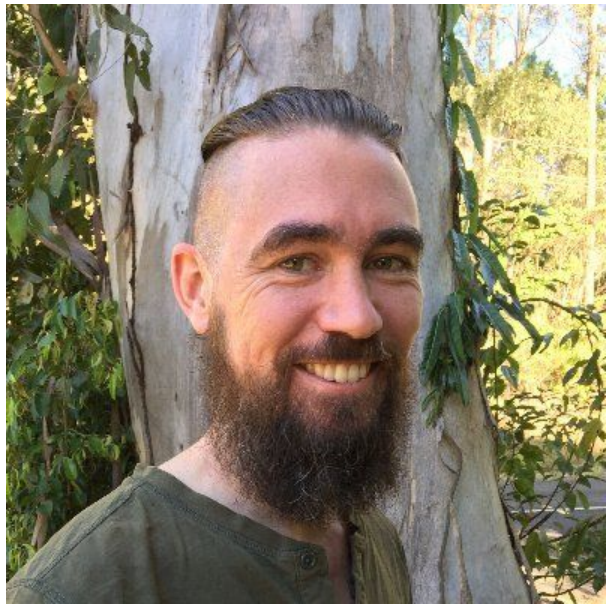
ADDO
ALL DAY DEVOPS

OCTOBER 28, 2021

Patrick Dwyer & Steve Springett

WTF is in your software?



Patrick Dwyer

@coderpatros

patrick.dwyer@owasp.org

- OWASP CycloneDX Core Working Group and Project Co-lead
- Multiple software transparency initiatives
- OSS Maintainer
- Dev Team Lead (Government)

Steve Springett

@stevespringett

steve.springett@owasp.org

- Leader of OWASP Dependency-Track
- Chair, OWASP CycloneDX Core Working Group
- Leader and co-author of OWASP SCVS
- Contributor to Package URL standard
- Multiple software transparency working groups
- Software security leadership at ServiceNow

WTF is in your software?

Package management is messy

- No standard package manifest format
- No standard version constraint format
- No standard version resolution approach
- Direct vs transitive dependencies
- Libraries and minimum version constraints

Let's start with a simple example

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
    <RuntimeIdentifier>linux-x64</RuntimeIdentifier>
    <SelfContained>True</SelfContained>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="System.Text.Json" Version="4.6.0" />
  </ItemGroup>
</Project>
```

Let's publish our app

```
$ dotnet publish
Microsoft (R) Build Engine version 17.0.0-preview-21460-01+8f208e609 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  Restored /home/user/code/PackageExample.csproj (in 48.92 sec).
  PackageExample -> /home/user/code/bin/Debug/net5.0/linux-x64/PackageExample.dll
  PackageExample -> /home/user/code/bin/Debug/net5.0/linux-x64/publish/

$ peres --file-version bin/Debug/net5.0/linux-x64/System.Text.Json.dll
File Version:                   5.0.921.35908

$ echo "WTF? That doesn't seem right."
```

Let’s check the package

```
$ wget https://api.nuget.org/v3-flatcontainer/system.text.json/4.6.0/system.text.json.4.6.0.nupkg
...
2021-10-27 14:51:40 (784 KB/s) - '4.6.0’ saved [393906/393906]

$ unzip system.text.json.4.6.0.nupkg
...

$ peres --file-version lib/netstandard2.0/System.Text.Json.dll
File Version:                   4.700.19.46214
```

Ok, let's use a package lock file

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
    <RuntimeIdentifier>linux-x64</RuntimeIdentifier>
    <SelfContained>True</SelfContained>
    <RestorePackagesWithLockFile>true</RestorePackagesWithLockFile>
    <RestoreLockedMode>true</RestoreLockedMode>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="System.Text.Json" Version="4.6.0" />
  </ItemGroup>
</Project>
```

Let's check our package lock file

```
$ dotnet restore
  Determining projects to restore...
  Restored /home/user/code/PackageExample.csproj (in 234 ms).

$ cat packages.lock.json
{
  "version": 1,
  "dependencies": {
    ".NETCoreApp,Version=v5.0": {
      "System.Text.Json": {
        "type": "Direct",
        "requested": "[4.6.0, )",
        "resolved": "4.6.0",
        "contentHash": "4F8Xe+JIkVoDJ8hDAZ7HqLkjctN/6WItJIzQaifBwClC7wmoLSda/Sv2i6i1kycqDb3hWF4JCVbpAweyOKHEUA=="
      }
    },
    ".NETCoreApp,Version=v5.0/linux-x64": {}
  }
}
```

Ok, now let's publish our app

```
$ dotnet publish
Microsoft (R) Build Engine version 17.0.0-preview-21460-01+8f208e609 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  Restored /home/user/code/PackageExample.csproj (in 48.92 sec).
  PackageExample -> /home/user/code/bin/Debug/net5.0/linux-x64/PackageExample.dll
  PackageExample -> /home/user/code/bin/Debug/net5.0/linux-x64/publish/

$ peres --file-version bin/Debug/net5.0/linux-x64/System.Text.Json.dll
File Version:                   5.0.921.35908

$ echo "Seriously? Again!"
```

WTF is happening?

