

# NDC { Sydney }

# The Rise of Software Supply-Chain Attacks

## How Secure is your .NET App?

Niels Tanis

**VERACODE**

0101  
0101

0101  
0101

## Who am I?

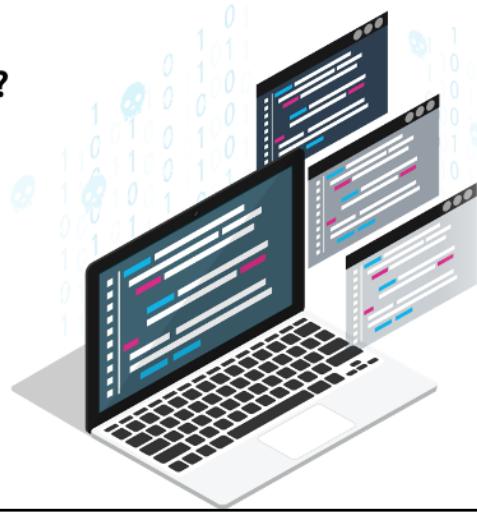
- Niels Tanis
  - Security Researcher @ Veracode
  - Background in .NET Development
  - Application Security Consultancy
  - Pen-testing & Ethical Hacking
  - ISC<sup>2</sup> CSSLP



0101  
0101

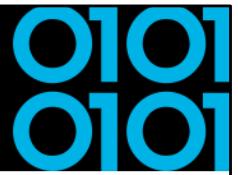
## The Rise of Software Supply-Chain Attacks

**How Secure is your .NET Application?**



Picture is from Veracode report/site:

<https://www.veracode.com/sites/default/files/pdf/resources/whitepapers/everything-you-need-to-know-about-open-source-risk/index.html>



## Agenda

- Hacker History
- Definition Software Supply-Chain
  - Development of .NET application
  - Building / Releasing / Deploying
- Securing our Software Supply-Chain
- Conclusion and Q&A

0101  
0101

## Hacking History

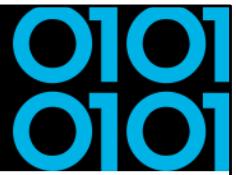
- Started out with phreaking in late '50-'60



[https://www.wikiwand.com/en/Blue\\_box](https://www.wikiwand.com/en/Blue_box)

[https://en.wikipedia.org/wiki/Kevin\\_Mitnick](https://en.wikipedia.org/wiki/Kevin_Mitnick)

<https://www.youtube.com/watch?v=8s4b2ZKyPHc>



## Getting connected!

- SATAN (Security Administrator Tool for Analyzing Networks) by Wietse Venema and Dan Farmer in 1995.
- NMAP (Network Mapper) by Gordon Lyon in 1997

```
Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-14 11:05 CET
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.16s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 996 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
25/tcp    filtered smtp
80/tcp    open     http
9929/tcp  open     nping-echo
```

[https://en.wikipedia.org/wiki/Security\\_Administrator\\_Tool\\_for\\_Analyzing\\_Networks](https://en.wikipedia.org/wiki/Security_Administrator_Tool_for_Analyzing_Networks)  
<https://nmap.org>

0101  
0101

## Smashing the Stack...

- Phrack #49 in November 1996  
**Aleph One wrote about buffer overflows**

.oo Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

File 14 of 16

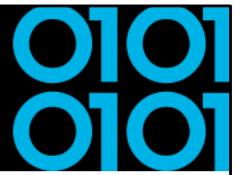
BugTraq, r00t, and Underground.Org  
bring you

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
Smashing The Stack For Fun And Profit  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

by Aleph One  
aleph1@underground.org

'smash the stack' [C programming] n. On many C implementations it is possible to corrupt the execution stack by writing past the end of an array declared auto in a routine. Code that does this is said to smash the stack, and can cause return from the routine to jump to a random address. This can produce some of the most insidious data-dependent bugs known to mankind. Variants include trash the stack, scribble the stack, mangle the stack; the term mung the stack is not used, as this is never done intentionally. See spam; see also alias bug, fandango on core, memory leak, precedence lossage, overrun screw.

<http://phrack.org/issues/49/14.html#article>



## SQL Injection

- Phrack #54 in December 1998  
**Rain Forest Puppy wrote about SQL injection**

----[ ODBC and MS SQL server 6.5

Ok, topic change again. Since we've hit on web service and database stuff, let's roll with it. Onto ODBC and MS SQL server 6.5.

I worked with a fellow W3'er on this problem. He did the good thing and told Microsoft, and their answer was, well, hilarious. According to them, what you're about to read is not a problem, so don't worry about doing anything to stop it.

- WHAT'S THE PROBLEM? MS SQL server allows batch commands.

- WHAT'S THAT MEAN? I can do something like:

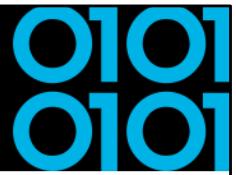
```
SELECT * FROM table WHERE x=1 SELECT * FROM table WHERE y=5
```

Exactly like that, and it'll work. It will return two record sets, with each set containing the results of the individual SELECT.

- WHAT'S THAT REALLY MEAN? People can possibly piggyback SQL commands into your statements. Let's say you have:

```
SELECT * FROM table WHERE x=%criteria from webpage user%
```

