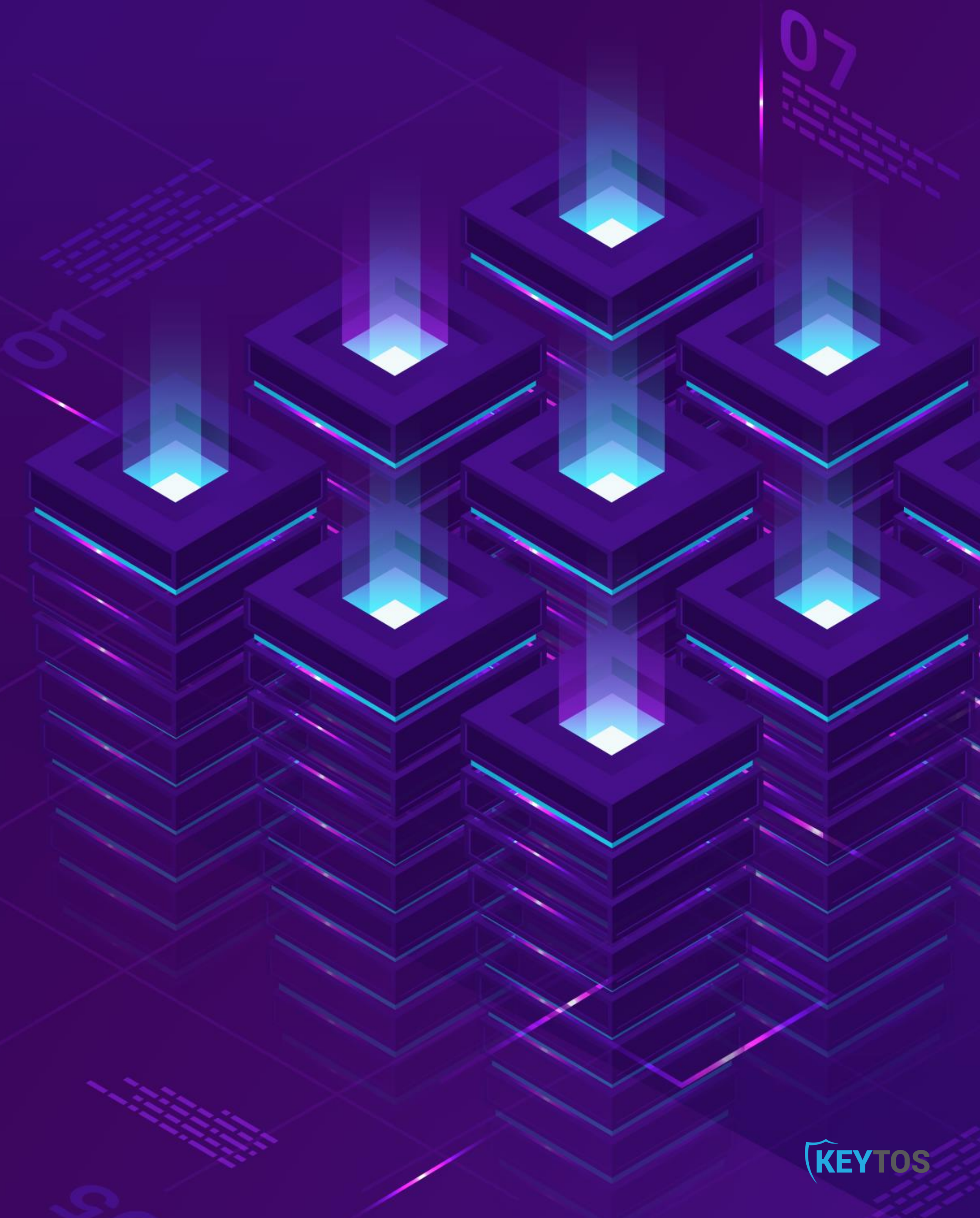
KEYTOS

EZSSH
SSH Made Easy!



KEYTOS

PROBLEM OVERVIEW

Cloud adoption is making companies move to a zero-trust networks.

99% of compromises involve a stolen credential[1]

Stolen SSH credential attacks are on the rise.

