERABILITY FINDINGS

Use of a One Way Hash without a Salt

Description

The application protects passwords with digest in computeMd5Hash, of /hmdm-control-master/app/src/main/java/net/majorkernelpanic/streaming/rtsp/RtspClient.java at line 543, using a cryptographic hash password. However, the code does not salt the hash with an unpredictable, random value, allowing an attacker to reverse the hash value.

Typical cryptographic hashes, such as SHA-1 and MD5, are incredibly fast. Combined with attack techniques such as precomputed Rainbow Tables, it is relatively easy for attackers to reverse the hashes, and discover the original passwords. Lack of a unique, random salt added to the password makes brute force attacks even simpler.

```
361
362             String uri = "rtsp://"+mParameters.host+":"+mParameters.port+mParameters.path;
363             String hash1 = computeMd5Hash(mParameters.username+":"+m.group(1)+":"+mParameters.password);
364             String hash2 = computeMd5Hash("ANNOUNCE"+":"+uri);
365             String hash3 = computeMd5Hash(hash1+":"+m.group(2)+":"+hash2);
366
367             mAuthorization = "Digest username=\""+mParameters.username+"\",realm=\""+realm+"\",nonce=\""+nonce+"\",uri=\""+uri+"\",response=\""+hash3+"\"";
368
369             request = "ANNOUNCE rtsp://"+mParameters.host+":"+mParameters.port+mParameters.path+" RTSP/1.0\r\n" +
370                     "CSeq: " + (++mCSeq) + "\r\n" +
371                     "Content-Length: " + body.length() + "\r\n" +
372                     "Authorization: " + mAuthorization + "\r\n" +
373                     "Session: " + mSessionID + "\r\n" +
374                     "Content-Type: application/sdp\r\n\r\n" +
375                     body;
376
377             Log.i(TAG,request.substring(0, request.indexOf("\r\n")));
378
379             mOutputStream.write(request.getBytes("UTF-8"));
380             mOutputStream.flush();
381             response = Response.parseResponse(mBufferedReader);
```

Figure 1: *Highlighted source line shows using a cryptographic hash password. Tracing the code, the code does not salt the hash with an unpredictable, random value, allowing an attacker to reverse the hash value.*

Risk

If an attacker gains access to the hashed passwords, the attacker would likely be able to reverse the hash due to this weakness, and retrieve the original password. Once the passwords are discovered, the attacker can impersonate the users, and take full advantage of their privileges and access their personal data. Furthermore, this would likely not be discovered, as the attacker is being identified solely by the victims' credentials.

Recommendation

1. Always use strong, modern algorithms for encryption, hashing, and so on.
2. Do not use weak, outdated, or obsolete algorithms.
3. Ensure you select the correct cryptographic mechanism according to the specific requirements.
4. Passwords should be protected using a password hashing algorithm, instead of a general cryptographic hash. This includes adaptive hashes such as bcrypt, scrypt, PBKDF2 and Argon2.
5. Tune the work factor, or cost, of the adaptive hash function according to the designated environment and risk profile.
6. Do not use a regular cryptographic hash, such as SHA-1 or MD5, to protect passwords, as these are too fast.
7. If it is necessary to use a common hash to protect passwords, add several bytes of unique, random data ("salt") to the password before hashing it. Store the salt with the hashed password, and do not reuse the same salt for multiple passwords.

```
private String protectPassword(String password) {
   byte[] data = password.getBytes("UTF-8");
    byte[] hash = null;

    try {
       SecureRandom rand = new SecureRandom();
       byte[] salt = new byte[32];
       rand.nextBytes(salt);

       SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
       PBEKeySpec spec = new PBEKeySpec(data, salt, ITERATION_COUNT, KEY_LENGTH);
       // ITERATION_COUNT should be configured by environment, KEY_LENGTH should be 256
       SecretKey key = skf.generateSecret(spec);

       hash = key.getEncoded();
   }
   catch (GeneralSecurityException gse) {
       handleCryptoErrors(gse);
   }
    finally {
       Arrays.fill(data, 0);
   }

    return Base64.getEncoder().encodeToString(hash);
}
```

Common Weakness Enumeration (CWE)

CWE-759

Attribute usesCleartextTraffic set

Description

The android:usesCleartextTraffic attribute indicates whether the app intends to use cleartext network traffic, such as cleartext HTTP. The default value for apps that target API level 27 or lower is "true". Apps that target API level 28 or higher default to "false".

```
<application android:theme="@style/AppTheme" android:label="@string/app_name"
android:icon="@mipmap/ic_launcher" android:allowBackup="true"
android:supportsRtl="true" android:extractNativeLibs="false"
android:usesCleartextTraffic="true" android:roundIcon="@mipmap/ic_launcher_round"
android:appComponentFactory="androidx.core.app.CoreComponentFactory">
        <activity android:theme="@style/AppTheme_NoActionBar"
android:label="@string/app_name" android:name="com.hmdm.control.MainActivity"
android:launchMode="singleTask">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
```

Recommendation

Explicitly set the attribute android:usesCleartextTraffic value to false and define an Android Network Security Config. The default value for apps that target API level 27 or lower is true. Apps that target API level 28 or higher default to false

References

Storage updates in Android 11

Clear text HTTP request

Description

Mobile Applications must use Transport Layer Security (TLS) to provide encryption at the transport layer and ensure the confidentiality and integrity of data in transit. This application does not use SSL/TLS and is vulnerable to traffic interception and modification.

