in-toto -- Securing the whole software supply chain

Santiago Torres-Arias, Hammad Afzali, Lukas Pühringer, Reza Curtmola, Justin Cappos
How is software made?
A stylized software supply chain
Attackers can hack the software supply chain
Attackers do hack the software supply chain

Linux Source Code Repository Kernel.Org Gets Hacked

By Brian Prince on September 01, 2011

A number of servers belonging to kernel.org were compromised last month in an attack that may have started with a stolen user credential.

According to a statement on kernel.org, which hosts the source code for the Linux kernel, the attack is not believed to have affected the source code repositories. While the situation remains under investigation, it is believed the attackers gained access to a server known as ‘Hera.’
Attackers do hack the software supply chain

IMPORTANT JUNIPER SECURITY ANNOUNCEMENT

CUSTOMER UPDATE: DECEMBER 20, 2015

Administrative Access (CVE-2015-7755) only affects ScreenOS 6.3.0r17 through 6.3.0r20. VPN Decryption (CVE-2015-7756) only affects ScreenOS 6.2.0r15 through 6.2.0r18 and 6.3.0r12 through 6.3.0r20.

We strongly recommend that all customers update their systems and apply these patched releases with the highest priority.

POSTED BY BOB WORRALL, SVP CHIEF INFORMATION OFFICER ON DECEMBER 17, 2015

Juniper is committed to maintaining the integrity and security of our products and wanted to make customers aware of critical patched releases we are issuing today to address vulnerabilities in devices running ScreenOS® software.

During a recent internal code review, Juniper discovered unauthorized code in ScreenOS that could allow a knowledgeable attacker to gain administrative access to NetScreen® devices and to steal sensitive information.
Attacked do hack the software supply chain

IMPORTANT JUNIPER SECURITY ANNOUNCEMENT

CUSTOMER UPDATE: DECEMBER 20, 2015

Administrative Access (CVE-2015-7755) only affects ScreenOS 6.3.0r20.

We strongly recommend that all customers update all devices running ScreenOS® software.

POSTED BY BOB WORRAI, SVP CHIEF INFORM

During a recent internal code review, Juniper discovered a vulnerability that could allow a knowledgeable attacker to gain administrative access to devices running ScreenOS® software.

phpMyAdmin corrupted copy on Korean mirror server

On September 25th, SourceForge became aware of a corrupted copy of phpMyAdmin being served from the 'cdnetworks-kr-1' mirror in Korea. This mirror was immediately removed from rotation.

The mirror provider has confirmed the attack vector has been identified and is limited to their mirror, with exploit having occurred on or around September 22nd.

Through validation we have confirmed the corrupted file (a modified copy of phpMyAdmin-3.5.2.2-all-languages.zip) was served only via the 'cdnetworks-kr-1' mirror.
Attacker do hack the software supply chain

IMPORTANT JUNIPER SECURITY ANNOUNCEMENT

CUSTOMER UPDATE: DECEMBER 20, 2015

Administrative Access (CVE-2015-7755) only affects ScreenOS 7.0.x. Key Decryption (CVE-2015-7756) only affects ScreenOS 6.3.0r20.

We strongly recommend that all customers update the software with the highest priority.

POSTED BY BOB WORRALL, SVP CHIEF INFORMATION SECURITY OFFICER

Juniper is committed to maintaining the integrity of the software. We are notifying our customers and are working to update them with critical patch releases as they are completed. We encourage our customers to maintain the most up-to-date version of their software.

During a recent internal code review, Juniper discovered vulnerabilities that could allow a knowledgeable attacker to gain administrative access.

Apple cleans up iOS App Store after first big malware attack

XcodeGhost dispels comforting notion that iOS is safe from malware
Attacker's do hack the software supply chain

Hacker explains how he put "backdoor" in hundreds of Linux Mint downloads

The hacker said their prime motivation for the backdoor was to build a botnet.

How can we fix this?
Many good point solutions
Many good point solutions

→ git signing, reference state log [Torres USENIX Sec 16], …
Many good point solutions

→ git signing, reference state log [Torres USENIX Sec 16], ...

→ TPMs, HSMs, reproducible builds, ...
Many good point solutions

→ git signing, reference state log [Torres USENIX Sec 16], ...

→ TPMs, HSMs, reproducible builds, ...

→ TLS, GPG, TUF
Fixed?
Gaps between steps? Compliance?
We want to secure the complete Software Supply Chain!

