How to provide security fixes in a high constraint ecosystem?

Practical examples with the Jenkins project

Wadeck Follonier

Friday 9th November 2018

What is the general context?

Jenkins and you

- Do you know Jenkins?
- Do you use Jenkins in your company?
- Do you administer an instance yourself?
- Do you apply security update when they are released?
  - Have you subscribed to the security advisory mailing list?
  - jenkinsci-advisories@googlegroups.com

Jenkins and me

- Member of the Jenkins Security team
- Security software engineer at CloudBees
- Daily tasks:
  - Providing security fixes
  - Detecting vulnerabilities and preventing them
  - Reviewing internal design / implementations
  - Improve security awareness / education
  - Helping support about security topics

The Jenkins project

- Most popular CI/CD tool on the market
- Open-source
- Over 1.7 Million users
- ~1500 plugins
- Integrated with most of the tools
  - Missing an integration? You can write your own plugin easily

Jenkins...

- Is a remote execution engine
  - build your project, execute scripts, etc.
- Has access to sensitive data
  - credentials, secrets, source code, artifacts, etc.
- Is a multi-user service
  - various kind of users with different expertise

Main security objectives (1)

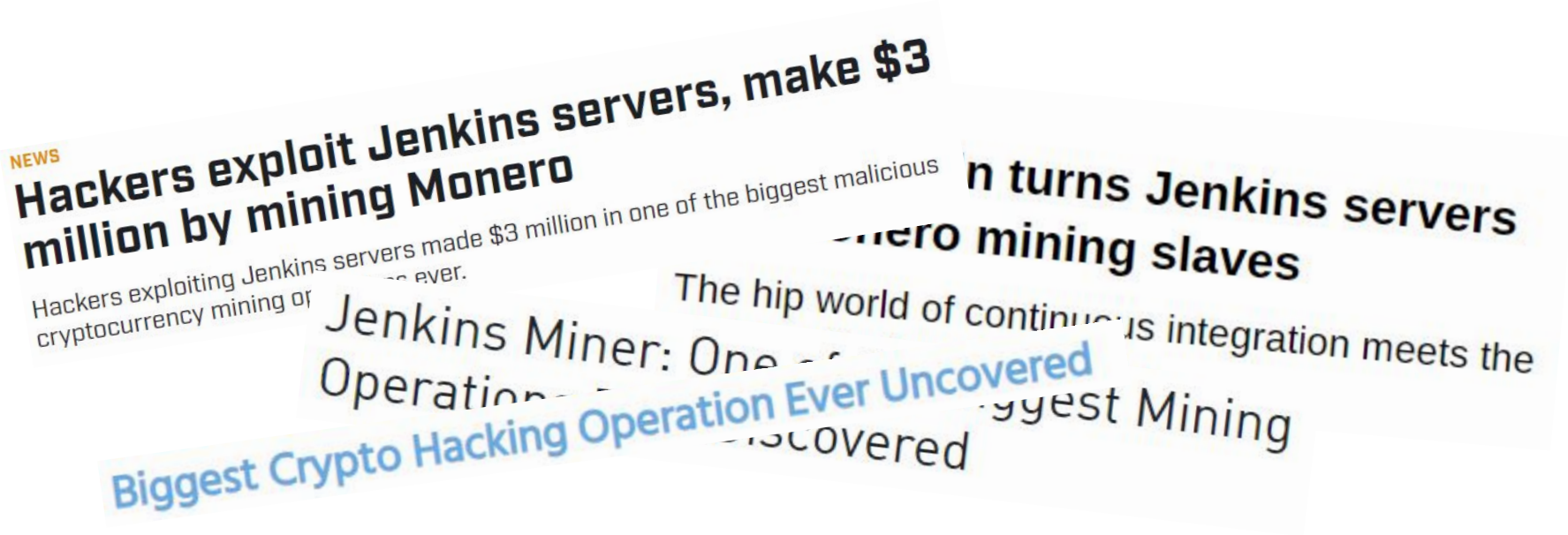
- Prevent intrusion
- Secure your data
- Avoid privilege escalation
- Enforce best practices

Main security objectives (2)

- Do not break your instance on security update
  - Update ASAP when a security advisory is published

Main security objectives (2)

- Do not break your instance on security update
  - Update ASAP when a security advisory is published



Main security objectives (3)

- About the attack
  - Discovered in February **2018**

Main security objectives (3)

- About the attack
  - Discovered in February **2018**
  - Patch was released in April **2017**

What the security team does

- jenkins.io/security/advisories

Security Advisories

This page lists all security advisories that have been published so far.

