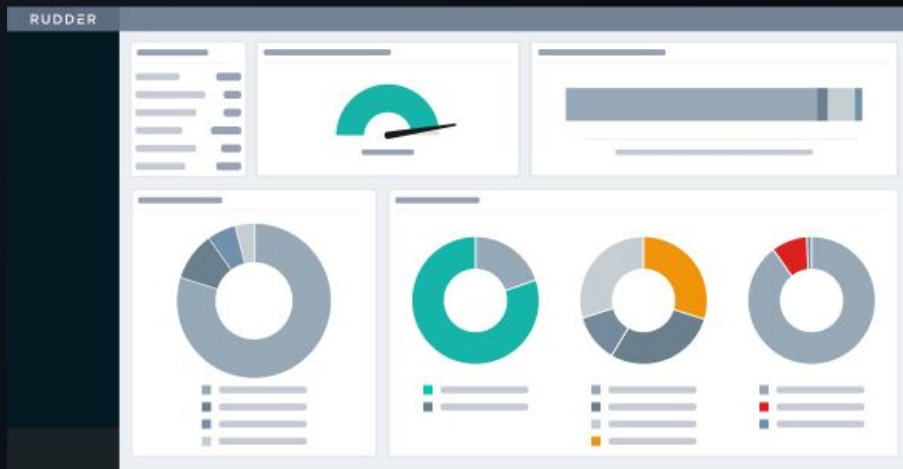


How do we make Rudder secure?

Security strategy for Rudder



Alexis Mousset

whoami

- Alexis Mousset
 - amo@rudder.io
 - twitter.com/AlexisMousset
- Sysadmin background
- Lead System Developer @Rudder
 - Agent
 - Configuration policies
 - Security
- Secure Code Working Group @Rust

Agenda

Overview of several security-related topics:

- Rudder hardening
- Vulnerabilities handling
- Software supply chain security
- Dev team culture & processes



Rudder

Rudder Server/Agent Security & Hardening

In Rudder itself

- Security of Rudder is (now) integrated as a source for the roadmap
 - Prioritized independently
- Regular integration of security items in releases

In Rudder

Node-server communication security cycle (since 6.1) (but actually started back in 2015)

- Recent TLS (1.2+) everywhere
- Switch to a unique key for both communication protocols
- Certificate verification everywhere (still TOFU by default)
- Splittable virtual env for Web/API vs. node-server communication
- Ports configuration allowing finer firewall rules

In Rudder

User authentication

- No default password for admin account
 - Especially in addition to the missing indexation configuration
- Proper hash for local passwords (bcrypt)
- Hide API tokens in UI
- 2FA with OpenID Connect/OAUTH2

In Rudder

Services hardening (6.X)

- relayd and slapd have a strict sandboxing policy
 - Run as dedicated system user
 - SELinux on RHEL
 - systemd-based hardening
 - `ProtectSystem=strict + ReadWritePaths=...`
 - `PrivateTmp=True`

In Rudder

Compilation hardening options (7.0)

- Follow best practices for C compilers
 - `-fstack-protector-strong`
 - `-Wl,-z,relro -Wl,-z,now` (full relro)
 - `-D_FORTIFY_SOURCE=2`
 - `-fPIE / -pie`

In Rudder

Frontend side (7.3)

- Proper session expiration
- XSS hardening (CSP, etc.)
- CSRF hardening (SameSite)
- HSTS (built-in setting)
- Package manager for JS/CSS dependencies (npm)

In Rudder

Identified **next steps**:

- Replace usage of AngularJS
- Don't store clear-text API tokens on the server
- Built-in 2FA (WebAuthn/TOTP)
- Stop running services listening on the network as root (jetty, cf-serverd)
- TLS 1.3
- etc.





Rudder

Vulnerability handling process

Vulnerability handling process

We now have defined a **policy & process** (before 2020: ad-hoc handling)

- Centralized reporting at: security@rudder.io
- Internal database which contains all data and actions taken
- Embargo with private tickets and late code push
- Communication
 - Customer notice in advance
 - Community notice a week after
 - Low public communication, to avoid attracting *The Eye of Sauron*