An attacker performing a man-in-the-middle (MITM) attack may:

1. Passively intercept the communication to access any sensitive data in transit like usernames, passwords, or credit card number.
2. Actively inject or remove content to forge and omit information or inject malicious scripts
3. Actively redirect the communication to the attacker in the context of the initial trusted party

res/values-de/strings.xml (line 61):

```
<string name="enter_correct_url">Bitte geben Sie eine korrekte Server-URL ein, die mit http:// oder https:// beginnt</string>
```

res/values/strings.xml (line 71):

```
<string name="enter_correct_url">Please enter a correct server URL starting with http:// or https://</string>
```

Resource is used in the method saveSettings with the Resource id 2131755085:

```
public static final int enter_correct_url = 2131755075;
```

Code showing the Resource id (2131755085) is shown below:

```
private boolean saveSettings()
  {
    com.hmdm.control.SettingsHelper v0_8 = this.editTextServerUrl.getText().toString();
    if ((v0_8.startsWith(http://)) || (v0_8.startsWith(https://))) {
      int v1_4 = this.editTextSecret.getText().toString();
      if (!v1_4.trim().equals()) {
        String v4_4 = this.editTextDeviceName.getText().toString();
        if (!v4_4.trim().equals()) {
          if (!this.areSettingsChanged()) {
            this.setResult(-1);
          } else {
            this.setResult(2000);
          }
          this.settingsHelper.setString(server_url, v0_8);
          this.settingsHelper.setString(secret, v1_4);
          this.settingsHelper.setBoolean(use_default, this.checkBoxUseDefault.isChecked());
          this.settingsHelper.setBoolean(translate_audio, this.checkBoxTranslateAudio.isChecked());
          this.settingsHelper.setInt(bitrate,
com.hmdm.control.SettingsActivity.bitrates[this.spinnerBitrate.getSelectedItemPosition()]);
          this.settingsHelper.setInt(frame_rate,
com.hmdm.control.SettingsActivity.frame_rates[this.spinnerFrameRate.getSelectedItemPosition()]);
          this.settingsHelper.setString(device_name, v4_4);
          this.settingsHelper.setString(dst_ip, this.editTextTestDstIp.getText().toString());
          this.settingsHelper.setBoolean(notify_sharing, this.checkBoxNotifySharing.isChecked());
          this.settingsHelper.setInt(idle_timeout,
com.hmdm.control.SettingsActivity.idle_timeouts[this.spinnerIdleTimeout.getSelectedItemPosition()]);
          return 1;
        } else {
          android.widget.Toast.makeText(this, 2131755086, 1).show();
          return 0;
        }
      } else {
        android.widget.Toast.makeText(this, 2131755087, 1).show();
        return 0;
      }
    } else {
      android.widget.Toast.makeText(this, 2131755085, 1).show();
      return 0;
    }
  }
```

Recommendation

1. It is recommended to ensure the use of an encrypted channel for requests transmitting sensitive data. It is, however, highly recommended to encrypt all requests made by the application, as the interception and modification of nonsensitive requests could be leveraged to access sensitive data.
2. The encrypted channel should use secure protocols and cipher suites, do not develop custom encryption protocols or algorithms.

References

- Missing Encryption of Sensitive Data (CWE-311)
- Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection
- Top 10 2013-A6-Sensitive Data Exposure
- Insufficient Transport Layer
- Cleartext Transmission of Sensitive Information (CWE-319)

Found Keywords (Application Files)

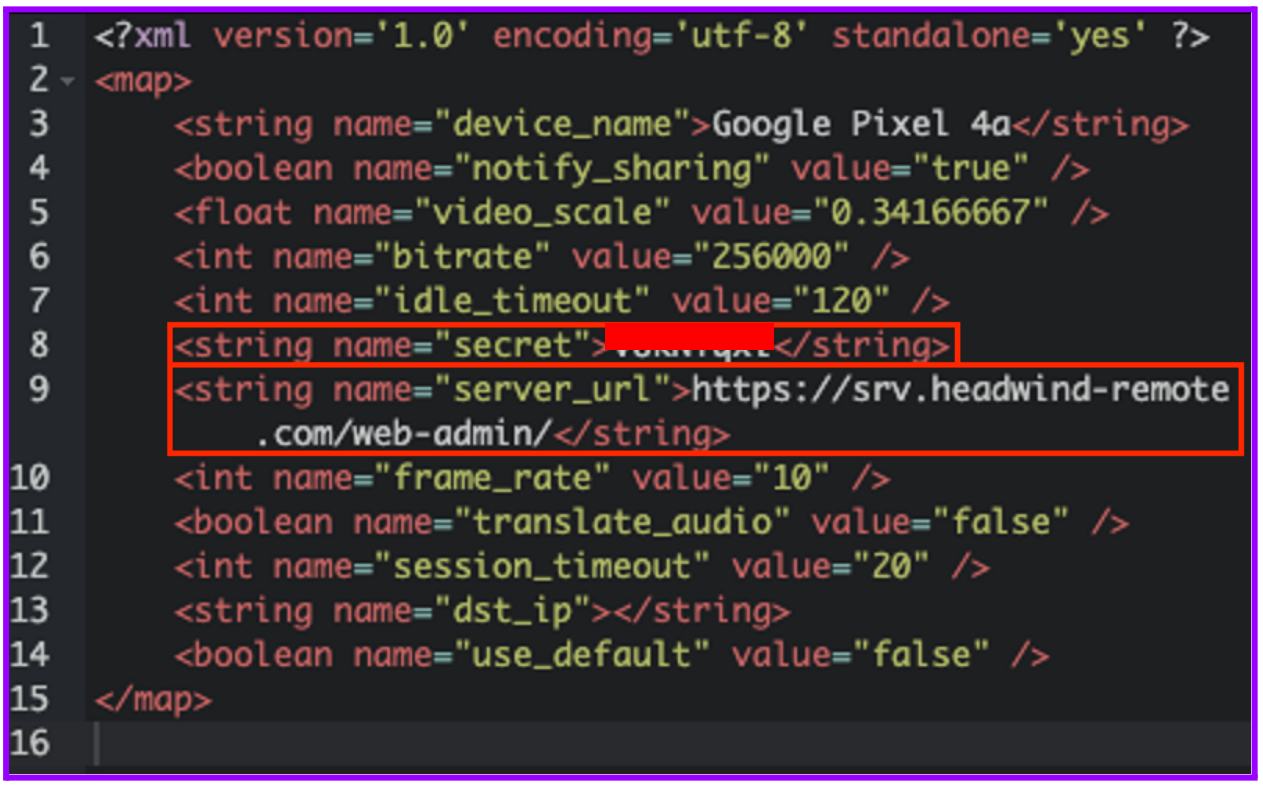
An instance of sensitive data has been identified through the utilization of specific values or patterns inputted during the process of data population (Setup > Search Terms). The device has yielded results in the 'Search Results' output, and it is advised to examine this output to ascertain the exact location of each identified value.