2018

- Jenkins Security Advisory 2018-10-29
  - Affects Plugins: Pipeline: Groovy Script Security
- Jenkins Security Advisory 2018-10-10
  - Affects Jenkins Core
- Jenkins Security Advisory 2018-09-25
  - Affects Plugins: Arachni Scanner Argus Notifier Artifactory Chatter Notifier Config File Provider Crowd 2 Git Changelog HipChat JIRA Job Configuration History JUnit mesos Metadata Monitoring MQ Notifie SonarQube Scanner
- Jenkins Security Advisory 2018-08-15
  - Affects Jenkins Core
- Jenkins Security Advisory 2018-07-30
  - Affects Plugins: AccuRev Agiletestware Pangolin Connector for TestRail Anchore Container Image Scann Inedo ProGet Plugin Kubernetes Maven Artifact ChoiceListProvider (Nexus) meliora-testlab Publish Ove SSH Agent Tinfoil Security TraceTronic ECU-TEST
- Jenkins Security Advisory 2018-07-18
  - Affects Jenkins Core
- Jenkins Security Advisory 2018-06-25
  - Affects Plugins: AWS CodeBuild AWS CodeDeploy AWS CodePipeline Badge CollabNet Plugins Confi IBM z/OS Connector OpenStack Cloud SAML SSH Credentials URLTrigger
- Jenkins Security Advisory 2018-06-04
  - Affects Plugins: AbsInt Astrée Black Duck Detect Black Duck Hub CAS Git GitHub GitHub Branch Sou
- Jenkins Security Advisory 2018-05-09
  - Affects Jenkins Core
  - Affects Plugins: Black Duck Hub Gitlab Hook Groovy Postbuild
- Jenkins Security Advisory 2018-04-16
  - Affects Plugins: Email Extension Google Login HTML Publisher S3 publisher
- Jenkins Security Advisory 2018-04-11
  - Affects Jenkins Core
- Jenkins Security Advisory 2018-03-26
  - Affects Plugins: Ansible Copy To Slave Cucumber Living Documentation GitHub Pull Request Builder Liq
- Jenkins Security Advisory 2018-02-26
  - Affects Plugins: Azure Slave Coverity CppNCSS Environment Injector Gerrit Trigger Git Google Play Ar promoted builds Subversion TestLink
- Jenkins Security Advisory 2018-02-14
  - Affects Jenkins Core
- Jenkins Security Advisory 2018-02-05
  - Affects Plugins: Android Lint CCM Credentials Binding JUnit Pipeline: Supporting APIs
- Jenkins Security Advisory 2018-01-22
  - Affects Plugins: Ant Checkstyle DRY FindBugs Pipeline: Nodes and Processes PMD Release Transla

Best practices

- Principle of least privilege
- Use different credentials
- Avoid running build on master
- Take care of the security warnings

What are the constraints?

Constraints for a security fix (1)

- Keep backward binary compatibility
- Confidential aspect of the security discovery until fixed
- Cannot fix only one plugin if the vulnerability is generalized in multiple plugins

Constraints for a security fix (2)

- Some vulnerabilities require structural change
- Multiple long term support versions in our commercial offering
- How to deal with features dangerous by design
- Time constraints due to disclosure policy

Constraints from community aspect

- Open-source community
- Private plugins
- Broad plugin ecosystem
- No common coding practice among plugins
  - Maintainers are not always Java developers
- Some plugins are not maintained
- Sync with maintainers for security release

Constraints from project (started in 2004)

- File system as storage by default
- Legacy parts kept for compatibility
- Lots of knowledge required to avoid breaking stuff

Our approach to tackle those problems

General process to handle reports

- Vulnerability reported in our private tracker
- Reproduce the problem
- Determine the impact / severity
- Correct the behavior
  - Either the plugin maintainer
  - Or a security team member (esp. for core)
- Review by the security team
- Synchronization for the next security release

Issue tracking

- Process: jenkins.io/security/
- Private project part of the public Jira
- Members of security team have full access
- Reporter and assignee have access to their issue

Developer education

- Improve security awareness
  - Inside community / company
  - Using documentation / best practices / talks
- Code review

Like today!

Vulnerabilities prioritisation

- Common vulnerability scoring system (CVSS)
- Popularity of the plugin (# of installations)
- Difficulty of the correction (potential breakage)

- Discussion inside the Jenkins security team

Plugin blacklisting

- Plugin with dangerous features
- Not easily fixed
- Could be temporary or permanent

Reduce the risk to break instances

- We ask the administrators to upgrade ASAP
- We cannot break their instance
- Requirement to deliver high quality code
  - Small amount of review
  - Keep changes as small as possible

Up-merge and non-regression

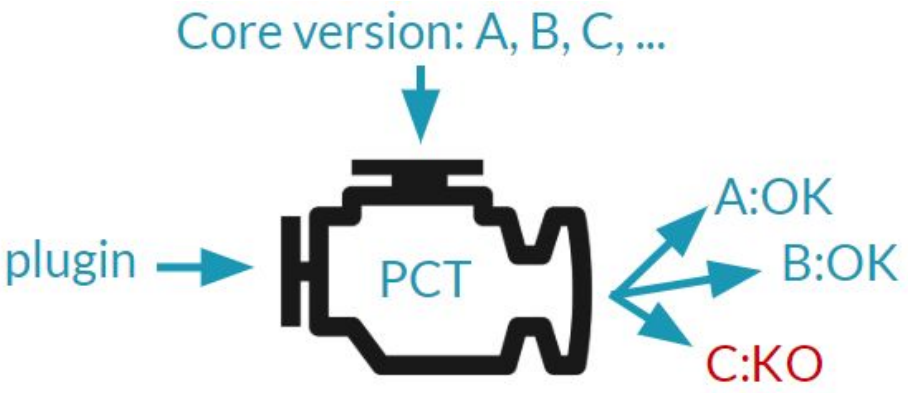
- Multiple LTS lines for CloudBees customers only
- Non-regression tests are very useful to detect conflicts