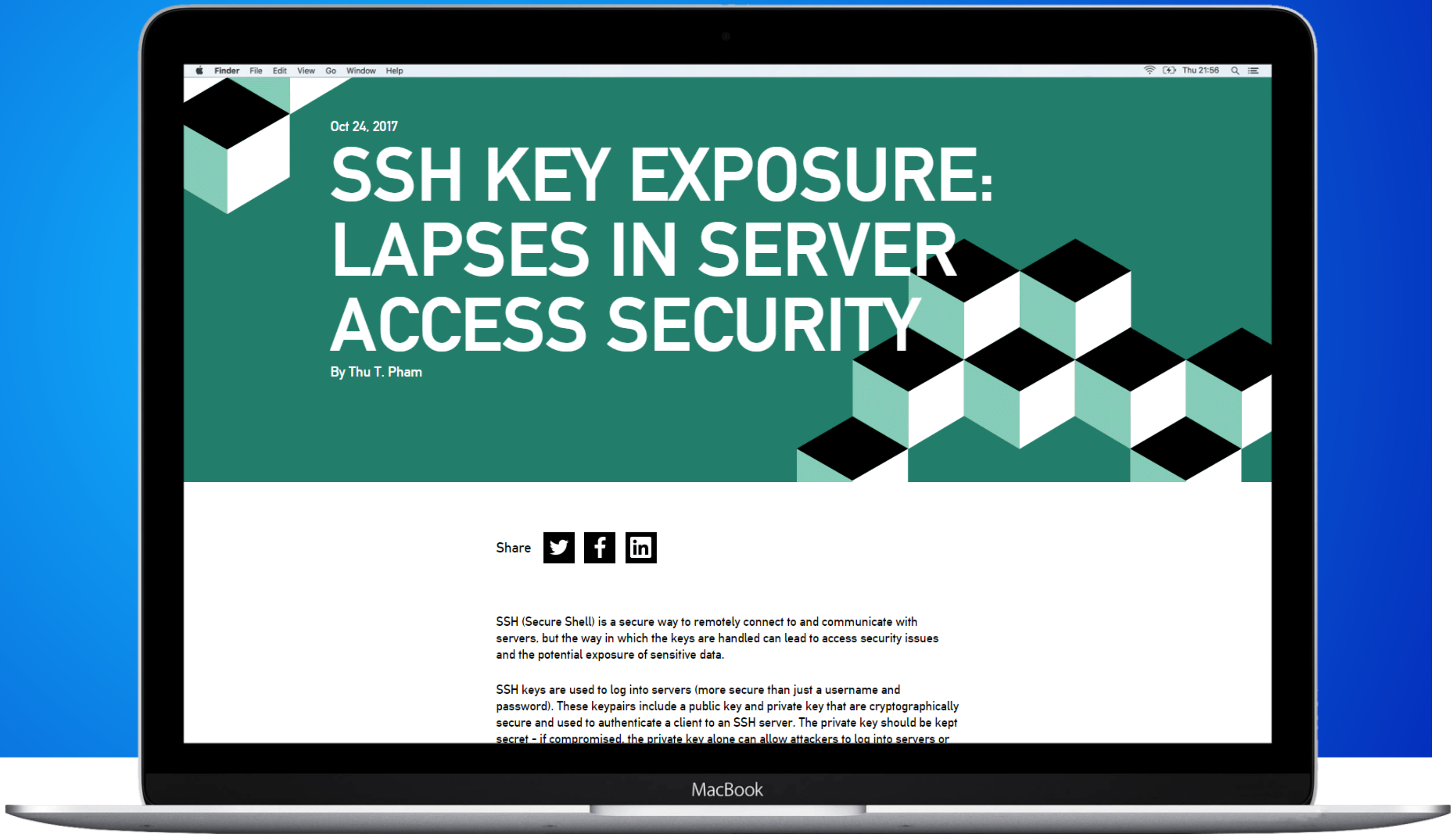
Companies are spending millions of dollars on improving their corporate identity.

Linux servers do not use Active Directory Accounts.

Thousands of keys are leaked on GitHub each day.[2]

KEYTOS

IN THE NEWS



Oct 24, 2017

SSH KEY EXPOSURE: LAPSES IN SERVER ACCESS SECURITY

By Thu T. Pham

Share

SSH (Secure Shell) is a secure way to remotely connect to and communicate with servers, but the way in which the keys are handled can lead to access security issues and the potential exposure of sensitive data.

SSH keys are used to log into servers (more secure than just a username and password). These keypairs include a public key and private key that are cryptographically secure and used to authenticate a client to an SSH server. The private key should be kept secret - if compromised, the private key alone can allow attackers to log into servers or

KEYTOS

WHY GO PASSWORD LESS?

Passwords are no longer secure due to brute force attacks.

72% of individuals reuse passwords in their personal life while nearly half (49%) of employees simply change or add a digit or character to their password when updating their company password every 90 days.

Compromised passwords are responsible for 81% of hacking-related breaches, according to the Verizon Data Breach Investigations Report.

Microsoft recently announced that a staggering 44 million accounts were vulnerable to account takeover due to compromised or stolen passwords.

KEYTOS

ARE SSH KEYS THE SOLUTION?

→ Linux servers in large corporations have between 50 and 200 SSH keys.
  - 90% of those keys are not used.
  - SSH keys never expire.
  - 50-200 keys per server.

→ Keys must be manually life cycled.

→ No Advance Identity Protection
  - Conditional Access
  - Smart Alerting
  - Just in Time Access

→ Hard to keep inventory of which key gives access to who.

→ Engineers don't follow best practices to protect the keys.

→ Current Linux Systems are protected in two ways:
  - Creating an account for production and sharing the credentials among engineers.
  - Creating accounts for each engineer in each of the servers.