<http://www.phrack.org/issues/54/8.html>



## Code Red & SQL Slammer

- Microsoft Internet Information Server, July 2001

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801  
%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3  
%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0
```

[https://en.wikipedia.org/wiki/Code\\_Red\\_\(computer\\_worm\)](https://en.wikipedia.org/wiki/Code_Red_(computer_worm))

[https://en.wikipedia.org/wiki/SQL\\_Slammer](https://en.wikipedia.org/wiki/SQL_Slammer)

0101  
0101

## Bill Gates - Email to all MS FTE

BILL GATES BUSINESS 01.17.02 12:08 PM

### Bill Gates: Trustworthy Computing

\*This is the e-mail Bill Gates sent to every full-time employee at Microsoft, in which he describes the company's new strategy emphasizing security in its products.\*From: Bill Gates  
Sent: Tuesday, January 15, 2002 5:22 PM  
To: Microsoft and Subsidiaries: All FTE  
Subject: Trustworthy computing

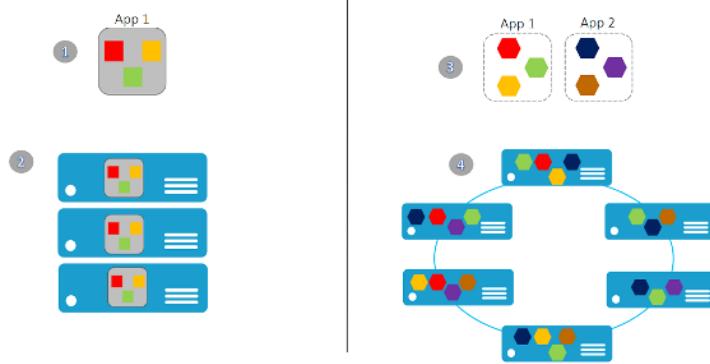
Every few years I have sent out a memo talking about the highest priority for Microsoft. Two years ago, it was the kickoff of our .NET strategy. Before that, it was several memos about the importance of the Internet to our future and the ways we could make the Internet truly useful for people. Over the last year it has become clear that

<https://www.wired.com/2002/01/bill-gates-trustworthy-computing/>

0101  
0101

## Changes in Software Architecture

- Monolith
- Microservices
- Serverless



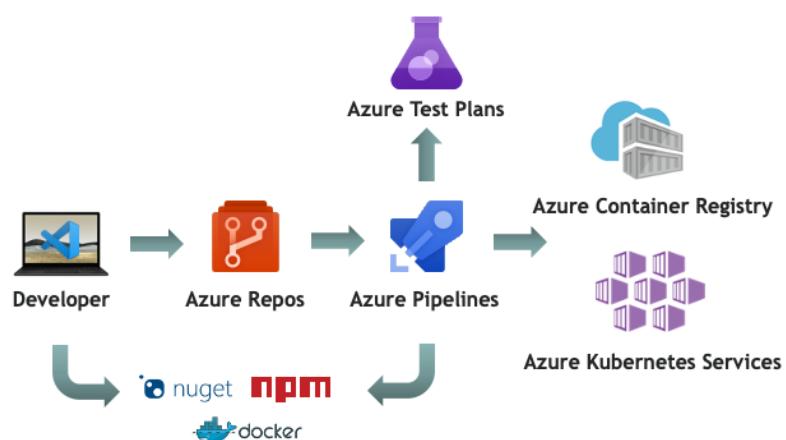
0101  
0101

## What is a Supply Chain?



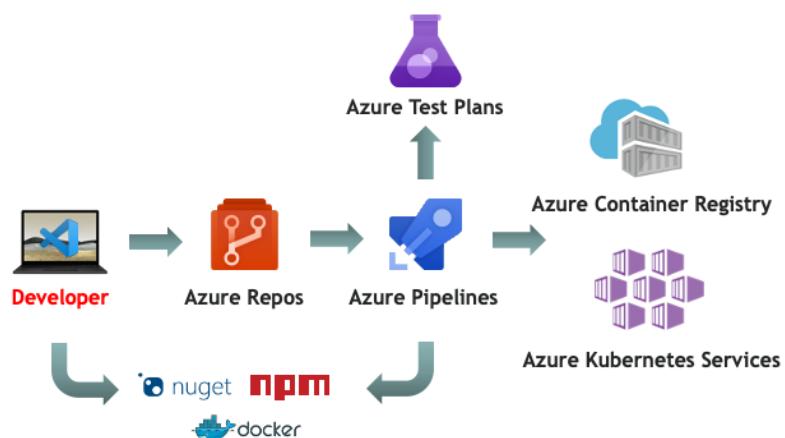
0101  
0101

## Software Supply Chain



0101  
0101

## Software Supply Chain



0101  
0101

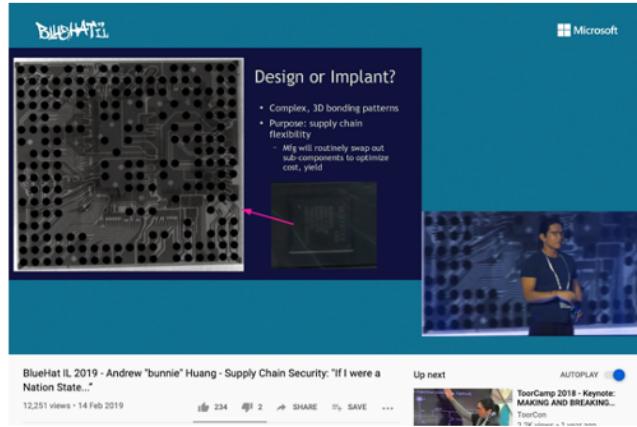
## Development Machine

- Secure Boot & Trusted Platform Module (TPM)
- Encrypt disk, harden operating system install updates
- But can you trust the hardware?

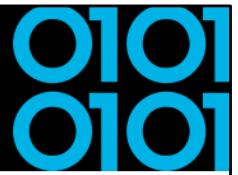


0101  
0101

# Hacking Hardware



<https://www.youtube.com/watch?v=RqQhWitJ1As>



