How to Secure Your Software Supply Chain and Speed-Up DFIR with Hashlookup

the harsh reality of the software supply chain

CIRCL
Computer Incident
Response Center
Luxembourg

Alexandre Dulaunoy
Jean-Louis Huynen
TLP:CLEAR

info@circl.lu / hashlookup.io

Pass the SALT - Lille - 4th July 2023

CIRCL and Open Source Tooling

To assist us carry out these missions, having **efficient tools is critical**:

- From use-cases to **tool development**[1].
- **Open source** tools [2].
- Associated **services**[3] are available.
- **Producing intelligence** from and for the available services.
- In 2023, CIRCL maintain more than 12 open source projects[4] (250+ official git repositories).

[1] *Eating your own dog food*
[2] *Public Money, Public Code*
[3] publicly accessible or restricted access services
[4] `https://opensource-metrics.circl.lu/`

MISP Project and CIRCL

- CIRCL **leads the development** of the Open Source MISP threat intelligence platform[5] which is used by many military or intelligence communities,private companies, financial sector, National CERTs and LEAs globally.
- **CIRCL runs multiple large MISP communities performing active daily threat-intelligence sharing**.
- Private sector such as the financial sector can request access to one or more information sharing communities operated by CIRCL.

[5] https://www.misp-project.org/

(Supply Chain) Attacks and our Open Source Software

- Having experience as both **incident responders** and **open source maintainers** gives us a comprehensive perspective.
- We are implementing the best practices recommended by CERTs/CSIRTs to ensure effective incident response.
- But this perspective has interesting implications on how we see supply chain attacks.

Security Vulnerability Reporting

MISP disclosure page

We firmly believe that, even though unfortunately it is often not regarded as common practice in our industry, being as transparent as possible about vulnerabilities, no matter how minor, is of crucal importance. At MISP Project, we care about the security of our users and **prefer to have a high number of published CVEs** rather than sweeping some of them under the rug.[a]

[a] `https://www.misp-project.org/security/`

Software Dependencies

Software Dependencies Mantra

Less dependencies is obviously better but our strategies are defensive:

- Can we **take over the dependency** if the upstream maintainer is giving up?
- Is the upstream maintainer **open to vulnerability disclosure**?
- How are new changes incorporated and "controlled" from upstream maintainer?
- Can we lock validated version outside package management?
- Can **review all the origin/integrity files** delivered by the upstream maintainer in the case of an incident?

ATT&CK Technique: Supply Chain Compromise (T1195)

- *Adversaries may manipulate products or product delivery mechanisms prior to receipt by a final consumer for the purpose of data or system compromise.*
- **Use verification of distributed binaries through hash checking**. But is this easy? Where can you find those hashes?

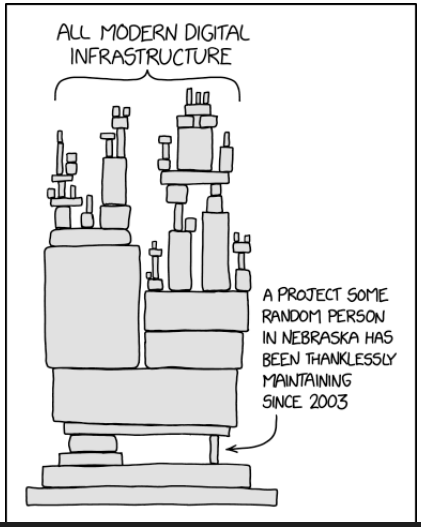
Mitigations

ID	Mitigation	Description
M1051	Update Software	A patch management process should be implemented to check unused dependencies, unmaintained and/or previously vulnerable dependencies, unnecessary features, comp
M1016	Vulnerability Scanning	Continuous monitoring of vulnerability sources and the use of automatic and manual code review tools should also be implemented as well.[9]