KEYTOS

CURRENT WORKFLOW

Each User Gets an Account

Engineer goes to a site and learns how to create an SSH key.

Engineer creates the key.

Sends it to security team or server admin to be added to the server.

KEYTOS

CURRENT WORKFLOW

Each User Gets an Account

Security team adds it to the server.

Engineer gets access to the server and now can start their work.

When engineer no longer needs access, is the account removed?

KEYTOS

EACH ENGINEER CREATES AN ACCOUNT PROBLEMS

Engineer goes to a site and learns how to create an SSH key.

Poor key hygiene, no key clean up since it is hard to keep track of who still needs access.

Long and tedious access reviews.

High price to onboard new team member

Key reuse over different scopes.

Keys are not properly protected by users.

KEYTOS

CURRENT WORKFLOW

Shared Accounts

Engineer goes to team wiki to get key locations

Many of these keys are reused between test and production.

Engineer gets the key from team shared location

Engineer saves the key in their system

Engineer accesses server and now can start their work

KEYTOS

ONE ACCOUNT FOR ALL ENGINEERS

Usually, keys are shared in unsecure ways such as: email, file shares, wikis, git.

Hard to rotate since all engineers would have to get the new key.

When an employee leaves, they can maintain access to servers.

Big insider threat opportunity (61% of CISOs worry about insider threats).

→ Not possible to know who did which change since all server logs show being done by the same account.

No approval flows to get access to server

Reuse of “team keys” for many services.

KEYTOS

YOUR CURRENT COST

- Engineering time creating and passing the keys
- Security team time adding and removing keys
- Added risk for having engineers manage key and access
- Added risk for life cycling accounts out of the environment when no longer needed
- SSH key inventory and access reviews for all your servers