Recommendations

It is recommended to minimize the storage of sensitive data whenever feasible. To achieve this, consider the following options:

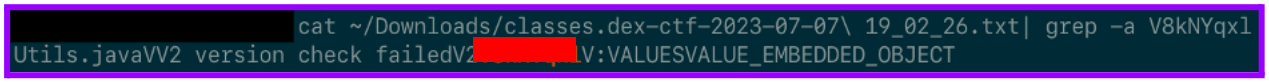
1. Transmit and display the data without persisting it in memory. However, it is crucial to pay special attention to prevent analog leaks, such as scenarios where screenshots of the data are inadvertently saved to disk.
2. Store the sensitive data exclusively in RAM (Random Access Memory) and ensure that it is cleared from memory when the application is closed.
3. Encrypt the data using strong encryption, combined with a master key encrypted with a passphrase required when the application starts.

```
Type
Password (plain-text)
Value
V******1
Path
/com.hmdm.control/shared_prefs/com.hmdm.control.PREFERENCES.xml
Type
Keywords (plain-text)
Value
https://srv.headwind-remote.com/web-admin/
Path
/com.hmdm.control/shared_prefs/com.hmdm.control.PREFERENCES.xml
```



```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="device_name">Google Pixel 4a</string>
    <boolean name="notify_sharing" value="true" />
    <float name="video_scale" value="0.34166667" />
    <int name="bitrate" value="256000" />
    <int name="idle_timeout" value="120" />
    <string name="secret">[redacted]</string>
    <string name="server_url">https://srv.headwind-remote
        .com/web-admin/</string>
    <int name="frame_rate" value="10" />
    <boolean name="translate_audio" value="false" />
    <int name="session_timeout" value="20" />
    <string name="dst_ip"></string>
    <boolean name="use_default" value="false" />
</map>
```

Figure 2: *Secret Key and URL found in application files Version 1.35*



```
[redacted] cat ~/Downloads/classes.dex-ctf-2023-07-07\ 19_02_26.txt| grep -a V8kNYqxl
Utils.javaVV2 version check failedV2[redacted]V:VALUESVALUE_EMBEDDED_OBJECT
```

Figure 3: *Secret Key 'classes.dex' file on Version 1.30*

Regulatory

CWE: 312

CWE: 313

CWE: 215

OWASP Mobile Top 10: M2-Insecure Data Storage OWASP Mobile Top 10: M3-Insecure Communications NIAP: FDP_DEC_EXT.1.5

FFIEC: May violate D3.PC.Am.A.1

GDPR: May violate Article 25

GDPR: May violate Article 32

PCI: May violate requirement 3.1 through 3.4

CVSS

4.0

Vector

Vector String: CVSS:3.0/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:N

Found Password (Logfile)

The password provided has been identified within the file(s) indicated in the output of the Search Results. This constitutes a crucial piece of information that should never be stored in a vulnerable format susceptible to decryption.

Ensuring secure data storage on a mobile device necessitates the implementation of proper techniques. It is highly recommended to refrain from storing or caching data whenever feasible, as this approach provides the most reliable means of preventing data compromise on the device.

In the context of Android, it is crucial to acknowledge that external storage, such as an SD Card, lacks fine-grained permissions. By default, any application possesses read access to the storage and can access all files. However, starting from Android 4.4, apps can securely store data on the SD Card under specific conditions (refer to http://source.android.com/devices/tech/storage/ for more information).

Both Android and iOS employ standard cryptographic libraries, including Advanced Encryption Standard (AES), which can be utilized to safeguard data. It is essential to note that the security of data encrypted with AES is only as strong as the password used to derive the encryption key and manage it. Therefore, it is advisable to consider the password policy, taking into account factors such as length, complexity, user convenience, and the method by which the encryption key is stored in memory. It is worth noting that if root access is obtained, it is possible to extract the memory of a running process and search for encryption keys within it.

Furthermore, it is important to be aware that employing the default cryptographic provider "AES" often results in the less secure AES-ECB mode. Following best practices, it is recommended to specify AES-CBC or AES-GCM modes with a 256-bit key and a random Initialization Vector (IV) generated using a secure source like SecureRandom. Additionally, deriving the encryption key from a passphrase using a well-tested function like Password-Based Key Derivation Function 2 (PBKDF2) is advisable.

Type	Value	Contents in Log File
Password (plain-text)	V******l	07-05 09:58:08.969 20739 22450 V H264Packetizer: SPS or PPS present in the stream. 07-05 09:58:09.047 20739 20811 D OkHttp : <-- 200 https://srv.headwind-remote.com/janus/2695087570625448?apisecret=V******l (30042ms) 07-05 09:58:09.047 20739 20811 D OkHttp : server: nginx -- 07-05 09:58:09.049 20739 20811 D com.hmdm.Control: Janus: Keep-Alive 07-05 09:58:09.050 20739 20811 D OkHttp : --> GET https://srv.headwind-remote.com/janus/2695087570625448?apisecret=V******l http/1.1 07-05 09:58:09.050 20739 20811 D OkHttp : --> END GET -- 07-05 09:58:39.082 20739 22450 V H264Packetizer: SPS or PPS present in the stream. 07-05 09:58:39.119 20739 20811 D OkHttp : <-- 200 https://srv.headwind-remote.com/janus/2695087570625448?apisecret=V******l (30069ms) 07-05 09:58:39.119 20739 20811 D OkHttp : server: nginx -- 07-05 09:58:39.121 20739 20811 D com.hmdm.Control: Janus: Keep-Alive 07-05 09:58:39.122 20739 20811 D OkHttp : --> GET https://srv.headwind-remote.com/janus/2695087570625448?apisecret=V******l http/1.1 07-05 09:58:39.122 20739 20811 D OkHttp : --> END GET

Figure 4: *Password Found in Logfile on Headwind Remote Version 1.35*

Figure 5: *Password Found in Logfile on Headwind Remote Version 1.30*

Recommendations

When sensitive data, such as a password, needs to be stored locally, it is imperative to employ appropriately implemented encryption to introduce substantial complexity to potential attacks. In this scenario, a master key should be encrypted using a user passphrase. To achieve this, a key derivation function like PBKDF2 should be utilized, with a significantly high number of iterations. This approach aims to consume a considerable amount of CPU resources, thereby rendering brute-force attempts computationally time-consuming, even on faster platforms.