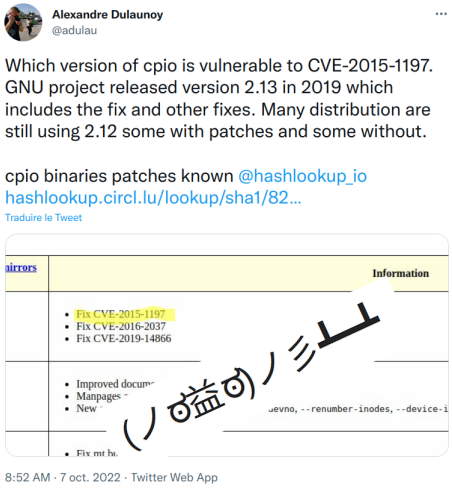
Detection

Use verification of distributed binaries through hash checking or other integrity checking mechanisms. Scan downloads for malicious signatures and attempt to test software and updates prior to deployment while t
Perform physical inspection of hardware to look for potential tampering.

Do you know about this little binary used everywhere?



Do you know about this little binary used everywhere?



US - Executive Order 14028 of May 12, 2021

> (vi) maintaining accurate and up-to-date data, provenance (*i.e.*, origin) of software code or components, and controls on internal and third-party software components, tools, and services present in software development processes, and performing audits and enforcement of these controls on a recurring basis;
>
> (vii) providing a purchaser a Software Bill of Materials (SBOM) for each product directly or by publishing it on a public website;

[6]

- SolarWinds was just a trigger,
- Havex (ICS distribution), Kingslayer (repackaging signed binaries), CCleaner (build environment), NetSarang (Backdooring a Windows Updater), ASUS (custom updater), software repositories (npm, PyPI)...

[6] `https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity`

Starting digital forensic investigation on a recent acquisition

- A single disk acquisition of a desktop or server operating system contains at least 150K files,
- Large portion of directories and files are not analysed due to a **lack of time**,
- Finding legitimate versus attacker-installed files can be difficult if the timeline is incorrect,
- Many legacy tools are used by attackers and mixed with custom binaries.

Known file filters - DFIR issues

- **State of current NIST NSRL**[7] databases and other known file filters (KFF)
- A lack of Operating Systems / Software available (e.g. OSX?, Linux distributions)
- nsrllookup.com / nsrlsrv use their own protocol, no ReST API
- nsrlsrv[8] only supports MD5s
- Many **sources are difficult to use** (e.g. NSRL ISOs/SQLite), **ill-maintained**, **outdated** or **expensive**,
- MISP integration (malicious hashes versus known hashes).

[7] `https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsrl`

[8] `https://rjhansen.github.io/nsrlsvr/`

Indexing all published software?

- **Regular updates of Linux distributions** including security updates on multiple architectures,
- 800+ software releases per hour on GitHub
- Bundling of software in **snap** images, **flatpak**, **AppImage**, etc.
- **Continuous release** of security updates
- Microsoft Windows and Apple custom software distribution schemes.

Known file filters - improvements required

- A need for a **public, open and easy** to use API for all sources (NSRL is not alone)
- A **global, public instance of all known sources**,
- A common ReST API normalises the access to several datasources
- Available for MD5, and SHA1 (and more)
- Includes fuzzy hashes
- Includes additional datapoints available by **combining a set of datasources**

CIRCL hashlookup public service

- https://hashlookup.circl.lu/[9] - **OpenAPI** Swagger[10]
- NIST NSRL - **all RDS hash sets** including current, modern, android, iOS and legacy sets
- Ubuntu package distribution
- CentOS core OS distribution
- Fedora project EPEL repository
- CDNjs repository (**JavaScript are criticials in DFIR)**
- **Kali linux package distribution, OpenSUSE distribution and more**
- **If you find it in a lot of trusted places, you may find that it's reasonable to trust it.**