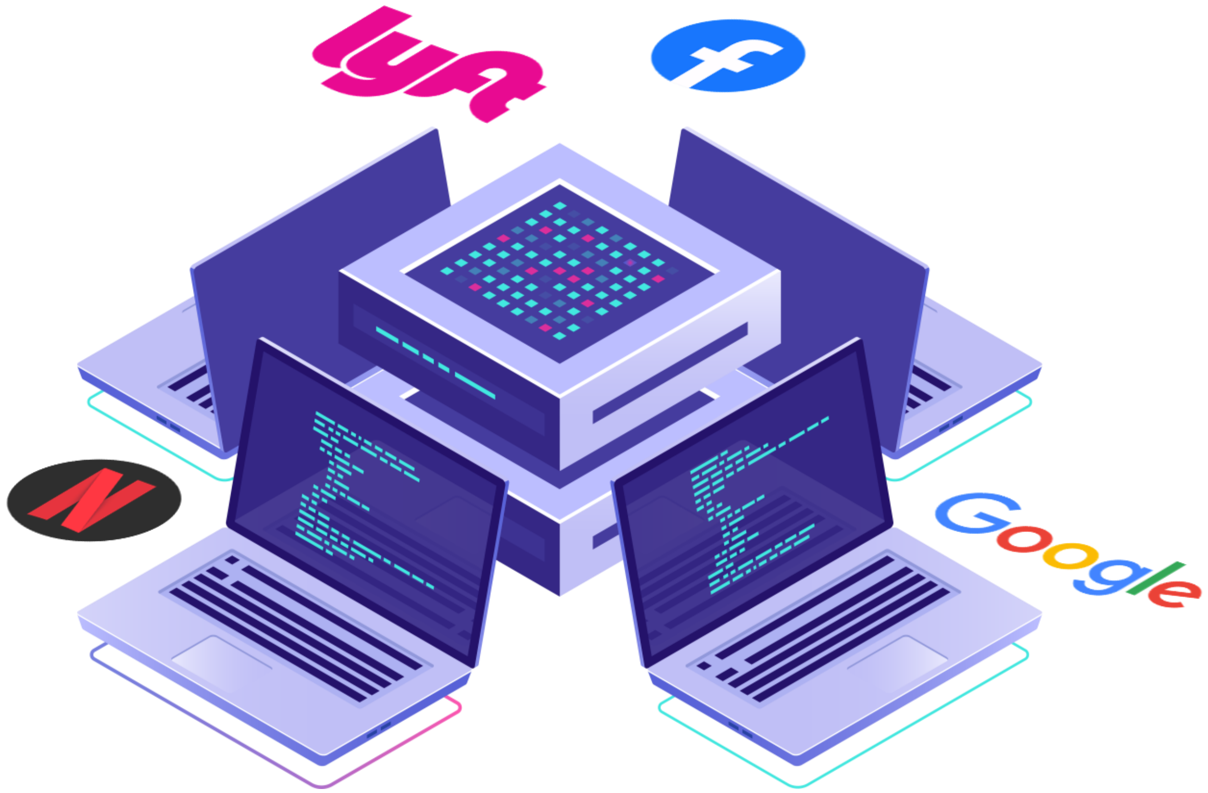
KEYTOS

SSH CERTIFICATES

Poorly documented

Need cryptographic knowledge

Manual setup and management



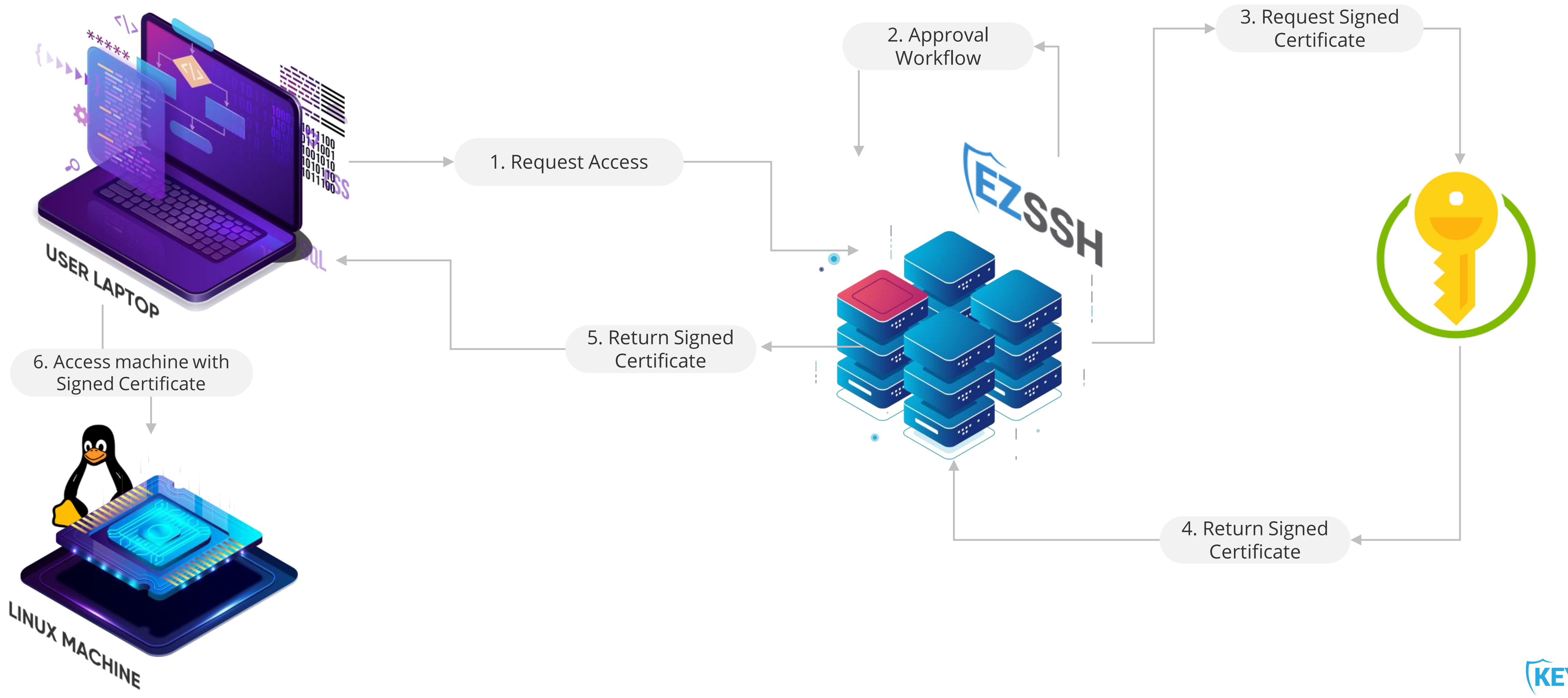
No automatic provisioning

No approval workflow

Most of the large companies use it (with custom built tools)

KEYTOS

HOW CERTIFICATES WORK


USER LAPTOP
1. Request Access
2. Approval Workflow
EZSSH
3. Request Signed Certificate
4. Return Signed Certificate
5. Return Signed Certificate
6. Access machine with Signed Certificate
LINUX MACHINE

KEYTOS

OUR SOLUTION

Seamlessly integrates with Azure

- Works with Azure Security tools such as Azure JIT and Azure PIM.
- Integrates with Azure RBAC for automatic access management.
- Automatically adds Azure Servers to your policies.

Works with hybrid and multi-cloud.

Easy setup for all your servers.

Uses your secure corporate account to create time bound certificates.