Regulatory

CWE: 256
CWE: 522
CWE: 215
OWASP Mobile Top 10: M2-Insecure Data Storage OWASP Mobile Top 10: M3-Insecure Communications NIAP: FCS_STO_EXT.1.1
FFIEC: May violate D3.PC.Am.A.1
GDPR: May violate Article 25
GDPR: May violate Article 32
FISMA Medium: May violate C-28 PROTECTION OF INFORMATION AT REST FISMA Medium: May violate SI-10 INFORMATION INPUT VALIDATION HIPAA: May violate §164.312(a)(1)
PCI: May violate requirement 3.1 through 3.4

CVSS

4.3

Vector

Vector String: CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Found Password (Memory Dump)

The presence of the application password was discovered in the memory dump.

During the operation of an application, both user-specific and application-specific data may be stored in the Random Access Memory (RAM) without being appropriately cleared when the user logs out or the session times out. In the case of Android, an application remains in memory until the memory is reallocated, even after it has been used. Consequently, encryption keys associated with the application may persist in the memory. In the event that an attacker gains access to the device, they can utilize a debugger to extract the application's memory dump or employ a kernel module to retrieve the complete contents of the RAM.

When handling passwords and other sensitive information, applications often retain this data in memory, even if the buffer is freed temporarily. This can pose a security concern, particularly if the application is susceptible to vulnerabilities such as buffer overflow, format string exploits, or data leaks. Exploiting these vulnerabilities, an attacker may be able to extract the process's memory and thereby retrieve the sensitive information stored within it.