# Development Machine

- Installs on machine - HomeBrew on Mac

README.md

## Homebrew (un)installer

### Install Homebrew

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

More installation information and options at <https://docs.brew.sh/installation.html>.



# Development Machine

- Installs on machine - Chocolatey on Windows

## Chocolatey Install:

[Individual](#)   [Organization](#)

1. First, ensure that you are using an [administrative shell](#) - you can also install as a non-admin, check out [Non-Administrative Installation](#).

2. Install with powershell.exe

**NOTE:** Please inspect <https://chocolatey.org/install.ps1> prior to running any of these scripts to ensure safety. We already know it's safe, but you should verify the security and contents of **any** script from the internet you are not familiar with. All of these scripts download a remote PowerShell script and execute it on your machine. We take security very seriously. [Learn more about our security protocols.](#)



# Octopus Scanner - NetBeans

May 28, 2020

## The Octopus Scanner Malware: Attacking the open source supply chain



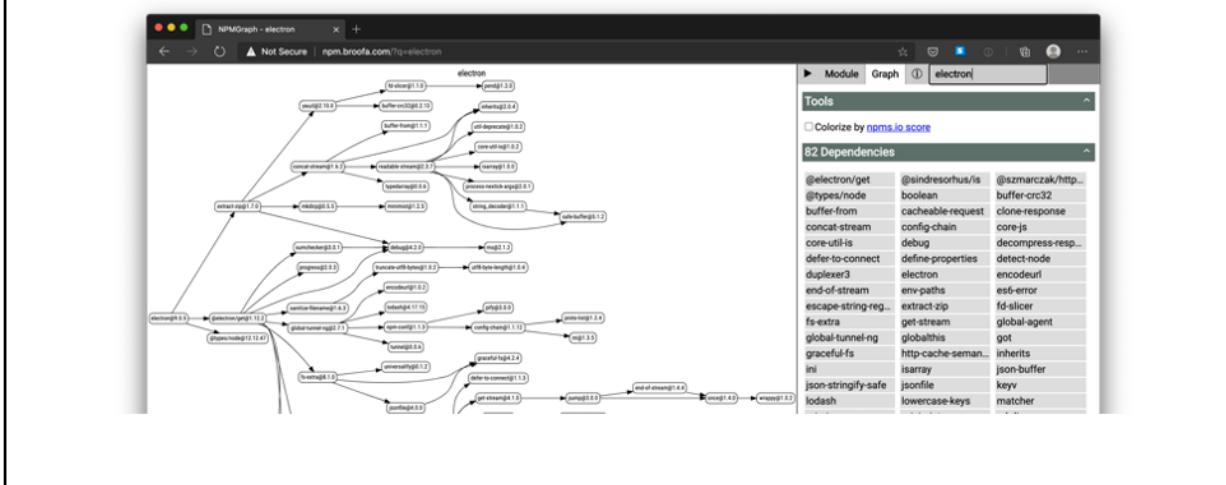
Alvaro Muñoz

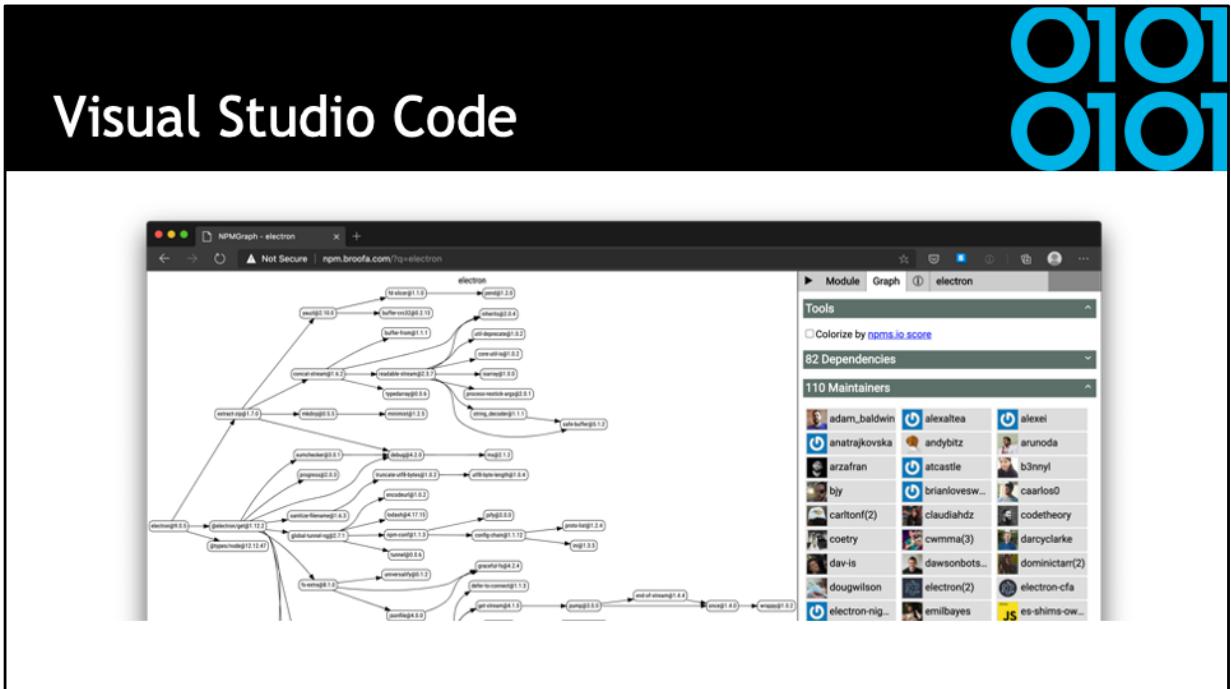
Securing the open source supply chain is an enormous task. It goes far beyond a security assessment or just patching for the latest CVEs. Supply chain security is about the integrity of the entire software development and delivery ecosystem. From the code commits themselves, to how they flow through the CI/CD pipeline, to the actual delivery of releases, there's the potential for loss of integrity and security concerns, throughout the entire lifecycle.

<https://securitylab.github.com/research/octopus-scanner-malware-open-source-supply-chain>

# Visual Studio Code

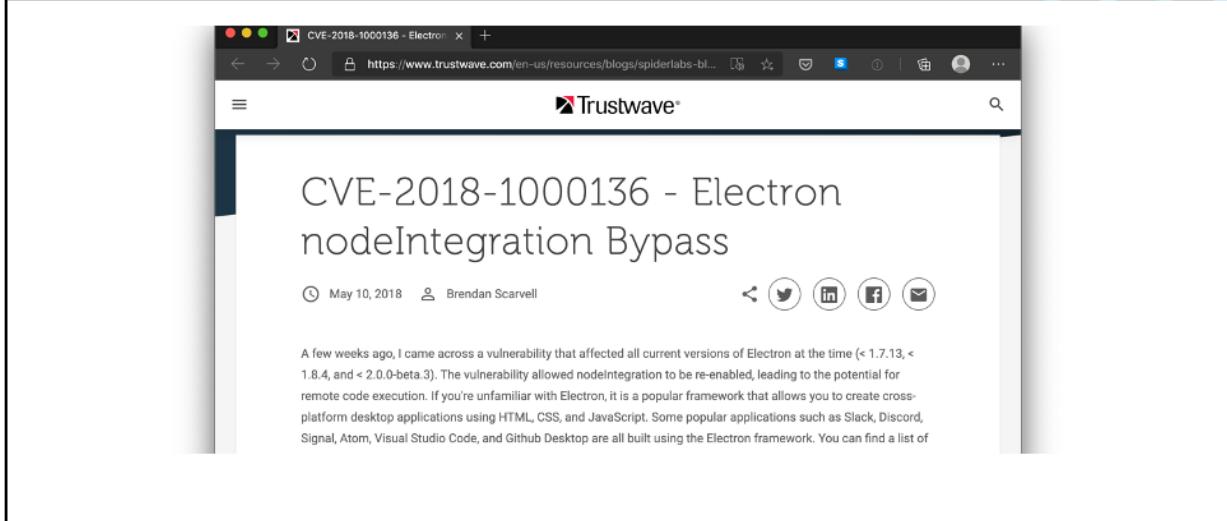
0101  
0101

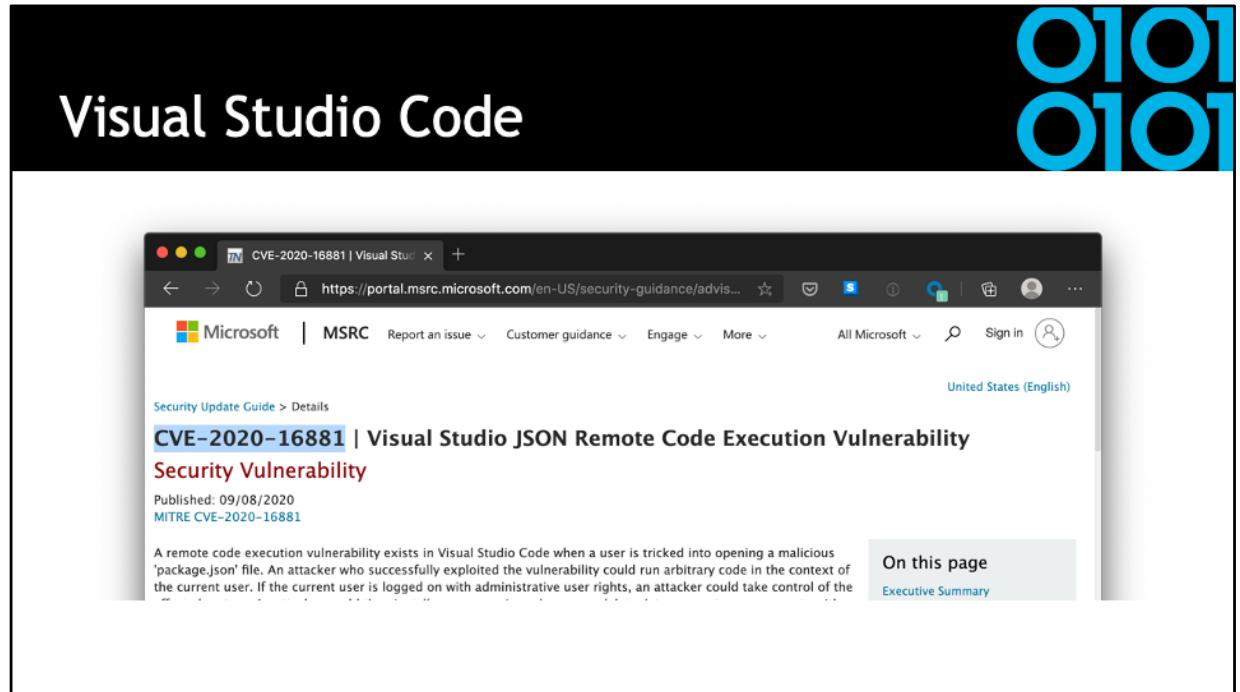


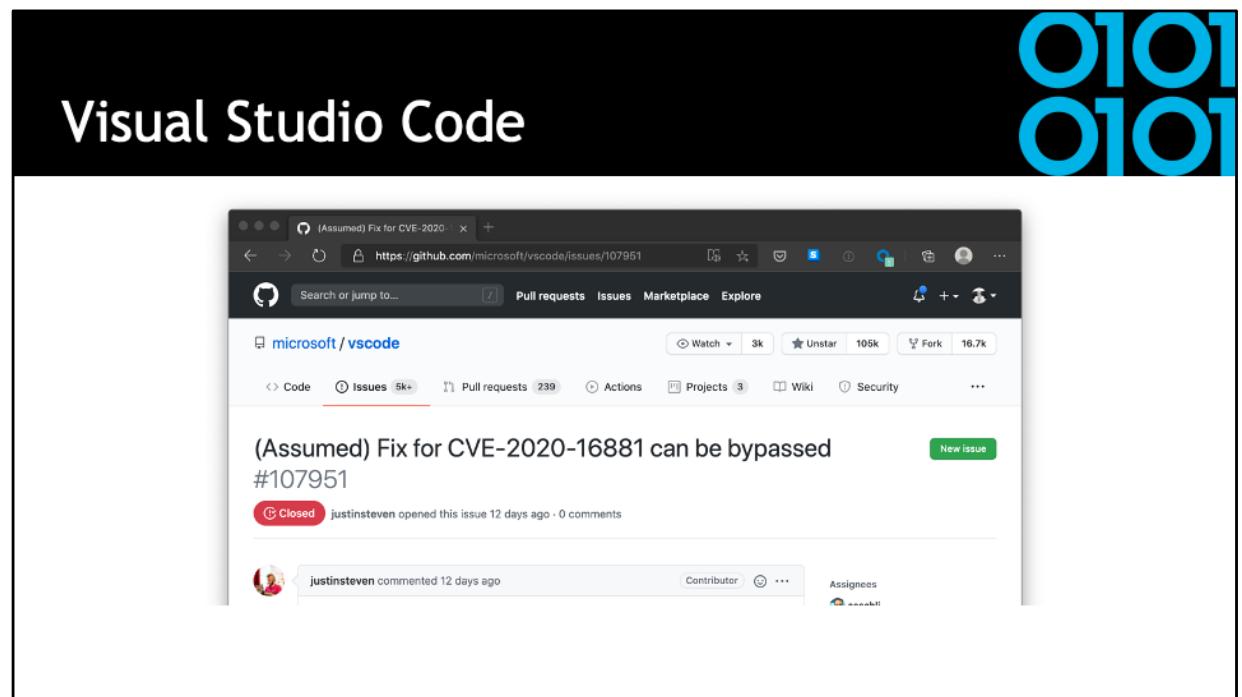


# Visual Studio Code

0101  
0101