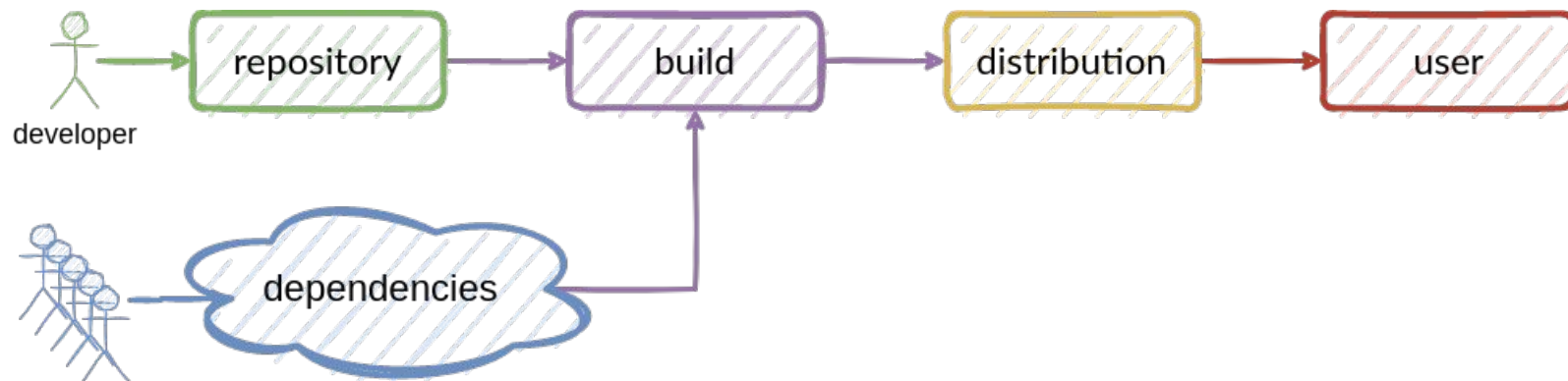
Aa ID	Description	Affected	Patched	Reported	Fix released	References	# Sco
 RUD-2022-21445	Possible CSRF on internal API	5.0 6.0 6.1 6.2 7.0 7.1	6.2.16 7.0.5 7.1.3	July 21, 2022	July 26, 2022		
 RUD-2022-21442	Various XSS vulnerabilities in the interface	5.0 6.0 6.1 6.2 7.0 7.1	6.2.16 7.0.5 7.1.3	July 20, 2022	July 26, 2022		
RUD-2021-20512	Use a proper CSPRNG to generate API tokens	5.0 6.0 6.1 6.2	6.1.19 6.2.13	January 4, 2022	April 8, 2022		
RUD-2021-20421	JNDI injection in logback configuration file	5.0 6.0 6.1 6.2	6.1.18 6.2.12	December 14, 20	December 17, 2021	CVE-2021-42550	
RUD-2021-19456	JS injection in inventory data	5.0 6.0 6.1 6.2	6.1.14 6.2.8	June 18, 2021	July 9, 2021		
RUD-2021-19442	Command injection in plugins repository file names	5.0 6.0 6.1 6.2	6.1.14 6.2.8	June 15, 2021	July 9, 2021		
RUD-2021-19252	Arbitrary get request in XML parser	6.1 6.2	6.1-1.1 6.2-1.2	May 11, 2021	May 31, 2021	CVE-2020-11988	
 RUD-2021-19211	Brute-force vulnerability for local Rudder user passwords	6.1 6.2	6.1.13 6.2.7	April 30, 2021	May 18, 2021	CVE-2020-28052	



Rudder

Software supply chain security

Context



Context

- Software supply chain risks
 - **SolarWinds**: Targeted attack against a critical supplier
 - **Log4shell**: Known vulnerability in low level components
- Will very likely continue to be an important topic in the coming years

Context

Rudder is a critical component with a lot of dependencies

- Runs everywhere with admin rights
- Talks on the network
- Exposes interfaces to multiple systems and people
- Rather complex piece of software with mixed technologies

Context

We need to:

- Carefully manage our own dependencies
- Apply good development practices regarding security
- Ensure the security of our sources, build process and linked infrastructure
- Provide the required information to our users (docs, advice, sbom, etc.)



Ad: *Tomorrow at 2 p.m., I will give a dedicated talk about software supply chain security in the security room.*

Software Supply Chain

- Dependencies : various code ecosystems
 - C (openssl, curl, CFEngine, etc.)
 - Scala/Java
 - Rust
 - F#
 - Elm
 - Javascript
 - Python
 - Perl
 - etc.

Software Supply Chain

Upgrade strategy (7.0+)

- Stricter version of our previous workflow
- Upgrade all our dependencies for every minor version (at least to a supported patch release)
- In practice, every 6 months
- This allows reacting faster to security problems (in addition to the bug fixes)
- Document and study exceptions

Software Supply Chain

Upgrade strategy

- Frontend code has been lacking in this regard
- AngularJS is a good example
- Starting from 7.3, proper package manager to improve upgrade process

Software Supply Chain

Known vulnerabilities monitoring

- Started in 2020
- Incrementally extended
- Now covers full scope

Software Supply Chain

Security monitoring

- Manual (oss-security ML, etc.)
- Automated when possible
- Runs daily
 - Evaluate every alert quickly
 - Either:
 - Ignore if not affected
 - Upgrade and publish according to our standard patch releases
 - Handle as Rudder vulnerability if serious
 - e.g. a serious vulnerability in local user authentication (bcrypt brute-forcing)

Software Supply Chain

Security monitoring

- Dedicated tools for (almost) each ecosystem
- Needs a specific integration

Software Supply Chain

Are we **SBOM** yet? Gives information to the end-user.

- Automated SBOM or nothing
- We have the required pieces
 - All dependencies versions known at build time
- A few tricky points
 - Plugins
 - Immature ecosystem in general

Infrastructure

- Bare-metal build systems
- Cloud and specific (e.g. AIX) hosting
- Ephemeral build environments
- Docker containers
 - Weekly upgrade, only official containers
 - Set as many components versions as possible
- Dedicated signature server (packages & plugins)
- Improved credentials management

Software Supply Chain

Security monitoring & remediation

- We need it for the build infra and tooling too!

Documentation

- Documentation should include best-practices
 - Avoid examples with a copy-paste command creating an “admin” user with the “admin” password
- Hardening documentation
 - Advice for stuff not includable in the default settings



Rudder

Dev Team Culture & Processes

Team Culture

- Regular internal training
 - In 2022:
 - General code security best practices
 - Frontend security
- Raising security topics awareness
- Systematic security assessment of new features/changes



Rudder

Questions?