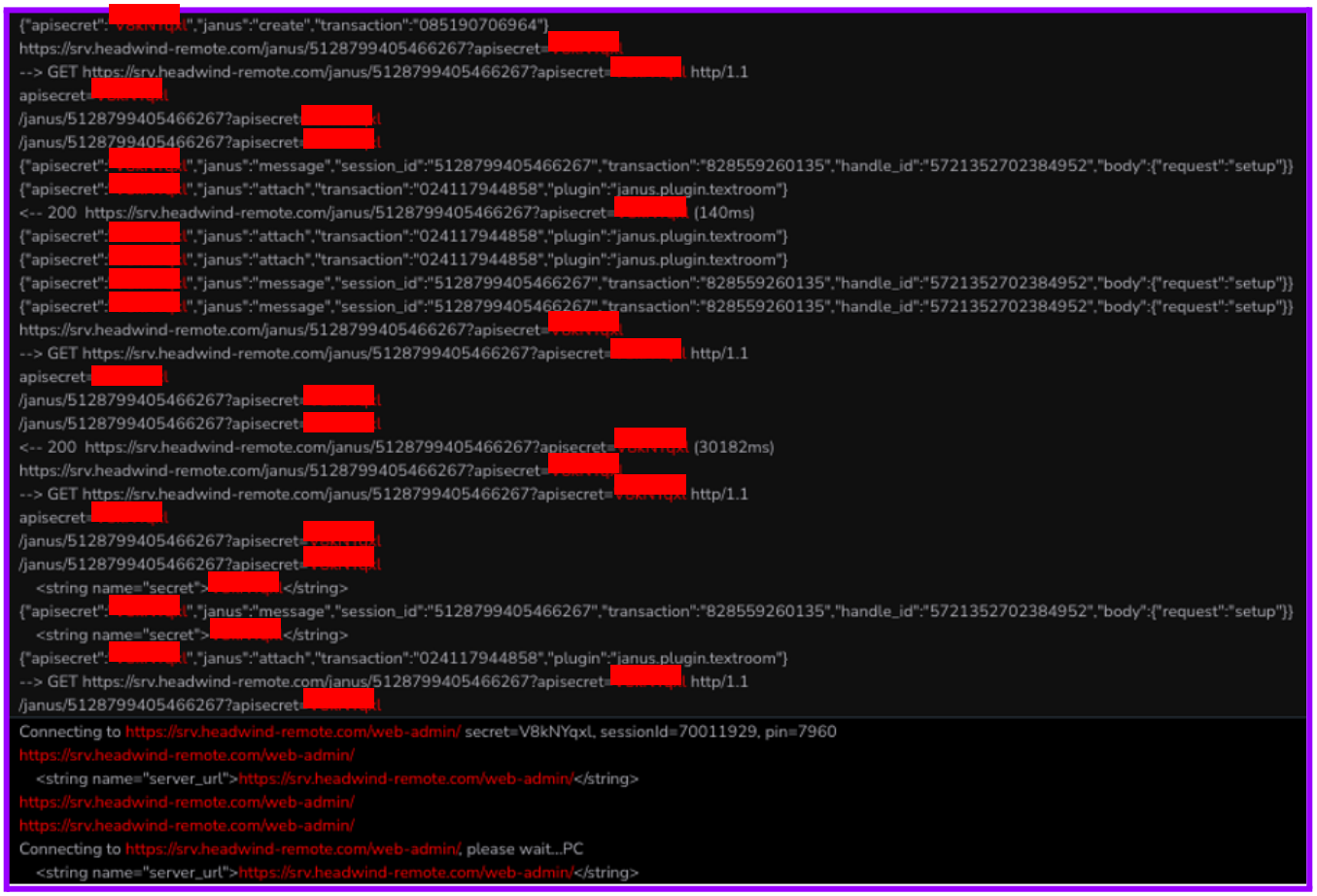


Figure 6: *Password Found in Memory Dump on Headwind Remote Version 1.35*

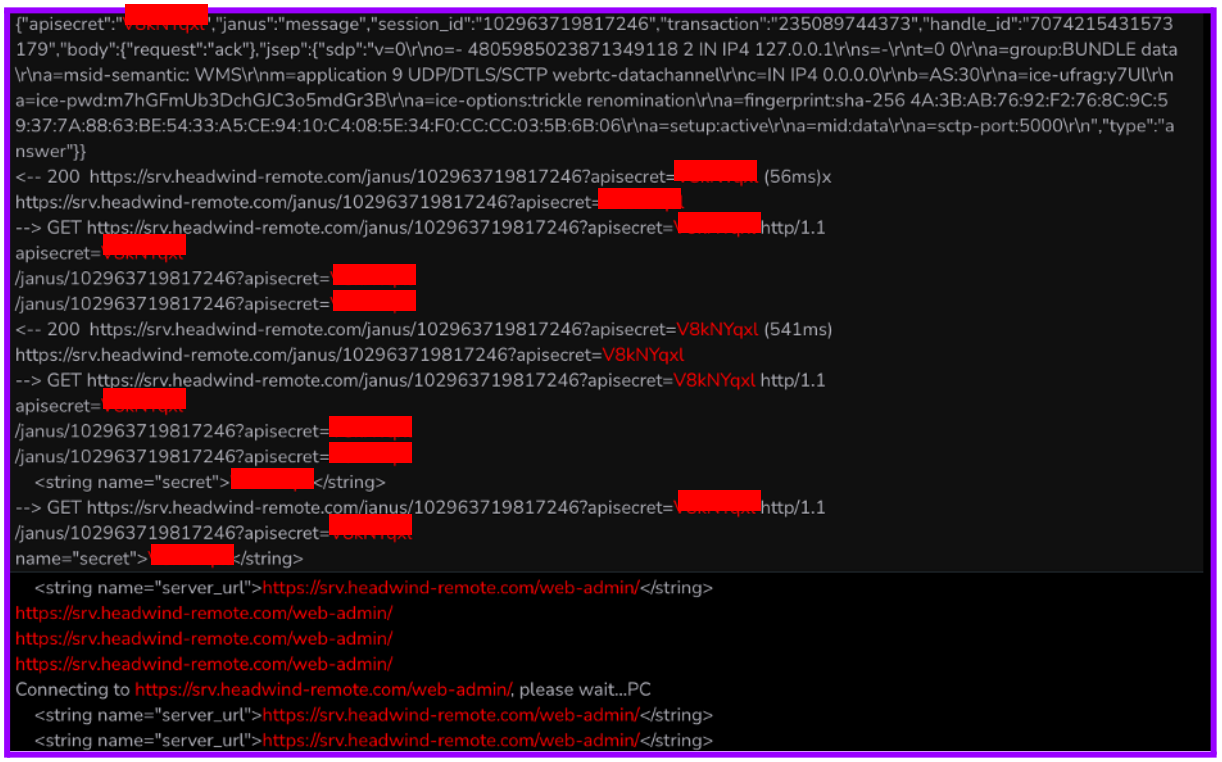


Figure 7: *Password Found in Logfile on Headwind Remote Version 1.30*

Recommendations

To minimize the risk associated with sensitive data, such as encryption keys, it is crucial not to keep them in the Random Access Memory (RAM) for longer than necessary. It is recommended to nullify variables that hold these keys after they have been used. Furthermore, it is advised to avoid using immutable objects, like the java.lang.String in Android, for storing sensitive keys or passwords, and instead opt for a char array. Immutable objects, even if their references are removed or nulled, may persist in memory until garbage collection occurs, which cannot be forced by the application.

However, it is important to note that achieving these measures effectively can be challenging in high-level languages, as compilers and just-in-time virtual machines often prioritize performance and may ignore such operations if they detect that the buffer is no longer in use after being overwritten.

Clearing buffers in a manner that bypasses compiler optimizations is possible, but it requires specific considerations based on the toolchain, programming language, and platform being used. Recommendations for achieving this level of buffer clearing are dependent on the specific toolchain, language, and platform, and therefore may vary.

Regulatory

M2 - Insecure Data Storage
CWE-316: Cleartext Storage of Sensitive Information in Memory
CWE-200: Information Exposure

ADDITIONAL INFORMATION APK (1.30 AND VERSION 1.35)

Android Source Obfuscated

The source code of the Android app has undergone obfuscation, which has rendered it less susceptible to straightforward reverse-engineering, accounting for approximately 19% of its protection. This measure serves to safeguard the app's intellectual property by making it more challenging for unauthorized individuals to understand and replicate the underlying code structure and logic.

Recommendation

It is advisable to verify that all of your Java classes, which encompass sensitive information, including those associated with external libraries linked to your application, have been adequately protected by your chosen obfuscation tool. This precautionary step ensures that the sensitive components within your code, as well as any dependencies from external libraries, have undergone effective obfuscation measures to mitigate the risk of unauthorized access or reverse-engineering.

Regulatory

CWE: 656
OWASP Mobile Top 10: M8-Code Tampering
OWASP Mobile Top 10: M9-Reverse Engineering

Android MiTM Possible

Mobile applications often store authentication information, credentials, tokens, or other artifacts to maintain session state and enhance the user experience. It is crucial to implement measures to safeguard this valuable information, as stale sessions can be targeted and exploited by attackers in their attacks. In many instances, the replay (reuse) of such sessions can go undetected for extended periods, potentially compromising the security of the application.

Mitigating Man-in-the-Middle (MitM) attacks and malicious proxy attacks is a fundamental aspect of cyber-defense strategy. Mobile MitM attacks specifically focus on intercepting and manipulating the communication between a mobile app and the server it connects to. Attackers employ various techniques, such as attaching proxies to insecure network or Wi-Fi connections, exploiting stale session IDs, modifying or redirecting DNS requests, among others, to execute these attacks.

If the attacker gains control over the user's network, they may attempt to impersonate the server-side and replace the server's legitimate certificate with their own fraudulent or malicious certificate.

To counteract this threat, it is essential to identify and block malicious certificates during the SSL Handshake. Applications should incorporate mechanisms to validate the authenticity of the SSL certificate used by the destination server. This validation process ensures that the application connects only to trusted, known, and secure destinations, servers, or websites, effectively preventing connections to untrusted or malicious entities.

Figure 8: *Secret captured with MITMPROXY*

Recommendations

1. Utilize secure protocols such as HTTPS/TLS to encrypt the communication between the mobile app and the server, preventing unauthorized interception and tampering.
2. During the SSL Handshake, validate the authenticity of the server's SSL certificate to ensure it is issued by a trusted Certificate Authority (CA) and matches the expected identity of the server.
3. Consider implementing certificate pinning to further enhance security. This involves associating the server's SSL certificate with its public key or fingerprint, allowing the app to verify the certificate's authenticity explicitly.