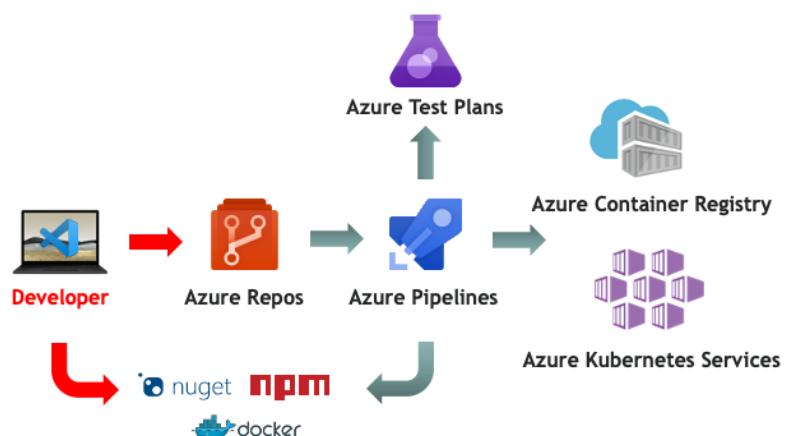






0101  
0101

## Software Supply Chain





## Development Machine

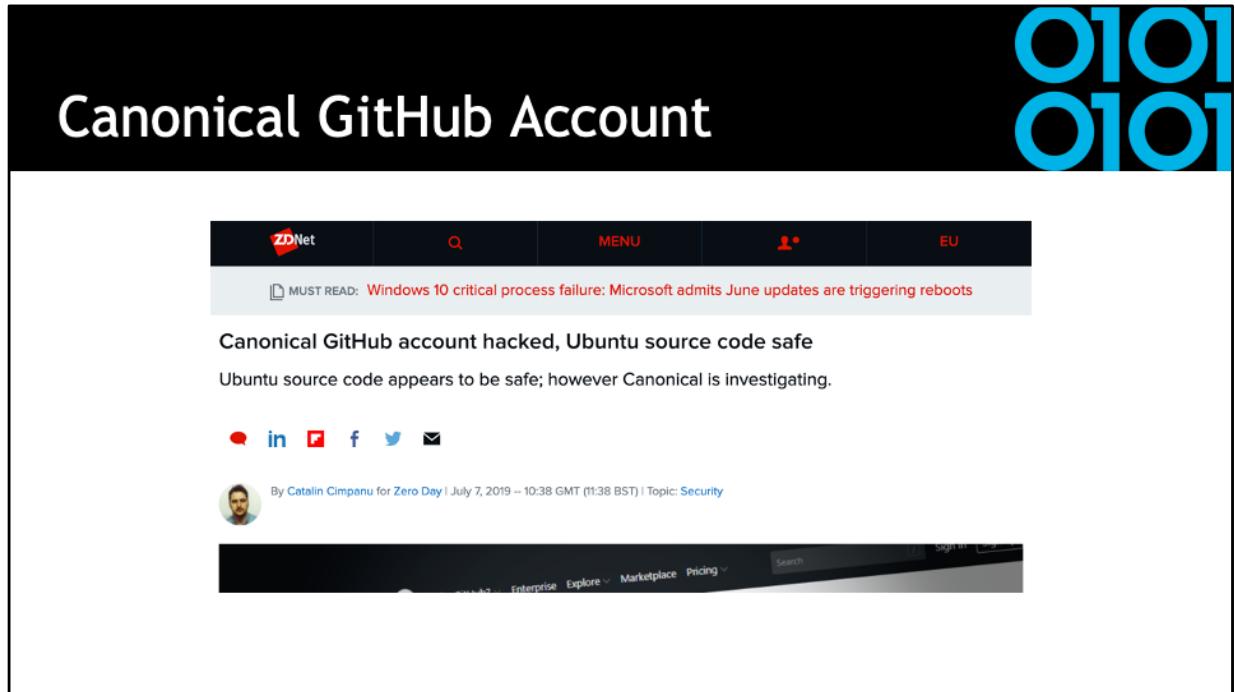
- Package manager e.g. NuGet / NPM
- Transport-Layer Security (TLS)
  - Root Authority Trust
  - Downgrade, TLS 1.0 - 1.1 deprecated on NuGet
- Domain Name Service (DNS)
  - DNSSEC → NuGet.org and GitHub.com don't support it



[https://www.ssllabs.com/downloads/SSL\\_Threat\\_Model.png](https://www.ssllabs.com/downloads/SSL_Threat_Model.png)

<https://devblogs.microsoft.com/nuget/deprecating-tls-1-0-and-1-1-on-nuget-org/>

<https://dnssec-analyzer.verisignlabs.com/nuget.org>



<https://www.zdnet.com/article/canonical-github-account-hacked-ubuntu-source-code-safe/>

The screenshot shows a news article from threatpost.com. The main title is "Report: Microsoft's GitHub Account Gets Hacked". Below the title is a small image of a computer monitor displaying a GitHub interface. To the right of the main article are three smaller news items under the heading "INFOSEC INSIDER".

**Report: Microsoft's GitHub Account Gets Hacked**

**INFOSEC INSIDER**

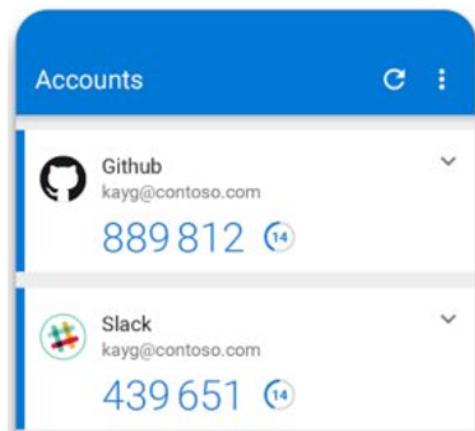
**Helping Remote Workers Overcome Remote Attacks**  June 10, 2020

**Understanding the Payload-Less Email Attacks Evasive Your Security Team**  June 4, 2020

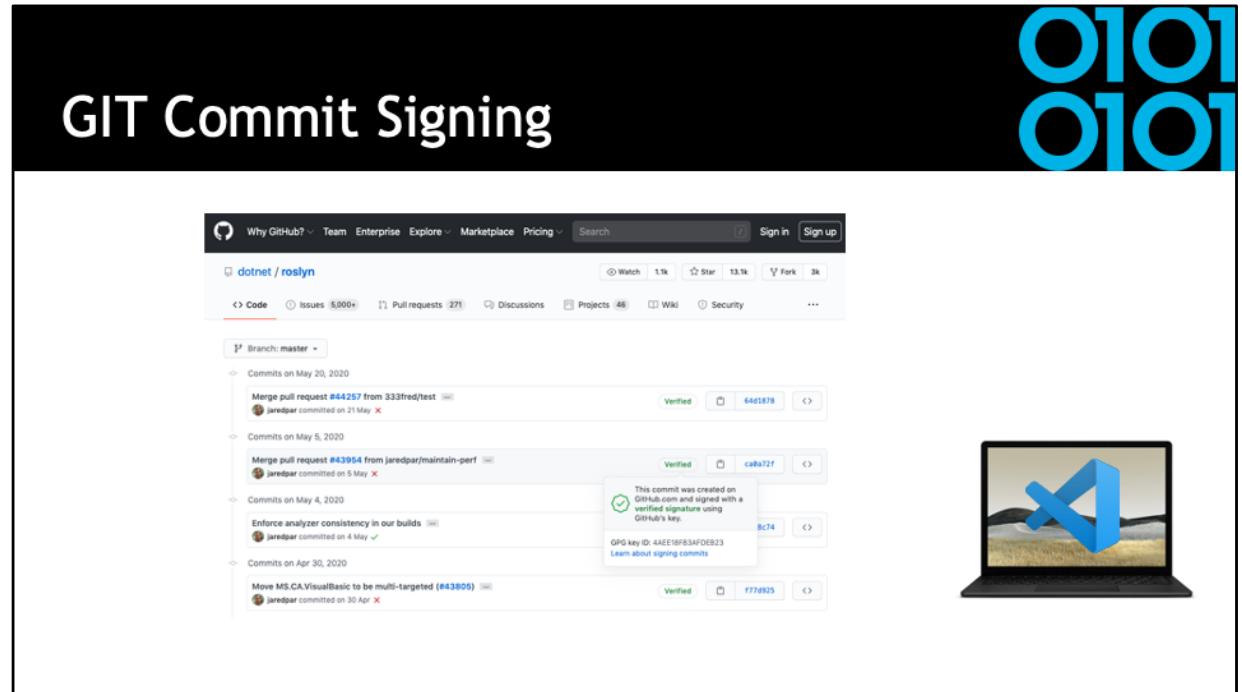
**Long Tail Analysis: A New Hope in the Cybercrime Battle** 

<https://threatpost.com/report-microsofts-github-account-gets-hacked/155587/>

## Use MFA on source-repository



<https://help.github.com/en/github/authenticating-to-github/configuring-two-factor-authentication>



<https://www.hanselman.com/blog/HowToSetupSignedGitCommitsWithAYubiKeyNEOA ndGPGAndKeybaseOnWindows.aspx>

## EvenStream NPM

0101  
0101

- November 2018
- Is transitive dependency of 2000 other libraries



Gary Bernhardt  
@garybernhardt

Follow

An NPM package with 2,000,000 weekly downloads had malicious code injected into it. No one knows what the malicious code does yet.