Makes security transparent to the user

Automatically onboards new team members

Approval workflow for critical environments

Automatically removes access when no longer needed

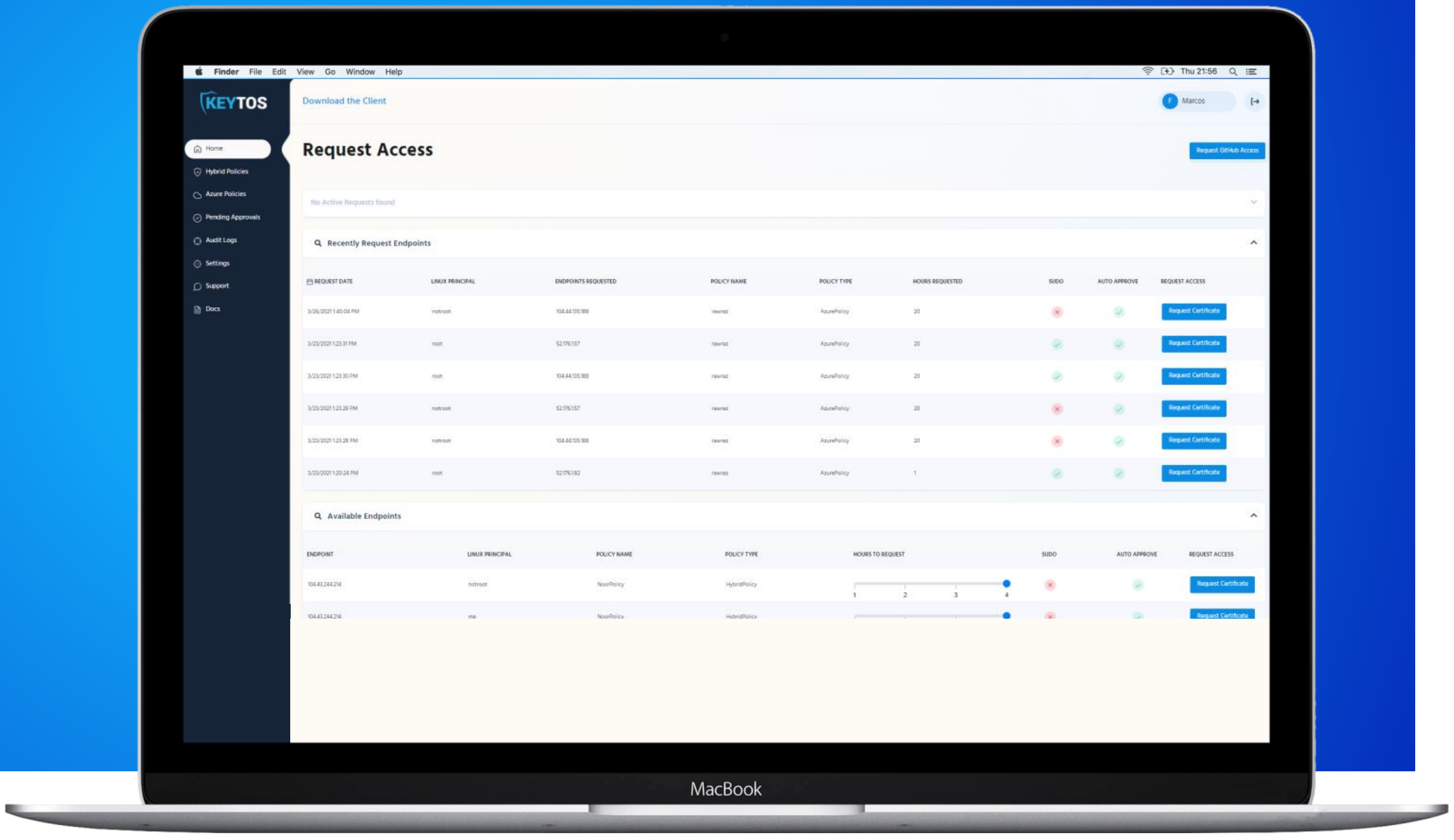
KEYTOS

EZSSH ADVANTAGES

- ✓ Designed for Zero Trust networks
- ✓ Reduce Onboarding time and cost by removing need to manage SSH keys
- ✓ Remove key management overhead from engineers.
- ✓ Reduce insider threat by having Just In Time Access with appropriate approval workflows.
- ✓ Reduce audit costs with easy to Audit access logs
- ✓ Reduce offboarding time and risk.
- ✓ Native Linux Authentication no custom PAM module or code runs on your servers.
- ✓ Bring your own Certificate Authority support

KEYTOS

DEMO



Request Access

No Active Requests found

Recently Request Endpoints

REQUEST DATE	LINUX PRINCIPAL	ENDPOINTS REQUESTED	POLICY NAME	POLICY TYPE	HOURS REQUESTED	SUDO	AUTO APPROVE	REQUEST ACCESS
3/26/2021 1:40:04 PM	notroot	104.44.135.188	rawraz	AzurePolicy	20	✗	✓	Request Certificate
3/23/2021 1:23:31 PM	root	52.176.157	rawraz	AzurePolicy	20	✓	✓	Request Certificate
3/23/2021 1:23:30 PM	root	104.44.135.188	rawraz	AzurePolicy	20	✓	✓	Request Certificate
3/23/2021 1:23:29 PM	notroot	52.176.157	rawraz	AzurePolicy	20	✗	✓	Request Certificate
3/23/2021 1:23:28 PM	notroot	104.44.135.188	rawraz	AzurePolicy	20	✗	✓	Request Certificate
3/23/2021 1:20:24 PM	root	52.176.182	rawraz	AzurePolicy	1	✓	✓	Request Certificate