Overpriviledged Permissions on Android Application

The application in question has declared permissions that remain unused during its execution. The comprehensive list of such permissions can be found in the output labeled "Overprivileged Permissions." Adhering to the principle of "least privilege" while coding an application ensures that it only accesses the necessary resources, thereby mitigating the risk of a malicious attacker gaining unauthorized access beyond the intended scope of the application.

It should be noted that false-positive results may arise during static analysis if the application employs reflection (loading a class by name or invoking a method by name) or utilizes a shared ID.

Permission(s) declared but not used (Overprivileged Permissions):

```
android.permission.FOREGROUND_SERVICE
android.permission.SYSTEM_ALERT_WINDOW
```

Recommendations

1. Carefully analyze the list of over privileged permissions identified in the "Overprivileged Permissions" output. Identify the permissions that are not required for the application's intended functionality. Remove those unused permissions from the application's manifest or permission declaration.
2. Ensure that the application follows the principle of least privilege while requesting permissions. Only request the specific permissions necessary for the application's legitimate operation. Avoid requesting excessive permissions that are not directly related to the functionality.
3. Conduct periodic audits of the application's permissions to identify any unused or unnecessary permissions. As the application evolves, new features or changes may render certain permissions obsolete. By regularly reviewing and updating permissions, you can minimize the risk of over privileged access.

Regulatory

CWE: 250
OWASP Mobile Top 10: M1-Improper Platform Usage
OWASP Mobile Top 10: M10-Extraneous Functionality
FISMA Medium: May violate C-6 LEAST PRIVILEGE

Malware Analysis on Android Binaries

Description

Aegisbyte executed the analysis of the sample on an actual hardware device with clean, reflash firmware. The AndroidManifest.xml file was used for examination.

The rafind2 program within Radare was employed to extract strings of interest from the file. It should be noted that the "strings" command did not function properly in this case. The following commands were executed:

```
$ rafind2 -ZS android AndroidManifest.xml
```

```
$ rafind2 -ZS permissions AndroidManifest.xml
```

```
$ rafind2 -ZS intent AndroidManifest.xml
```

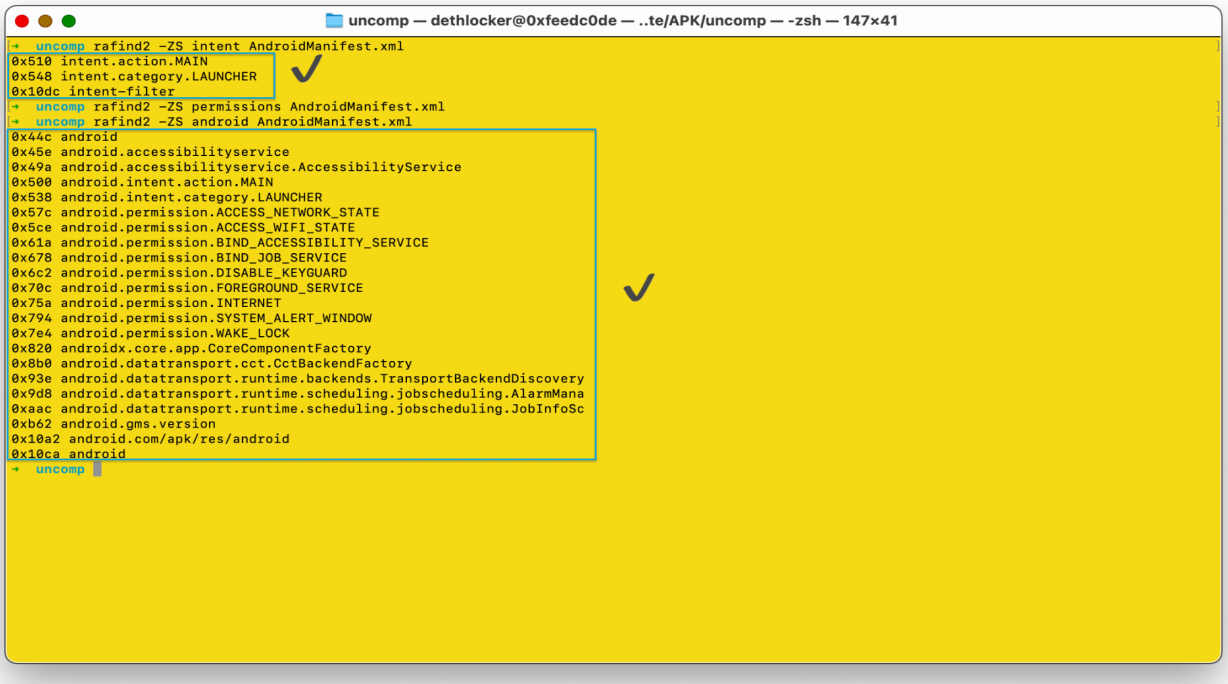


Figure 9: *Malware Analysis Output. No excessive or intrusive permissions detected.*

It is worth mentioning that some common permissions employed by malware were not analysis identified during the analysis, which generally include:

```
permission.CHANGE_WIFI_STATE
permission.CHANGE_NETWORK_STATE
permission.INSTALL_PACKAGES
permission.INSTALL_SHORTCUT
permission.SYSTEM_OVERLAY_WINDOW
permission.ACCESS_DOWNLOAD_MANAGER
permission.MOUNT_UNMOUNT_FILESYSTEMS
permission.RECORD_AUDIO
permission.RECEIVE_BOOT_COMPLETED
permission.KILL_BACKGROUND_PROCESSES
permission.ACCESS_MTK_MMHW
permission.DISABLE_KEYGUARD
permission.SYSTEM_ALERT_WINDOW
permission.GET_TASKS
```

Additional

Recently, CRIL has identified a malicious shortened URL, **hxxps[://]bit[.]ly/nubankmodulo**, which redirects users to the GoatRAT URL, **hxxps://goatrat[.]com/apks/apk20.apk**. The website hosting this URL contains Android malware, and the downloaded APK file is named "apk20.apk," masquerading as an application associated with NU bank, a Brazilian bank. Our analysis through PCAP, dynamic analysis, and source code review of the headwind Remote application did not reveal any evidence of it reaching out to this URL or being part of the IOCs listed in Appendix D.