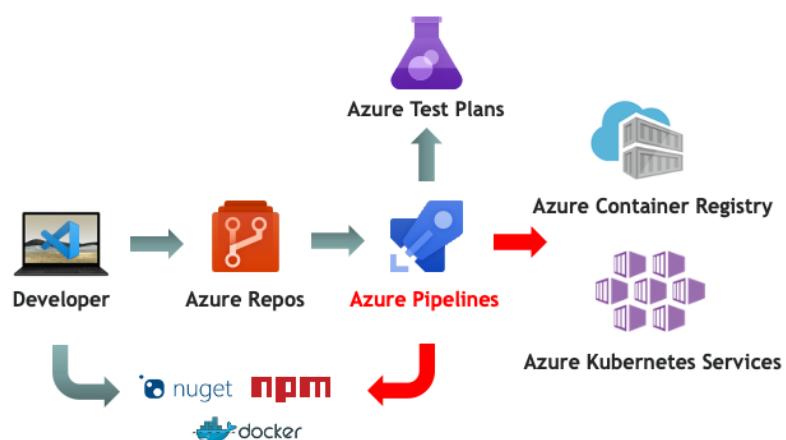
I don't know what to say. · Issue #116 · dominictarr...  
EDIT 26/11/2018: Am I affected?: If you are using anything crypto-currency related, then maybe. As discovered by @maths22, the target seems to have b...

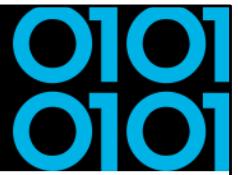
github.com

<https://twitter.com/garybernhardt/status/1067111872225136640>

0101  
0101

## Software Supply Chain



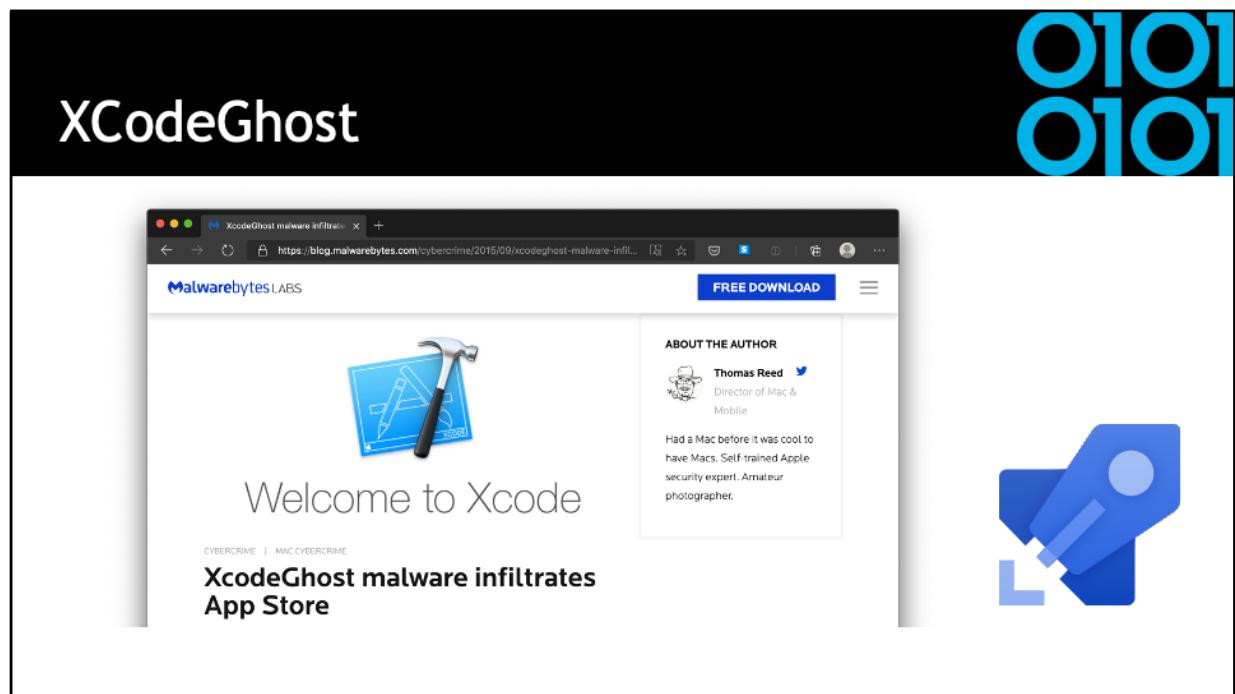


## Build / Deployment

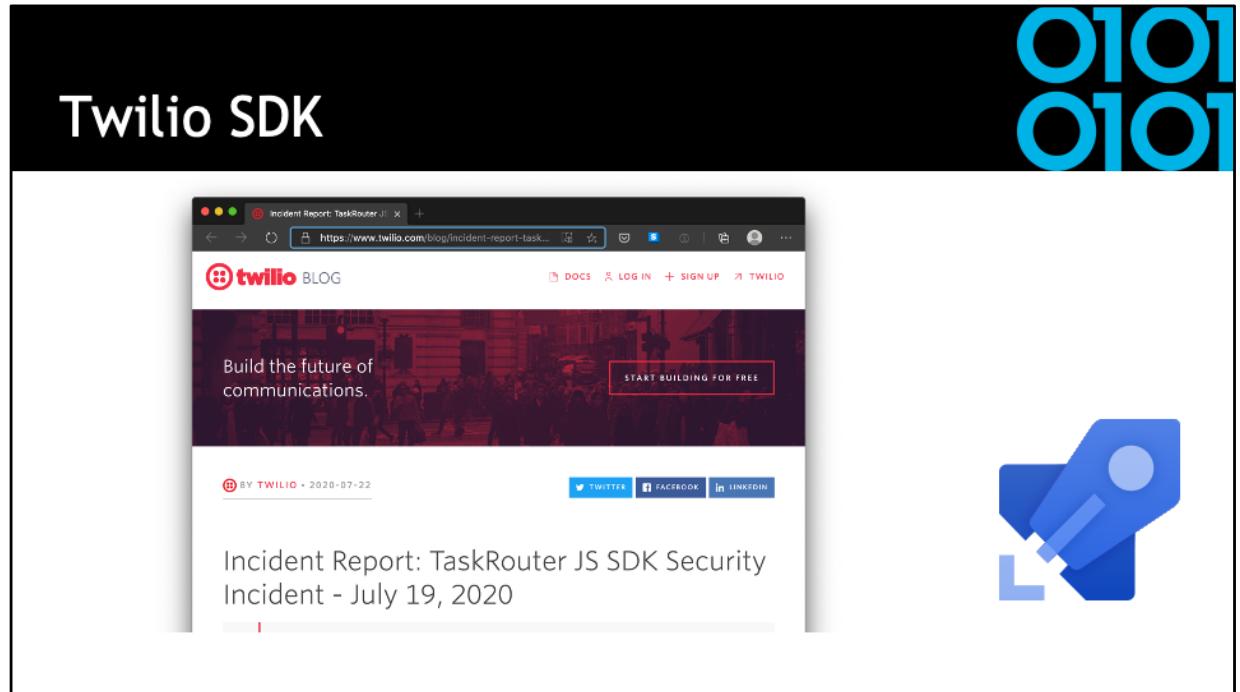
- What about hardware? Vendor trust?
- TLS issues?
- Compromised Docker Images
  - Two-Factor authentication in beta
- Build Server can be compromised



<https://docs.docker.com/docker-hub/2fa/>



<https://blog.malwarebytes.com/cybercrime/2015/09/xcodeghost-malware-infiltrates-app-store/>



<https://www.twilio.com/blog/incident-report-taskrouter-js-sdk-july-2020>

0101  
0101

# Webmin Backdoor



The screenshot shows the official Webmin website. At the top, there's a navigation bar with links for Home, Downloads, Documentation, Usermin, Virtualmin, Cloudmin, and Community. Below this, a sidebar on the left lists download options for Webmin 1.941, including RPM, Debian Package, TAR file, Solaris Package, Development Versions, and Third-Party Modules. It also contains a section for Webmin Links with links to Introduction To Webmin and Supported Systems. The main content area features a heading "Webmin 1.890 Exploit - What Happened?" followed by a detailed explanation of the vulnerability. A blue graphic of a hand holding a stylus is positioned on the right side of the main content area.

**Webmin 1.890 Exploit - What Happened?**

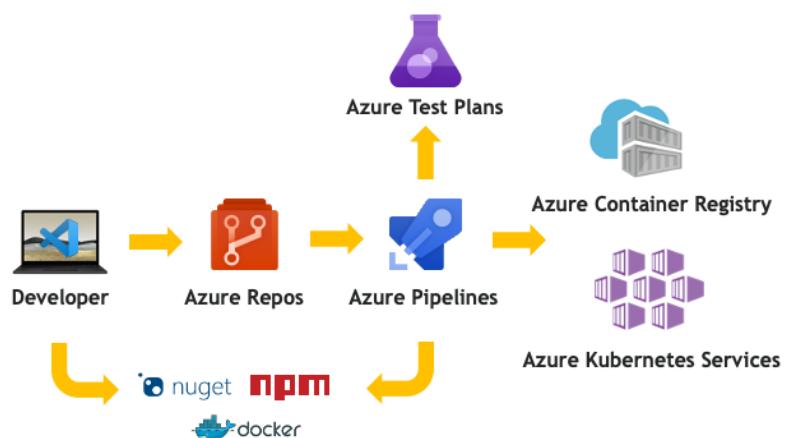
Webmin version 1.890 was released with a backdoor that could allow anyone with knowledge of it to execute commands as `root`. Versions 1.900 to 1.920 also contained a backdoor using similar code, but it was not exploitable in a default Webmin install. Only if the admin had enabled the feature at Webmin -> Webmin Configuration -> Authentication to allow changing of expired passwords could it be used by an attacker.