Available Endpoints

ENDPOINT	LINUX PRINCIPAL	POLICY NAME	POLICY TYPE	HOURS TO REQUEST	SUDO	AUTO APPROVE	REQUEST ACCESS
104.43.244.214	notroot	NoorPolicy	HybridPolicy	1 2 3 4	✗	✓	Request Certificate
104.43.244.214	me	NoorPolicy	HybridPolicy		✗	✓	Request Certificate

KEYTOS

EZGIT
Protecting GitHub with SSH Certificates

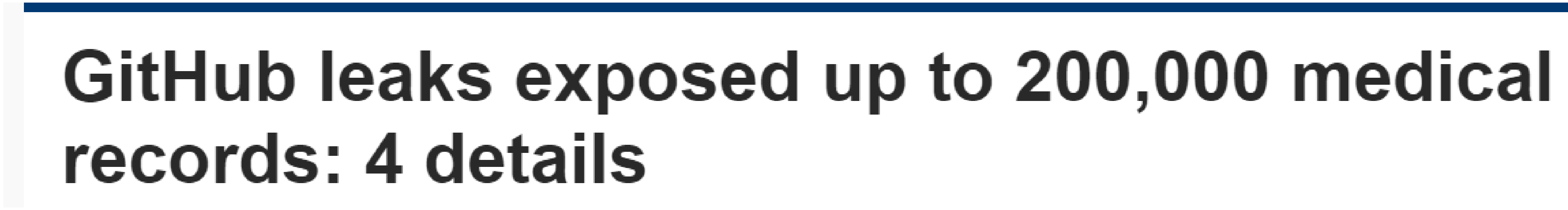
KEYTOS

PROBLEM OVERVIEW

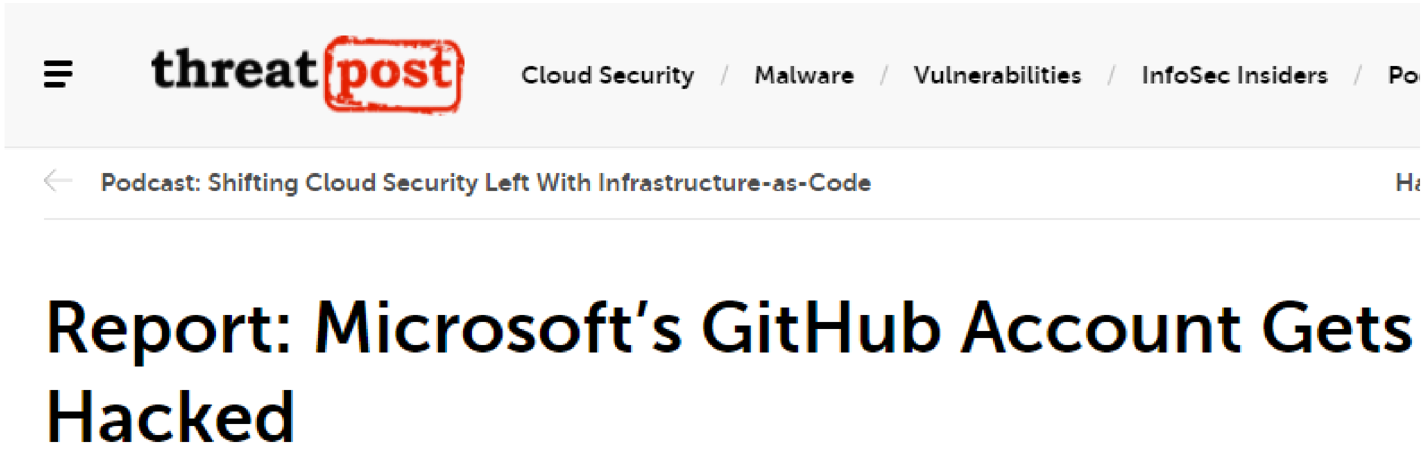
- Hackers are targeting developer credentials to steal code.
- SSH Keys are not properly managed by users.
- SSH Certificates are supported but there is no infrastructure to issue them.
- Need Secure infrastructure to run your own Certificate Authority.
- Conditional Access does not apply to the most critical operations