Upon analyzing the malicious APK file, we did not find any instances of the malware certificate "**38661ea0b53f278f620a3f2c8db6da8ef8ca890e**," which is also present in other malicious applications associated with GoatRAT. Additionally, the domain h**xxps[://]goatrat[.]com**, from which the APK is downloaded, serves as the administrative panel for GoatRAT.

Originally developed as an Android Remote Administration Tool, GoatRAT has now evolved into a Banking Trojan primarily targeting Brazilian banks. Similar to other Banking Trojans, GoatRAT exploits the Accessibility service to implement an Automatic Transfer System (ATS) framework, allowing the fraudulent transfer of funds from the victim's account using the PIX key. The PIX key is utilized for instant payments. Currently, this malware targets three Brazilian banks: NUBank, Banco Inter, and PagBank.

Aegisbyte conducted thorough testing on the following APK versions and did not detect any traces of the GoatRat remote access trojan. This finding aligns with the analysis conducted by MalwareBytes, confirming that the code within the APK is clean on the APK versions tested on this report.

Reference Link to MalwareBytes:
https://forums.malwarebytes.com/topic/299334-headwind-remote-for-android-is-detected-as-androidtrojanagentgx/

ESET NOD32 detected the presence of Android/Spy.GoatRat.C. The hostile code was found in all the versions, namely 1.35, and 1.30. Aegisbyte's recent analysis, CRIL (Cybersecurity Research and Intelligence Lab) has identified Indicators of Compromise (IOCs) and other artifacts, which are listed in Appendix D.

During our investigation, we noticed certain resemblances between the source code of Headwind Remote and GoatRat. Specifically, GoatRat includes the following receivers and services:

com.goatrat.GestureDispatchService
com.goatrat.janus.JanusSessionPollService
com.goatrat.ScreenSharingService

Similarly, Headwind Remote also features similar components:

com.hmdm.control.GestureDispatchService
com.hmdm.control.janus.JanusSessionPollService
com.hmdm.control.ScreenSharingService

This similarity may be attributed to the utilization of Headwind Remote's open source code, which is publicly available on GitHub at the following link: **https://github.com/h-mdm/remote-control-android**.

Upon conducting a thorough analysis of the malicious APK file, we did not discover any traces of malicious code within the analyzed APK binaries and versions listed on the report.

Note: GoatRat is a relatively new type of malware that typically gets installed through an http URL redirecting to its installation site. Its functionalities include acting as a remote control application, injecting other malware, unauthorized remote control of devices, and harvesting information from infected Android devices.

Summary

Phishing code that pretends to come from a trustworthy source, requests a user's authentication credentials or billing information, and sends the data to a third-party

NOT DETECTED

Code that intercept the transmission of user credentials in transit.

NOT DETECTED

REPORT SUMMARY

Aegisbyte conducted comprehensive testing on the provided Android versions and binaries. The analysis revealed the following findings:

Malware Code: Not Detected
Aegisbyte did not detect any presence of malware code within the tested Android versions and binaries. This indicates that the analyzed files are clean from known malicious code.

Overall Risk Rating: Moderate
The overall risk rating for the tested Android versions and binaries is assessed as moderate. While no malware was found, there are identified vulnerabilities that pose a certain level of risk.

Vulnerability Summary:

- **HIGH:** 1 vulnerability
- **MEDIUM:** 4 vulnerabilities
- **LOW:** 0 vulnerabilities
- **INFO:** 3 pieces of information

Android Binaries Tested:

1. Android - Headwind Remote Version 1.30:
    a. Download link: https://headwind-remote.com/files/hremote-1.30-donate.aab
        i. **SHA1:** 3950c28d1af47bf6809e2a52c995f43cf7c8cb26
    b. Download link: https://headwind-remote.com/files/hremote-1.30-donate.apk
        i. **SHA1:** f528fe11b7dae67e0a52e5cc256a5646137cb840
2. Android - Headwind Remote Version 1.35:
    a. Download link: https://headwind-remote.com/files/hremote-1.35-premium.apk
        i. **SHA1:** e71af46a7a0a08d8af5a111f4b530b6891c50522

It is important to note that although no malware code was detected, the identified vulnerabilities should be addressed to mitigate potential risks. Aegisbyte recommends taking appropriate measures to address the vulnerabilities found in the tested versions and binaries.

APPENDICES

Appendix A: Tools Used

Tool	Description
Burp Suite	Burp Suite is an integrated platform for attacking web applications.
Dradis Framework	Collaboration and reporting software for information security teams.
Nessus	Nessus is one of the most popular and capable vulnerability scanners, particularly for UNIX systems.
Mobile Security Framework	Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis. MobSF supports mobile app binaries (APK, XAPK, IPA & APPX) along with zipped source code and provides REST APIs for seamless integration with your CI/CD or DevSecOps pipeline.The Dynamic Analyzer helps you to perform runtime security assessment and interactive instrumented testing.
Frida	Frida is a free open-source dynamic instrumentation toolkit for developers, reverse engineers, and security researchers
Radare2	Radare2 (also known as r2) is a complete framework for reverse-engineering and analyzing binaries; composed of a set of small utilities that can be used together or independently from the command line. Built around a disassembler for computer software which generates assembly language source code from machine-executable code, it supports a variety of executable formats for different processor architectures and operating systems.
Metasploit	Metasploit is a penetration testing software created through a collaboration between the open source community and Rapid7.
Ghidra	Ghidra is a free and open source reverse engineering tool developed by the National Security Agency (NSA) of the United States. The binaries were released at RSA Conference in March 2019; the sources were published one month later on GitHub. Ghidra is seen by many security researchers as a competitor to IDA Pro. The software is written in Java using the Swing framework for the GUI. The decompiler component is written in C++, and is therefore usable in a stand-alone form
Hopper	Hopper Disassembler is a reverse-engineering tool that lets you disassemble, decompile, and debug your applications. Hopper displays the code using different representations.

