The deputy shot the sheriff

Privilege escalation in build pipelines
it’s not just your employees

JavaScript Packages Caught Stealing Environment Variables

By Catalin Cimpanu

On August 1, npm Inc. — the company that runs the biggest JavaScript package repository — removed 38 JavaScript npm packages that were caught stealing environment variables from infected projects.

Check your repos... Crypto-coin-stealing code sneaks into fairly popular NPM lib (2m downloads per week)

Node.js package tried to plunder Bitcoin wallets

By Thomas Claburn in San Francisco 25 Nov 2016 at 20:58

RESEARCHERS FOUND BACKDOOR IN PYTHON LIBRARY THAT STEAL SSH CREDENTIALS

Recently we saw an attempt to hide a back door in a code library, and today there is a new case. This time, information security experts found the backdoor in a Python module.

In the SSH Decorator module (ssh-decorate), created by the Israeli developer Uri Goren, which is a library for handling SSH connections from the Python code.
Agenda

- evolution of software delivery systems
- potential problems
- solution example
- mitigation strategies

q & a
Andreas Sieferling
Senior Cloud Platform Engineer

Twitter: @webratz
All things AWS, CI/CD, GitHub
Evolution of Software Delivery

Code → target system
Evolution of Software Delivery
Evolution of Software Delivery

- Code
- VCS
- black magic
- target systems
Evolution of Software Delivery

- Code
- VCS
- black magic
- target systems
Evolution of Software Delivery

VCS

black magic

target systems
Evolution of Software Delivery

- Workflow got more complicated
- More involved components
- Often shared components
- Bigger user base
- Often very centralized
black magic

it’s not
What does a CI/CD pipeline
What does a CI/CD pipeline
The confused deputy

In a picture
A confused deputy is a legitimate, more privileged computer program that is tricked by another program into misusing its authority on the system. It is a specific type of privilege escalation.

- Wikipedia
Confusing the CI/CD system

VCS

black magic

target systems
Confusing the CI/CD system

- fetch different repo
- escape build env
- use different of many target roles
What does a CI/CD pipeline | Problems

Takes code, builds it in a controlled env, deploys it to some environment

- Big, central systems have a huge blast radius
- Acts on behalf of someone with its own identity
- Masks / separates original / triggering user
- Might even make changes to repo
- Effectively allows everyone with push access, access to prod
- All components have own IAM, usually not synced in any way
- Credentials need to be exposed
- Components don’t identify each other
- Artifacts are not signed
- build untrusted code (eg. open source)
How to fix?
Step by step
Acting on behalf

- Acts on behalf of someone with its own identity
- Masks original / triggering user
Acting on behalf

Code

Build

Deploy

Deploy Target
Acting on behalf

Option 1:
   Pass on role with the commit. Afaik not possible right now

Option 2:
   • Make all systems identity aware, do not allow to go beyond permissions of pusher
   • Remove permission management in between if possible, if needed check out of band
   • Reduce confusion possibilities
Example solution that is in use at Scout24
uses common components: GitHub, Jenkins, AWS

sorry for the complex graphic
1) Webhook
   - Jenkins
   - GitHub
   - Azure AD

2) Create ephemeral Agent

3) Request temp credentials

4) Request trusted metadata

Cross Account proxy 169.254.169.254

AWS STS

Target AWS Account

Target role

Custom AWS Auth Service maps Groups to roles in AWS Accounts

Get Policy of User
Build Agent -> 3) request temp credentials -> Cross Account proxy 169.254.169.254

trusted metadata store

-> 4) request trusted metadata

-> 5) map GitHub User to AD Identity

-> 6) Fetch group memberships

Jenkins

1) Webhook

GitHub

0) Push

Azure AD

Custom AWS Auth Service maps Groups to roles in AWS Accounts

Get Policy of User

AWS STS

Target AWS Account

Target role
3) request temp credentials
4) request trusted metadata
5) map GitHub User to AD Identity
6) Fetch group memberships
7) map groups to AWS Permissions
8) get policy for User in given Account
Pro & Con of solution

pro
• one identity used
• out of band check of permissions of user
• user can not gain additional rights
• transparent to existing tools
• target role can restrict to repo
• credential life-time can be very short
• Things can be traced back to user due to Session Name containing the ID

con
• reduced user management, but still options for confusion
• complexity increases
• creator of IAM role needs to ensure external ID check
• needs mapping of git users to roles in AWS accounts
• might not work with other platforms
Blast radius

Big, central systems have a huge blast radius

- Gaining access to one component gives access everywhere
- Outages affect everyone
Blast radius
Blast radius

VCS

CI/CD

target systems

@webratz | Privilege escalation in build pipelines | 2019-11-06 | Velocity Berlin
Blast radius

VCS

CI/CD

target systems
Blast radius

- Segment into many small independent systems
- Automate / standardize these as much as possible

Scout24:
Had one huge build system for each AutoScout24 and ImmoScout24
Now: over 100 small but automated and standardized instances with limited scope
Exposed credentials

- We need credentials
- Credentials could be echoed
- Credentials are sent to a malicious third party
- Credentials could be stored somewhere and used in other contexts

RESEARCHERS FOUND BACKDOOR IN PYTHON LIBRARY THAT STEAL SSH CREDENTIALS

Recently we saw an attempt to hide a back door in a code library, and today there is a new case. This time, information security experts found the backdoor in a Python module.

In the SSH Decorator module (ssh-decorate), created by the Israeli developer Uri Goren, which is a library for handling SSH connections from the Python code.
Exposed credentials

- There will always be a need to expose credentials to build & deploy
- Trust to 3rd party dependencies whole topic itself – but locking helps

Reduce impact of stolen tokens:

- limit scope to what's really needed
- rotate very often (at least hourly)
Pushing from within CI/CD

- Whole identity model bases upon actual git users
- No way to track / trace changes done by machine users
- Often non-scoped credentials are in use: CI/CD system can push anywhere 😱
Pushing from within CI/CD

- Don’t
- Find alternative strategies (eg release via Tags)

If you have to:
- Get user & repo scoped credentials
- handling follow up actions as initiated by pushing user
- alternative: don’t run if you can’t identify pushing user
Push to Git == Full Access to prod

- Everyone with push access to your repo can access prod (and more)
- PR builds from forks can be dangerous as source is unclear
Push to Git == Full Access to prod

- Deal with it
- Use same auth source for everything: Sync users and groups
- Be careful with (fork) PR builds. Never give them access to prod credentials / don’t build them
- Branch protection & mandatory code reviews / few trusted writers
- Regular reviews of permissions
- Use permissions of pusher for following steps (not committer)
And now?

- Are we there yet? – Sorry, nope
- Tackle easy things first
- Build capabilities to link all permissions to identities
- Get rid of separate permission management wherever possible
- Improve step by step
- Talk about it
How does Scout24 handle GitHub access?

- [https://www.youtube.com/watch?v=2psQDViMGlc](https://www.youtube.com/watch?v=2psQDViMGlc) | Talk at GitHub Satellite by Jannet Faiz

Detailed Info about Scout24 CI/CD system (mid 2018)

- [https://www.slideshare.net/PhilippGarbe1/run-jenkins-as-managed-product-on-ecs-aws-meetup](https://www.slideshare.net/PhilippGarbe1/run-jenkins-as-managed-product-on-ecs-aws-meetup)
Privilege escalation in build pipelines

kthxbye

Andreas Sieferling
Senior Cloud Platform Engineer
@webratz