KEYTOS

GitHub Breaches



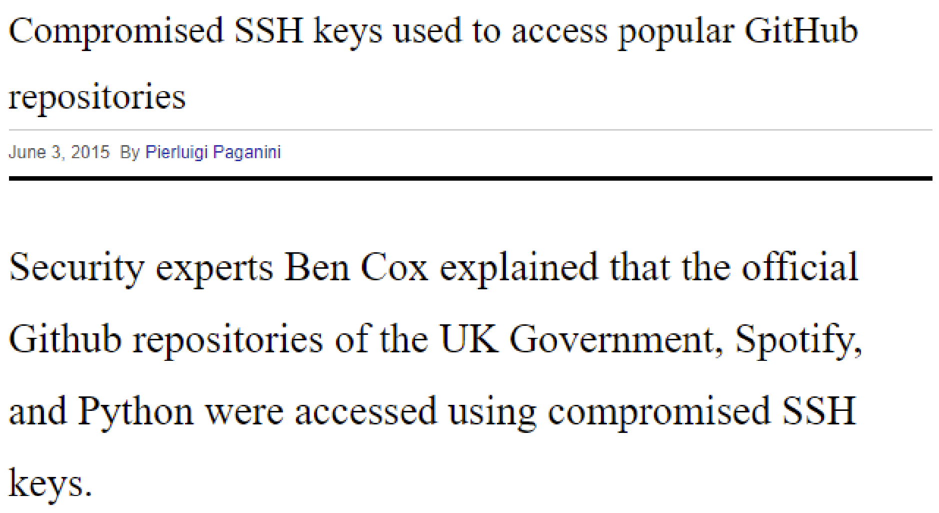
GitHub leaks exposed up to 200,000 medical records: 4 details



threatpost Cloud Security / Malware / Vulnerabilities / InfoSec Insiders / Po

Podcast: Shifting Cloud Security Left With Infrastructure-as-Code Ha

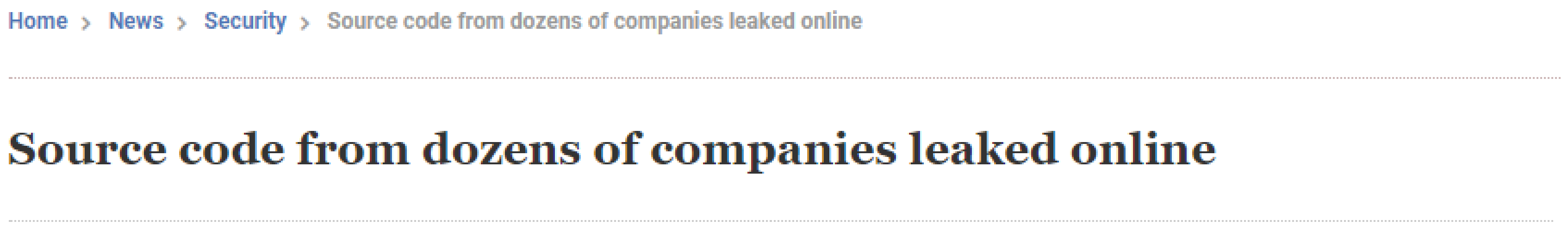
Report: Microsoft's GitHub Account Gets Hacked



Compromised SSH keys used to access popular GitHub repositories

June 3, 2015 By Pierluigi Paganini

Security experts Ben Cox explained that the official Github repositories of the UK Government, Spotify, and Python were accessed using compromised SSH keys.



Home › News › Security › Source code from dozens of companies leaked online

Source code from dozens of companies leaked online

OPPORTUNITY

- GitHub is forcing you to go password-less in 2021.
  - Gives you an opportunity to modernize your development security stack.

Reduce surface area with short-term SSH Certificates

Make audits easier with easy to audit logs

Reduce engineer onboarding time

Make security transparent for your users.

KEYTOS

OUR SOLUTION

Uses your Secure Azure AD Identity for Authentication of your developers.

Seamlessly integrates with our VM offering

Easy setup with any Git offering.

Uses your secure corporate account to create time bound certificates.