Appendix B: References

The contractor shall be familiar with Federal policies, program standards, and guidelines such as, but not limited to, those listed below or later versions as amended:

Reference	Description
FISMA	*Federal Information System Modernization Act (FISMA) (2014)*
FIPS 199	*Federal Information Processing Standards (FIPS) Publication 199 - Standards for Security Categorization of Federal Information and Information Systems*
FIPS 200	*Minimum Security Requirements for Federal Information and Information Systems*
NIST SP 800-30 Rev 1	*National Institute of Standards and Technology (NIST) Guide for Conducting Risk Assessments*
NIST SP 800-35	*Guide to Information Technology Security Services*
NIST SP 800-37 Rev 2	*Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy*
NIST SP 800-39	*Managing Information Security Risk: Organization, Mission, and Information System View*
NIST SP 800-44 Version 2	*Guidelines on Securing Public Web Servers*
NIST SP 800-53 Rev 4	*Security and Privacy Controls for Federal Information Systems and Organizations*
NIST SP 800-53A Rev 4	*Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective Assessment Plans*
NIST SP 800-61 Rev 2	*Computer Security Incident Handling Guide*
NIST SP 800-83 Rev 1	*Guide to Malware Incident Prevention and Handling for Desktops and Laptops*
NIST SP 800-86	*Guide to Integrating Forensic Techniques into Incident Response*
NIST SP 800-101 Rev 1	*Guidelines on Mobile Device Forensics*
NIST SP 800-115	*Technical Guide to Information Security Testing and Assessment*
NIST SP 800-128	*Guide for Security-Focused Configuration Management of Information Systems*

Reference	Description
NIST SP 800-137	*Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations*
NIST SP 800-150	*Guide to Cyber Threat Information Sharing*
NIST SP 800-153	*Guidelines for Securing Wireless Local Area Networks (WLANs)*
NIST SP 800-160 Vol 1	*Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*
NIST SP 800-171 Rev 1	*Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations*
NIST SP 800-171A	*Assessing Security Requirements for Controlled Unclassified Information*
NIST SP 800-181	*National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework*
P.L. 93-579	*Public Law 93-579 Privacy Act, December 1974 (Privacy Act)*
40 U.S.C. 11331	*Responsibilities for Federal Information Systems Standards*
OMB M-19-03	*Office of Management and Budget (OMB) Memorandum 19-03, Strengthening the Cybersecurity of Federal Agencies by enhancing the High Value Asset Program*
OMB A-130	*OMB Circular A-130, Managing Information as a Strategic Resource*
BOD 18-02	*Department of Homeland Security's Binding Operational Directive 18-02, Securing High Value Assets*

Appendix C: MITRE ATT&CK Techniques for GoatRAT

Tactic	Technique ID	Technique Name
Initial Access	T1476	Deliver Malicious App via Other Means.
Initial Access	T1444	Masquerade as a Legitimate Application
Discovery	T1418	Application discovery
Impact	T1516	Input Injection

Appendix D: Indicators of Compromise (IOCs)

Indicators	Indicator Type	Description
6583a9b6b83738e0bf2a261fc04483e18772da3241e467fdef37a8e27b1869a7	SHA256	Hash of analyzed malicious APK
60358f26853ccba6f137901c57147442e868041b	SHA1	Hash of analyzed malicious APK
9a8e85cf1bbd32c71f0efa42ffedf1a0	MD5	Hash of analyzed malicious APK
hxxps://bit[.]ly/nubankmodulo	URL	Shortened malware distribution URL
hxxps://goatrat[.]com/apks/apk20.apk	URL	Malware distribution URL
hxxp://api.goatrat[.]com:3008	URL	C&C server to receive PIX key

Appendix E: Assessors Credentials and Certifications

Name	Certs	Links
Jason Brewer	• GIAC Defensible Security Architect Certification (GDSA) • GIAC Critical Controls Certification (GCCC) • GIAC Certified Project Manager (GCPM) • GIAC Certified Intrusion Analyst Certification (GCIA) • GIAC Security Essentials (GSEC) • GIAC Exploit Researcher and Advanced Penetration Tester (GXPN) • GIAC Penetration Tester Certification (GPEN) • GIAC Web Application Penetration Tester (GWAPT)	• https://www.giac.org/certified-professional/Jason-Brewer/154046
Jason Bernier	• Offensive Security Certified Professional (OSCP) • Hack The Box Certified Penetration Testing Specialist (HTB CPTS) • GIAC Penetration Tester (GPEN) • GIAC Incident Handler (GCIH) • Offensive Security Wireless Professional (OSWP)	• https://www.credly.com/earner/earned/badge/2d6104eb-a712-4806-8d04-1e2e567a48e5 • https://www.credly.com/earner/earned/badge/8784d280-db87-46a4-a646-38834d19bde9 • https://www.credly.com/earner/earned/badge/11de6bd8-c2fa-4998-8eeb-0398554d121b • https://www.credly.com/earner/earned/badge/1df60625-4ccf-4a2b-8942-c48015502871 • https://www.credly.com/earner/earned/badge/7b811d52-c74e-4d6c-8ada-d59192b5f436 • https://www.credly.com/earner/earned/badge/f1ddecba-8902-46ef-802f-68241f9ea883

Name	Certs	Links
Felix Alcala	• Offensive Security Web Expert (OSWE) • Web Application Penetration Tester eXtreme eLearnSecurityeLearn Security • Certified Red Team Operator (CRTO) Zero-Point Security Ltd • Offensive Security Certified Professional (OSCP) • Burp Suite Certified Practitioner (BSCP) PortSwigger • eLearnSecurity Certified Professional Penetration Tester (eCPPT) • eLearnSecurity Certified Penetration Tester Extreme (eCPTX) • eLearnSecurityeLearn Security • Offensive Security Experienced Penetration Tester (OSEP)	• https://www.credential.net/9bb95c71-d7d5-4342-925f-90b8b6d9f9ae#gs.32s1kb • https://www.credential.net/4fb0f7b4-8709-48e7-8c99-e2a915fb1656#gs.32s35t • https://www.credential.net/23d96f16-4576-4de2-b5e9-eef191cb8d65 • https://api.eu.badgr.io/public/assertions/6c-k8UHXQQSvKh9UOgs9zQ?identity__email=felix.alcala%400xdeadbeef.ai • https://portswigger.net/web-security/e/c/cb3facb82bd9160e • https://verified.elearnsecurity.com/certificates/3033afc5-f6a4-48a3-b834-657ab608c6ef • https://www.elearnsecurity.com/certification/verify?c=d633c8d3-20da-4693-b7f8-2588128c1305 • https://www.elearnsecurity.com/certification/verify?c=18a72f42-c858-4248-ab5a-561b20f4964d