Neither of these were accidental bugs - rather, the Webmin source code had been maliciously modified to add a non-obvious vulnerability. It appears that this happened as follows :

- At some time in April 2018, the Webmin development build server was exploited and a vulnerability added to the `password_change.cgi` script. Because the timestamp on the file was set back, it did not show up in any Git diffs. This was included in

0101  
0101

## Software Supply Chain



# Reproducible/Deterministic Builds

0101  
0101



Home

Contribute

**Documentation**

Tools

Who is involved?

News

Events

Talks

## Definitions

### When is a build reproducible?

A build is **reproducible** if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.

The relevant attributes of the build environment, the build instructions and the source code as well as the expected reproducible artifacts are defined by the authors or distributors. The artifacts of a build are the parts of the build results that are the desired primary output.

<https://reproducible-builds.org/docs/definition/>



## Reproducible/Deterministic Builds

- Roslyn v1.1 started supporting some kind of determinism on how items are emitted
- Given same inputs, the compiled output will always be deterministic
- Inputs can be found in Roslyn compiler docs  
‘Deterministic Inputs’

<https://blog.paranoaicoding.com/2016/04/05/deterministic-builds-in-roslyn.html>

<https://github.com/dotnet/roslyn/blob/master/docs/compilers/Deterministic%20Inputs.md>

<https://github.com/clairernovotny/DeterministicBuilds>

0101  
0101

## Automotive Industry



0101  
0101

## Car Supply Chain



### Tata Steel Factory

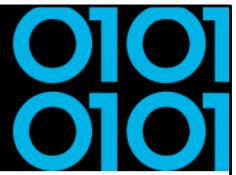
- Iron Ore from Sweden
- ISO 6892-1 Tested/Certified
  - Batch #1234

### Bosch Factory

- Steel Batch #1234 Tata
- ECE-R90 Tested/Certified
  - Serie #45678