[9] `https://hashlookup.circl.lu/`

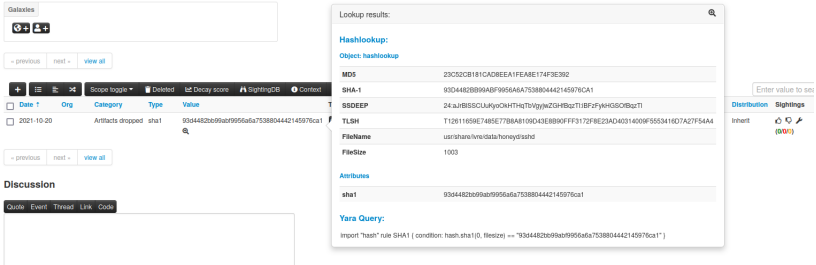
[10] `https://hashlookup.circl.lu/swagger.json`

hashlookup.circl.lu API example

```
adulau@maurer:~$ curl -s https://hashlookup.circl.lu/lookup/sha1/732458574c63c3790cad093a36eadfb990d11ee6 | jq .
{
  "FileName": "./bin/ls",
  "FileSize": "142144",
  "MD5": "E7793F15C2FF7E747B4BC7079F5CD4F7",
  "SHA-1": "732458574C63C3790CAD093A36EADFB990D11EE6",
  "SHA-256": "1E39354A6E481DAC48375BFEBB126FD96AED4E23BAB3C53ED6ECF1C5E4D5736D",
  "SHA-512": "233382698C722F0AF209865F7E998BC5A0A957CAB389E8A84BA4172F2413BEA1889DD79B12607D9577FD2FC17F300C8E7F2
  "SSDEEP": "1536:BgfDyKo9d0mLrTpjQ2xioEbuGMC0kDLmLUFqpfgBLO+qDutbxHFb65RRnSULS0pF:BADnGd0mxst7DLmg0OBLIupbn0pJqN
  "TLSH": "T178D32C07F15308BCC5D1C071865B9262BA31BC599332263F3A8CF6791F66F795B7AA20",
  "insert-timestamp": "1655501032.5410244",
  "mimetype": "application/x-sharedlib",
  "source": "snap:uycWNqU7Kjtw6mXXJrSxh6jCDdHvEjVt_21",
  "hashlookup:parent-total": 45,
  "parents": [
    {
      "SHA-1": "00363CBD7E44AA37137E8A6E797507704EF111AC",
      "snap-authority": "canonical",
      "snap-filename": "BC52ksa3GpCgET5MpLjg1WtmtpKvwI6c_11.snap",
      "snap-id": "BC52ksa3GpCgET5MpLjg1WtmtpKvwI6c_11",
      "snap-name": "qt5-core20",
      "snap-publisher-id": "ccpcJpODSdWMi621YDqnMi9Q8UO6hb8L",
      "snap-signkey": "BWDEoaqyr25nF5SNCvEv2v7QnM9QsfCc0PBMYD_i2NGSQ32EF2d4D0hqUel3m8ul",
      "snap-timestamp": "2022-02-17T20:28:04.914700Z",
      "source-url": "https://api.snapcraft.io/api/v1/snaps/download/BC52ksa3GpCgET5MpLjg1WtmtpKvwI6c_11.snap"
    },
    {
      "SHA-1": "0844D3CB657F353AB2CE1DB164CE6BDFFD2BB6FD",
      "snap-authority": "canonical",
      "snap-filename": "8BtI009xODljWTvzy37M55T8ZQiOiVft_3.snap",
      "snap-id": "8BtI009xODljWTvzy37M55T8ZQiOiVft_3",
      "snap-name": "osreport",
      "snap-publisher-id": "Yrin91Qs2D8dW9QVSQgQg9VxaGkpfQsr",
      "snap-signkey": "BWDEoaqyr25nF5SNCvEv2v7QnM9QsfCc0PBMYD_i2NGSQ32EF2d4D0hqUel3m8ul",
      "snap-timestamp": "2021-05-11T18:56:58.598072Z",
      "source-url": "https://api.snapcraft.io/api/v1/snaps/download/8BtI009xODljWTvzy37M55T8ZQiOiVft_3.snap"
    },
```

hashlookup MISP module

- A hover and expansion module[11] to quickly check if a hash is part of the known files of hashlookup:



[11] https://misp.github.io/misp-modules/expansion/#hashlookup

hashlookup MISP module - import

1)

hashlookup - offline lookup with Bloom filters

- DFIR requires **fast-lookup** and **offline** (for privacy and confidentiality reasons).
- hashlookup provides a weekly Bloom filter dump[12] for this purpose (see rationale here[13]),
- Bloom filter can be loaded in tools such as hashlookup-forensic-analyser[14], hashlookup-gui[15], and many others.