Restore dependency resolution

vs

Build dependency resolution

Package version resolution

These version references all mean >= 4.6.0

- C# (NuGet) Version="4.6.0"
- Python (pip) >= 4.6.0
- Node.js (npm) "^4.6.0"

Package version resolution

With a version constraint of >= 4.6.0

- C# (NuGet) will resolve the minimum possible version
- Python (pip) will resolve the latest version
- Node.js (npm) will resolve the latest 4.6 version

What do we need? SBOM!

- Software Bill of Materials
- A nested inventory of all dependencies
- Standard, ecosystem agnostic format

Achievable use cases

- Security - lots and lots of security use cases
- Inventory of components and services
- Supply chain management
- License compliance

  … many, many more

Existing SBOM standards

CycloneDX
• Modern standard • OWASP Foundation • Security focused • Largest ecosystem of available tools

SPDX
• Older standard • Linux Foundation • License and intellectual property focused

OWASP Dependency-Track

- Consumes and analyzes SBOMs at high velocity
- Ideal for use in modern DevSecOps pipelines
- Ideal for procurement and M&A
- Identifies security, license, and operational risk
- Quickly identify if impacted, and where

https://dependencytrack.org/

OWASP SCVS

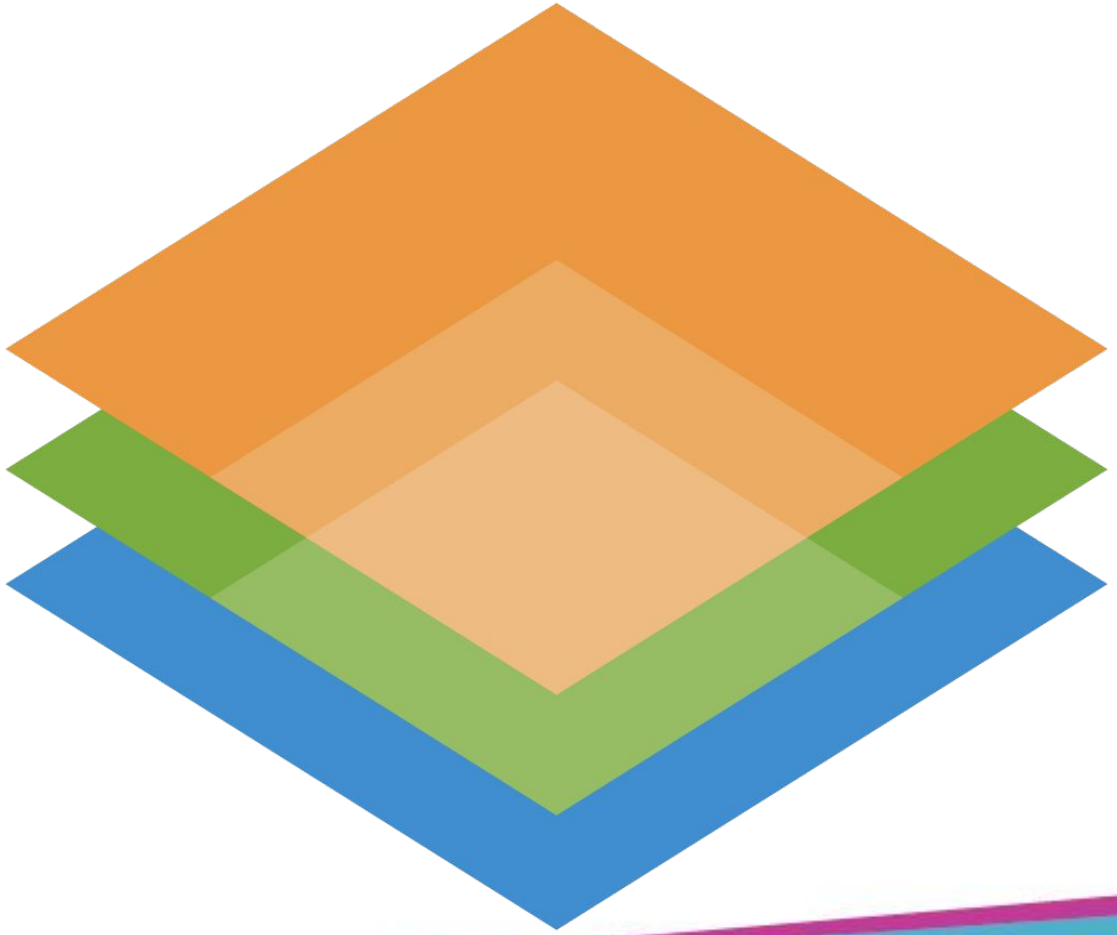
Software Component Verification Standard

- Measure and improve software supply chain assurance
- Six control families
  - Inventory
  - Software Bill of Materials (SBOM)
  - Build Environment
  - Package Management
  - Component Analysis
  - Pedigree and Provenance

https://owasp.org/scvs

OWASP SCVS

Software Component Verification Standard



Level 3 - Critical infrastructure, safety, and end-to-end software supply chain transparency

Level 2 - Regulatory/contractual requirements. Use with risk management frameworks

Level 1 - Implementation of best practices

Links to more information

OWASP CycloneDX https://cyclonedx.org/

SPDX https://spdx.dev/

OWASP Dependency-Track https://dependencytrack.org/

OWASP SCVS https://owasp-scvs.gitbook.io/scvs/