- Used by Ford, Volkswagen and KIA

### Ford Manufacturing

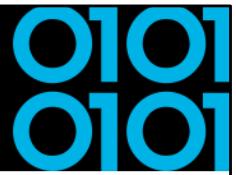
- Bosch Disk #45678
- Bosal Exhaust #RE9876
- Goodyear Tires #GY8877
- Focus VIN 1234567890



## Software Bill of Materials (SBOM)

- Industry standard of describing the software
  - Producer Identity - Who Created it?
  - Product Identity - What's the product?
  - Integrity - Is the project unaltered?
  - Licensing - How can the project be used?
  - Creation - How was the product created? Process meets requirements?
  - Materials - How was the product created? Materials/Source used?
- CycloneDX - Lightweight SBOM with dependency graph
- NTIA.org - SBOM

The image shows a screenshot of a web browser displaying the [in-toto](https://in-toto.io) website. The page has a dark header with the word "In-toto" in white. To the right of the header is a large graphic of two blue binary strings: "0101" on top and "0101" on the bottom. Below the header is a navigation bar with links for "About", "Community", "Get started", and "Learn more". The main content area features a video player for a "In-toto introduction" video from Veracode Technical Briefs. The video player shows a play button, a progress bar at 02:09/3, and a Vimeo logo. Below the video player is a diagram illustrating the software supply chain process: "code" (represented by a red icon), "test" (blue hexagon icon), "build" (cyan atom icon), and "package" (green cargo ship icon), connected by arrows. At the bottom of the slide, there is a copyright notice: "© Veracode, Inc. 2020 Confidential".



## In-Toto - Demo - Terminology

- **Functionaries** that are identified by public key our supply chain.  
Niels (Project-Owner), Aimee (Developer) and Noud (Packager)
- **Project-Owner** defines a (**Supply Chain**) **Layout** that describes **what** happens and by **who** and what the produced **Materials** and **Byproducts** are
- Link metadata is output of executed step in the **Layout**  