[12] `https://cra.circl.lu/hashlookup/hashlookup-full.bloom`
[13] `https://tinyurl.com/hashlookup-bloom`
[14] `https://www.github.com/hashlookup/hashlookup-forensic-analyser`
[15] `https://www.github.com/hashlookup/hashlookup-gui`

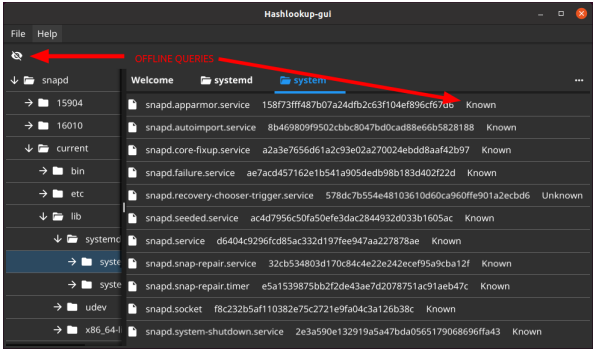
hashlookup-forensic-analyser

- Analyse[16] **a forensic target** to find and report files, which were found or not found, from the hashlookup public service or the Bloom filter from CIRCL's hashlookup.
- Lookup **live processes** on Linux (using /proc) to discover unknown processes.
- Generate machine-readable reports for forensic triage.

[16] `https://github.com/hashlookup/hashlookup-forensic-analyser`

hashlookup-gui - offline lookups with Bloom filters

- hashlookup-gui[17] a multi-platform Graphical User Interface for querying hashlookup services.



[17] https://github.com/hashlookup/hashlookup-gui

Some experiments around yara

- A **Bloom filter library in C**[18] compatible with DCSO's bloom[19]
- A **yara module**[20] to query a filter directly from yara,
- Main ideas:
  - Avoid processing known files
  - Keep track of progress

```
import "araygrass"
import "hash"

rule HashlookupMatching
{
    condition:
        araygrass.check_string(hash.sha1(0, filesize), 1) == 1
}
```

[18] https://github.com/hashlookup/fleur

[19] https://github.com/DCSO/bloom

[20] https://github.com/hashlookup/a-ray-grass

What's the future for the adversaries?

- We are still at **basic supply chain attacks** compared to Ken Thompson's paper on "Reflections on Trusting Trust" [21] (1984),
- The increased sources of distribution channels (software repackaged in packages - **hiding the mess**)
- SolarWinds attacks are just **the tip of iceberg** when it comes to the security state of the software supply chain
- Software reuse is finally here but the risks of libraries dependencies are increasing.

[21] https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf

What can I do?

- Require your supplier to provide a **software bill of materials (SBOM)** for each software release
- **Exercise your incident response procedure** and most importantly review your capability to baseline the origin of the software installed
- **Verify the claims** of your software vendors/suppliers (e.g. zero dependencies)
- Acquire internal capabilities to **verify software release integrity**

hashlookup.io future

- **Additional sources** of software publishers will be added on a regular basis
- Improving Bloom filters per type and categories of software
- Having Bloom filters on beginning of files
- Investigate having false negatives instead of false positives (e.g. LRU caches)
- Add an **API for known software publishers** to submit their hashes into hashlookup
- It's an open source project, so feel free to **contribute**

Conclusion

- One of the advantages of open-source is that all the data is publicly available, but **its analysis is not widely distributed**, which could leave it vulnerable to potential supply chain attacks.
- It's important to keep in mind that while **SBOM (Software Bill of Materials) is a significant step forward, it is not a cure-all solution** for supply chain security concerns.
- Complexity is still a major positive factor for successful supply chain attacks (e.g. complex pipelines of deployment).

Contact

- `https://hashlookup.io/`
- `https://circl.lu/services/hashlookup/`
- Twitter: @adulau @gallypette @circl_lu
- More free: @adulau@infosec.exchange @gallypette@infosec.exchange @circl@social.circl.lu

What's up with Bloom filters? and API lookup?