ActiveState

Software Supply Chain Security Checklist

Attackers have recently shifted their focus to infiltrating software vendor development environments. A single compromised vendor can result in thousands of vulnerable customers when they install an upgrade, patch or new version of the compromised software.

If you build software in your organization, or use a vendor's software in your development process, you'll want to assess the strength of:

- Import controls that ensure code sourced from public repositories is secure
- Build controls that ensure built artifacts are secure
- Consume controls that ensure artifacts run in your environments are secure

This datasheet provides you with a checklist of controls that can help you evaluate whether your vendors' and/or your own software supply chain truly is secure and can be trusted.

Import Process Controls

The majority of supply chain attacks are still aimed at compromising public repositories. While most organizations have no direct control over the public repositories they use, there are a number of controls that can help create a secure import process:

- [ ] Verify the identity of uploader(s)/ author(s) haven't suddenly changed
- [ ] Verify that at least two reviewers have reviewed the submission
- [ ] Verify that the timestamp of the submission is valid
- [ ] Verify the revision history. A lack of history can be indicative of typosquatted packages.
- [ ] Verify the URL/ immutable reference to counter dependency confusion*

*can occur when a build system mistakenly pulls in a similarly named dependency from a public repository rather than your private repository

Public repositories are usually very quick to resolve common issues, such as typosquatting. Simply creating a quarantine service for packages that fail the above criteria can be an effective way to deal with imported components that are suspect.

Build Process Controls

You and your vendors' dev environment are now the security frontline for your customers. Following best practices such as those listed below can help protect and secure development environments:

- [ ] Builds are scripted and have no manual inputs
- [ ] Builds are run by a dedicated service (as opposed to on a developer's workstation)
- [ ] Build steps are executed in ephemeral environments that are discarded at the end of the step
- [ ] Build steps are executed independently in isolated environments
- [ ] Build steps are executed in hermetic environments that have no public network access
- [ ] Builds are reproducible

Putting these controls into practice can thwart common attack vectors like tampered build scripts, unconstrained packages, and dynamic packages that attempt to include remote resources.

Consume Process Controls

Implementing secure import and build processes go a long way to ensuring the code you run is also secure. However, you'll still want to ensure the packages you consume are signed, and that the signature includes the following information via a cryptographic hash:

- [ ] Output artifact
- [ ] Build system used
- [ ] Source (ie., an immutable reference to the build script)
- [ ] Transitive dependencies (ie., dependencies of dependencies)
- [ ] Build parameters, if any

ActiveState Platform: Turnkey Supply Chain Security

Implementing all the controls listed above can be both time and resource intensive. The shortcut most organizations rely on is to pick a trusted vendor and exclusively use their signed packages. Unfortunately, this method can be compromised when:

- Developers install a package directly from the public repository
- Attackers compromise the development environment of trusted vendors prior to the signing service, such as happened with SolarWinds and Codecov.

Instead, consider using the ActiveState Platform to secure your Python, Perl and Tcl supply chains. It implements many of the best practices listed here in order to create verifiably reproducible builds in which the provenance (ie., the origin) can be established for each artifact.

Rather than cobbling together custom code and multiple solutions from multiple vendors, the ActiveState Platform can provide an out-of-the-box, end-to-end solution saving organizations considerable time, resources and money.

You can try the ActiveState Platform by signing up for a free account at **platform.activestate.com**

ActiveState is the de-facto standard for millions of developers around the world who have been using our commercially-backed, secure open source language distributions for over 20 years. With the ActiveState Platform, developers can now automatically build their own Python, Perl or Tcl Environments for Windows, Linux or Mac—all without requiring language or operating system expertise.

ALTAIR
NORTHROP GRUMMAN
STATE STREET
GILEAD
Honeywell
MSC Software
PEPSICO
SIEMENS
TESCO
TOYOTA

www.activestate.com
Toll-free in NA: 1-866.631.4581
solutions@activestate.com

ActiveState