→ Verifiably define the steps of the software supply chain
→ Verifiably define the authorized actors
→ Guarantee that everything happens according to definition, and nothing else
in-toto -- Project Definition -- Steps

```json
{
   "_type": "layout",
   "expires": "2017-08-31T12:44:15Z",
   "keys": {
      "0c6c50": { ... }
   },
   "signatures": [...],
   "steps": [
      {
         "_type": "step",
         "name": "checkout-code",
         "expected_command": ["git", "clone", "..."],
         "expected_materials": [],
         "expected_products": [
            ["CREATE", "demo-project/foo.py"], ...],
         "pubkeys": ["0c6c50..."],
         "threshold": 1
      }, ...],
   "inspections": [...]
}
```
in-toto -- Project Definition -- Functionaries

Bob

Carol

Dave

Erin

{ "_type": "layout", "expires": "2017-08-31T12:44:15Z", "keys": { "0c6c50": { ... } }, "signatures": [ ... ], "steps": [{ "_type": "step", "name": "checkout-code", "expected_command": ["git", "clone", "..."], "expected_materials": [], "expected_products": [{ "CREATE": "demo-project/foo.py"}], "pubkeys": ["0c6c50..."], "threshold": 1 }, ... ], "inspections": [ ... ] }
in-toto -- Project Definition -- Materials/Products
in-toto -- Project Definition -- Rules

```json
{
  "_type": "layout",
  "expires": "2017-08-31T12:44:15Z",
  "keys": {
    "0c6c50": {}
  },
  "signatures": [...],
  "steps": [{
    "_type": "step",
    "name": "checkout-code",
    "expected_command": ["git", "clone", "..."],
    "expected_materials": [],
    "expected_products": [
      ["CREATE", "demo-project/foo.py"], ...
    ],
    "pubkeys": ["0c6c50..."],
    "threshold": 1
  }, ...],
  "inspections": [...]}
}```
in-toto -- Project Definition -- Signed

{...
  "_type": "layout",
  "expires": "2017-08-31T12:44:15Z",
  "keys": { "0c6c50": { ... } }
  "signatures": [ ... ],
  "steps": [ { "_type": "step",
    "name": "checkout-code",
    "expected_command": [ "git", "clone", "..." ],
    "expected_materials": [],
    "expected_products": [ [ "CREATE", "demo-project/foo.py" ], ... ],
    "pubkeys": [ "0c6c50..." ],
    "threshold": 1
  } ],
  "inspections": [ ... ]
}
in-toto -- Signed Evidence for each Step

$ in-toto-run -- ./do-the-supply-chain-step
DEMO: Grep -- Debian'ized & in-toto'ized

fetch (dget) → extract (dpkg-source) → modify (interactive) → build (dpkg-buildpackage)
$ in-toto-run <opts> -- dget http://cdn.debian.net/debian/pool/main/g/grep/grep_2.12-2.dsc
$ in-toto-run <opts> -- dpkg-source -x grep_2.12-2.dsc
$ cd grep-2.12
$ in-toto-record start <opts>
$ dch -i
$ vi debian/rules
$ in-toto-record stop <opts>
$ in-toto-run <opts> -- dpkg-buildpackage -us -uc

$ in-toto-verify --layout-keys <key> --layout grep_2.12-2.layout

→ goo.gl/hgPMHA (demo screencast + demo metadata)
DEMO: Grep -- Debian'ized & in-toto'ized

- **fetch** (dget)
- **extract** (dpkg-source)
- **modify** (interactive)
- **build** (dpkg-buildpackage)
Create in-toto layout and protect your software supply chain's integrity

An in-toto layout defines the steps of your software supply chain that you carry out in order to write, test, package and distribute your software.

This website helps you to create a basic layout for your software project, specifying who does what and how everything fits together, so that your clients can be sure that your software was produced exactly how you wanted it to be. Learn more about in-toto on our project website and on GitHub.

You will be asked to run a couple of in-toto commands while walking through this website, so make sure to have in-toto installed:

```
pip install git+git://github.com/in-toto/in-toto.git
```

Start

These are the commands you run as part of your software supply chain

We distinguish between modifying and non-modifying steps. Modifying steps change your software product (e.g. building steps), whereas non-modifying steps don't (e.g. testing steps).

![Diagram of in-toto workflow with commands: fetch, extract, modify, build.]

This is your Software Supply Chain Layout

Below you can download your custom in-toto layout tailored to fit your software supply chain. It is a JSON formatted text file so you can easily take a look at it and also tweak it to your heart's content using any text editor.

Download your custom in-toto Layout

Now, if you don't already have one, create a project owner key and sign your new layout. You can use in-toto's command line tools as shown in the snippet below.

```
in-toto-keygen --project-owner
in-toto-sign --key <PROJECT-OWNER> <YOUR_LAYOUT>.layout
```
Thank You! Questions?

https://in-toto.io/
jcappos@nyu.edu