**Materials** are input, **Products** are output and can be used as **Materials** in later steps

<https://in-toto.io/>

# DataDog & In-Toto

The screenshot shows a web browser displaying a DataDog blog post. The title is "Secure Publication of Datadog-Agent". The main content discusses end-to-end verification using In-Toto. Below the text is a diagram illustrating the verification process across three stages: DEVELOPERS, CI/CD, and AGENT.

**DEVELOPERS:**

- TAG → YUBIKEYS

**CI/CD:**

- WHEELS-BUILDER → WHEELS-BUILDER KEY
- WHEELS-SIGNER → WHEELS-SIGNER KEY

**AGENT:**

- UNZIP

Arrows indicate the flow from left to right between stages, and red arrows point downwards from each stage to its respective key storage.

<https://www.datadoghq.com/blog/engineering/secure-publication-of-datadog-agent-integrations-with-tuf-and-in-toto/>



## Grafeas and Kritis by Google

- Grafeas - Component Metadata API
  - Container Analysis API on Google Cloud Platform
- Kritis - Deployment Authorization for Kubernetes Apps
  - Binary Authorization on Google Cloud Platform



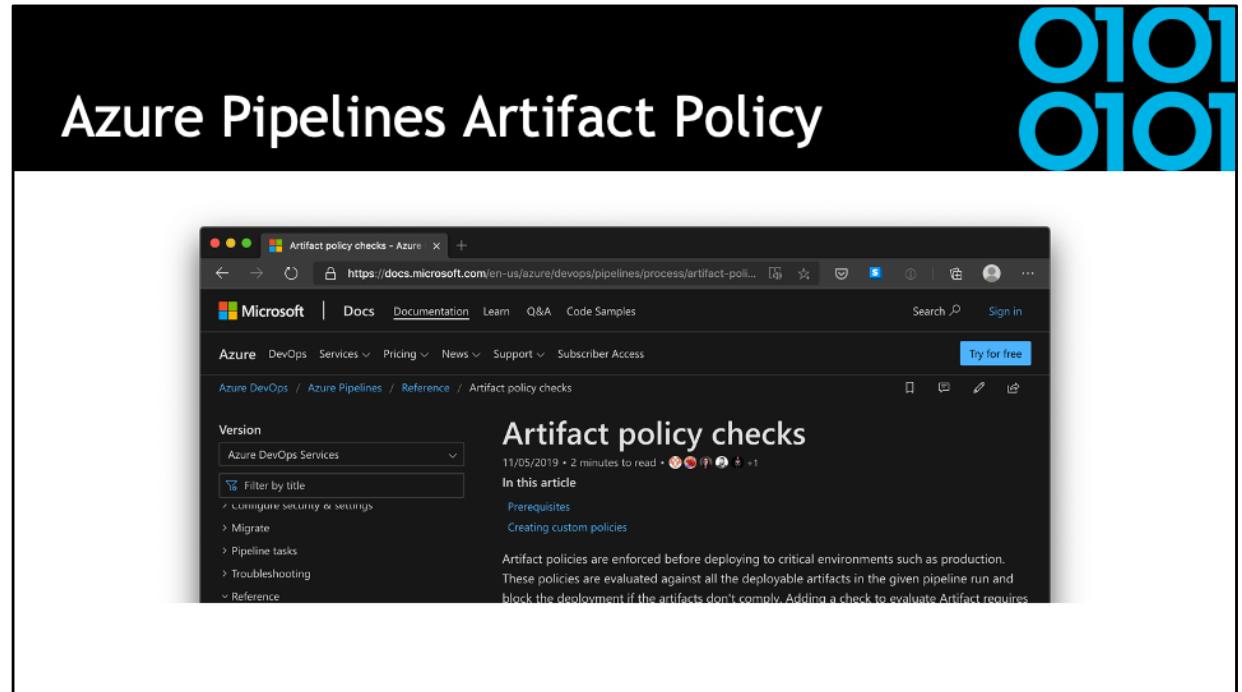
<https://grafeas.io/>

<https://github.com/grafeas/kritis/blob/master/docs/binary-authorization.md>

<https://www.infoq.com/presentations/supply-grafeas-kritis/>

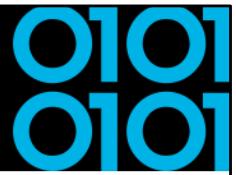
<https://www.youtube.com/watch?v=hOzH3mOApjs>

<https://www.youtube.com/watch?v=05zN-YQxEAM>



<https://devblogs.microsoft.com/devops/secure-software-supply-chain-with-azure-pipelines-artifact-policies/>

<https://docs.microsoft.com/en-us/azure/devops/pipelines/process/artifact-policy?view=azure-devops>



## Conclusion

- Be aware of your own (and other used) software supply chain(s).
- Know what you're consuming and pulling into software projects.
- Use MFA on all accounts!
- Integrate security into your software lifecycle.
- Learn more on Software Bill of Materials (SBOM).

**VERACODE**

Thanks! Questions?

<https://github.com/nielstanis/ndcsydney2020>

ntanis at veracode.com

@nielstanis on Twitter