Our tools to ease our job

Plugin compatibility tester (PCT) to reduce risks

- Before release a security fix

- Ensure compatibility with the different core versions
  - No recompilation
  - Tests must pass



`@Restricted` annotation (1)

- `public` modifier in Java
- Open API for other plugins
- Jenkins requires some cases to be `public`
  - `@Extension` classes
  - Form validation

- `@Restricted(NoExternalUse.class)`
  - Enforce other plugins are not using that code

`@Restricted` annotation (2)

- Allow developer to limit their exposed API
- Reduce the risk of breaking backward compatibility
- Avoid call to methods in unintended ways / state

Default configuration for fresh / upgrade

- Existing behavior / configuration is not safe

- We add a new one that is secure
  - On fresh installation, enabled by default
  - On upgrade, we must keep the legacy behavior

Administrative monitors

- When a feature configuration is potentially dangerous
  - Display a message to the administrators
  - When possible, actionable buttons

- Ease migration from existing configuration

Escape hatches

- If the legacy / dangerous behavior was expected
- Administrators could use some system property flags to keep the legacy behavior at their own risk
- Implication of such configuration is explained in the security advisory

Telemetry

- Improve anonymous usage statistics
- Currently used to detect the unused escape hatches
- Could be used in the future to help understanding real usage of certain features

Custom scripts to search code

- During vulnerability correction
  - We need to ensure no other plugins are sharing code
  - Search for code similarities
- Avoid publishing an advisory that will disclose other vulnerabilities
- Unfortunately not possible for closed-source plugin
  - Incentive for code sharing!

API plugin

- Special kind of plugin that contain only a library
- Avoid upgrading the version in every plugin
- Esp. useful for libraries with active security research
- Examples:
  - Bouncy Castle
  - Jackson

Setup Wizard

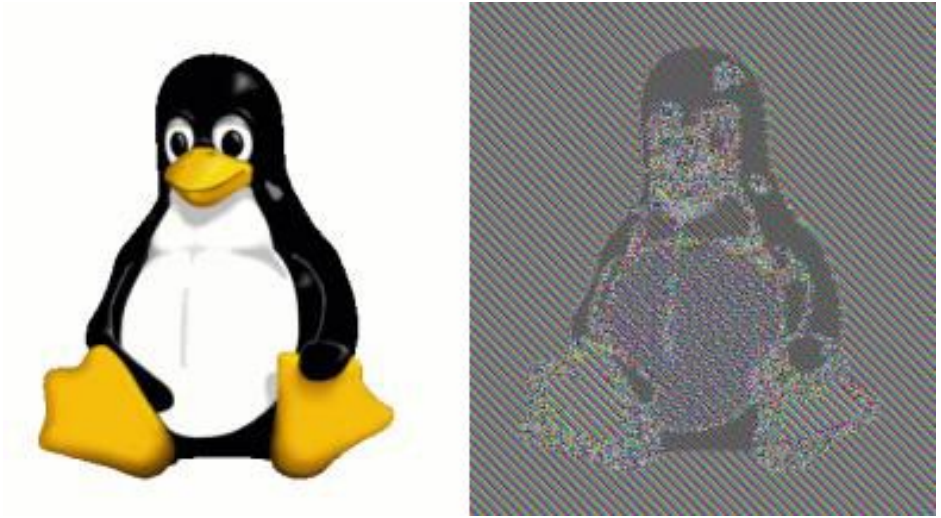
- Ease the initial configuration
- Secure by default
- Enforce best practices

`Secret` class (1)

- Allow developers to store sensible information
- Easy migration from previously plain text (`String`)
- No cryptography knowledge required

`Secret` class (2)

- Do you use `Cipher.getInstance("AES")` ?
- Default mode on Oracle JVM: ECB



Practical examples

API Token (1)

- Previous version:
  - Data was stored in encrypted form on disk
  - Was possible to recover the token value
  - Only one token per user

API Token (2)

- New version:
  - Multiple tokens per user
  - Only the hash is stored
  - Possibility to revoke token individually
  - Easy migration

XML deserialization (1)

- XML External Entity (XXE)
  - If the parser is not safely configured

```
setFeature("http://xml.org/sax/features/external-general-entities", false);
setFeature("http://xml.org/sax/features/external-parameter-entities", false);
[+ other depending on the library]
```

- Deserialization
  - URL to generate DNS requests
  - ysoserial

XML deserialization (2)

- Java deserialization could be very dangerous
- Initially we blacklisted the dangerous classes
- Too many reports
- Migrate to a whitelist pattern that is more secure

Script sandboxing

- Reduce scripting capacity
- Script approval
- Using a whitelist
  - Methods / fields the administrators pre-approve

Conclusion

- **Apply the updates!**
- Ease update process
- Beware of unsecured default settings
- Most of your users are not sufficiently “educated”