Makes security transparent to the user

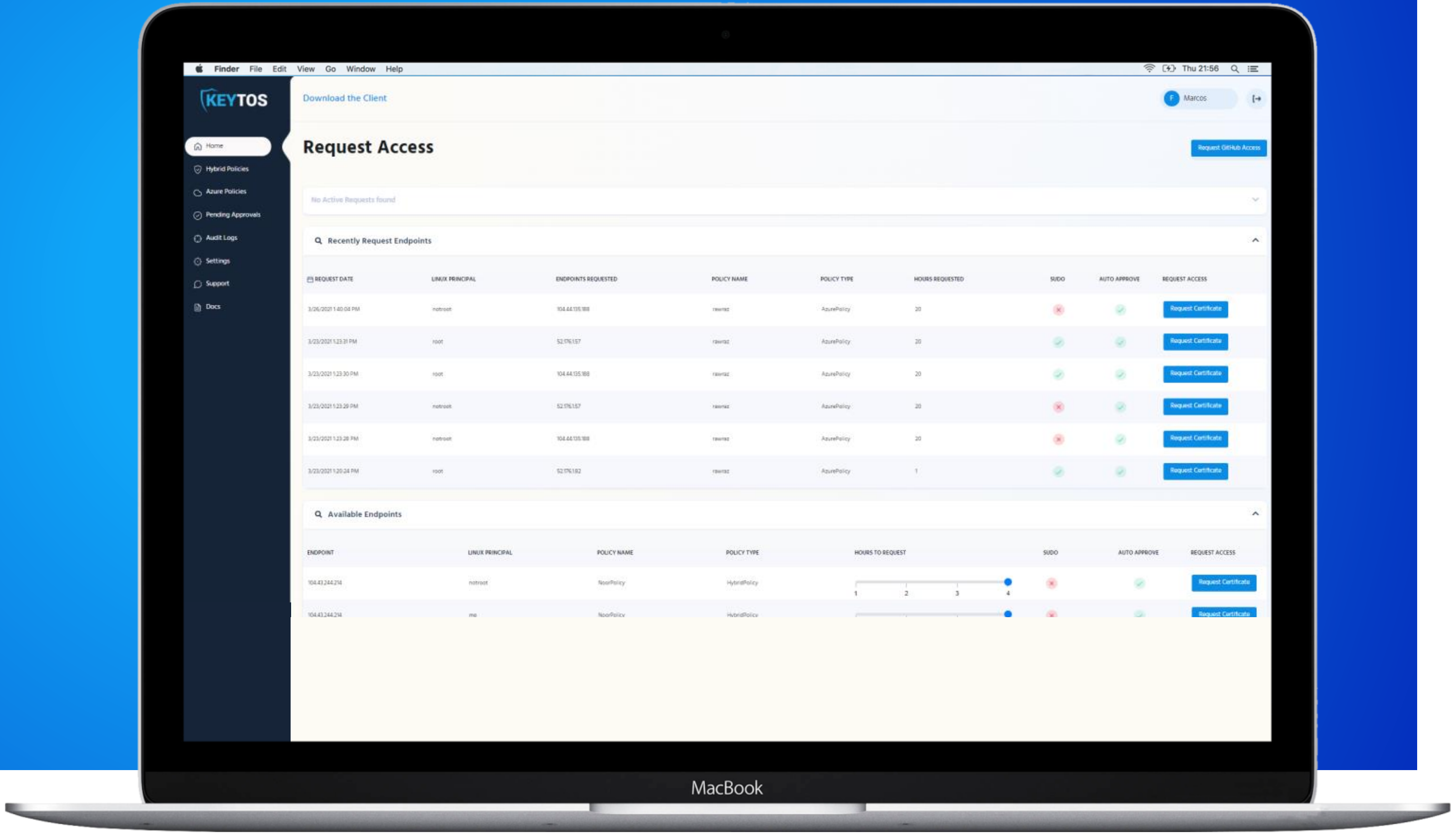
Automatically onboards new team members

Integrates with any git tool that uses ssh-agent as authentication method.

Automatically removes access when no longer needed

KEYTOS

DEMO



Request Access

No Active Requests found

Recently Request Endpoints

REQUEST DATE	LINUX PRINCIPAL	ENDPOINTS REQUESTED	POLICY NAME	POLICY TYPE	HOURS REQUESTED	SUDO	AUTO APPROVE	REQUEST ACCESS
3/26/2021 1:40:04 PM	notroot	104.44.135.188	rawraz	AzurePolicy	20	✗	✓	Request Certificate
3/23/2021 1:23:31 PM	root	52.176.157	rawraz	AzurePolicy	20	✓	✓	Request Certificate
3/23/2021 1:23:30 PM	root	104.44.135.188	rawraz	AzurePolicy	20	✓	✓	Request Certificate
3/23/2021 1:23:29 PM	notroot	52.176.157	rawraz	AzurePolicy	20	✗	✓	Request Certificate
3/23/2021 1:23:28 PM	notroot	104.44.135.188	rawraz	AzurePolicy	20	✗	✓	Request Certificate
3/23/2021 1:20:24 PM	root	52.176.182	rawraz	AzurePolicy	1	✓	✓	Request Certificate

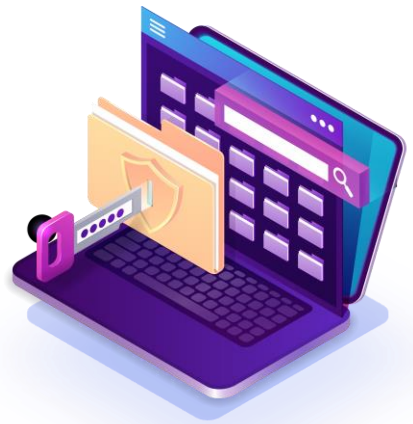
Available Endpoints

ENDPOINT	LINUX PRINCIPAL	POLICY NAME	POLICY TYPE	HOURS TO REQUEST	SUDO	AUTO APPROVE	REQUEST ACCESS
104.43.244.214	notroot	NoorPolicy	HybridPolicy	1 2 3 4	✗	✓	Request Certificate
104.43.244.214	me	NoorPolicy	HybridPolicy		✗	✓	Request Certificate

KEYTOS

HISTORY OF SSH AUTHENTICATION

1995 SSH Created



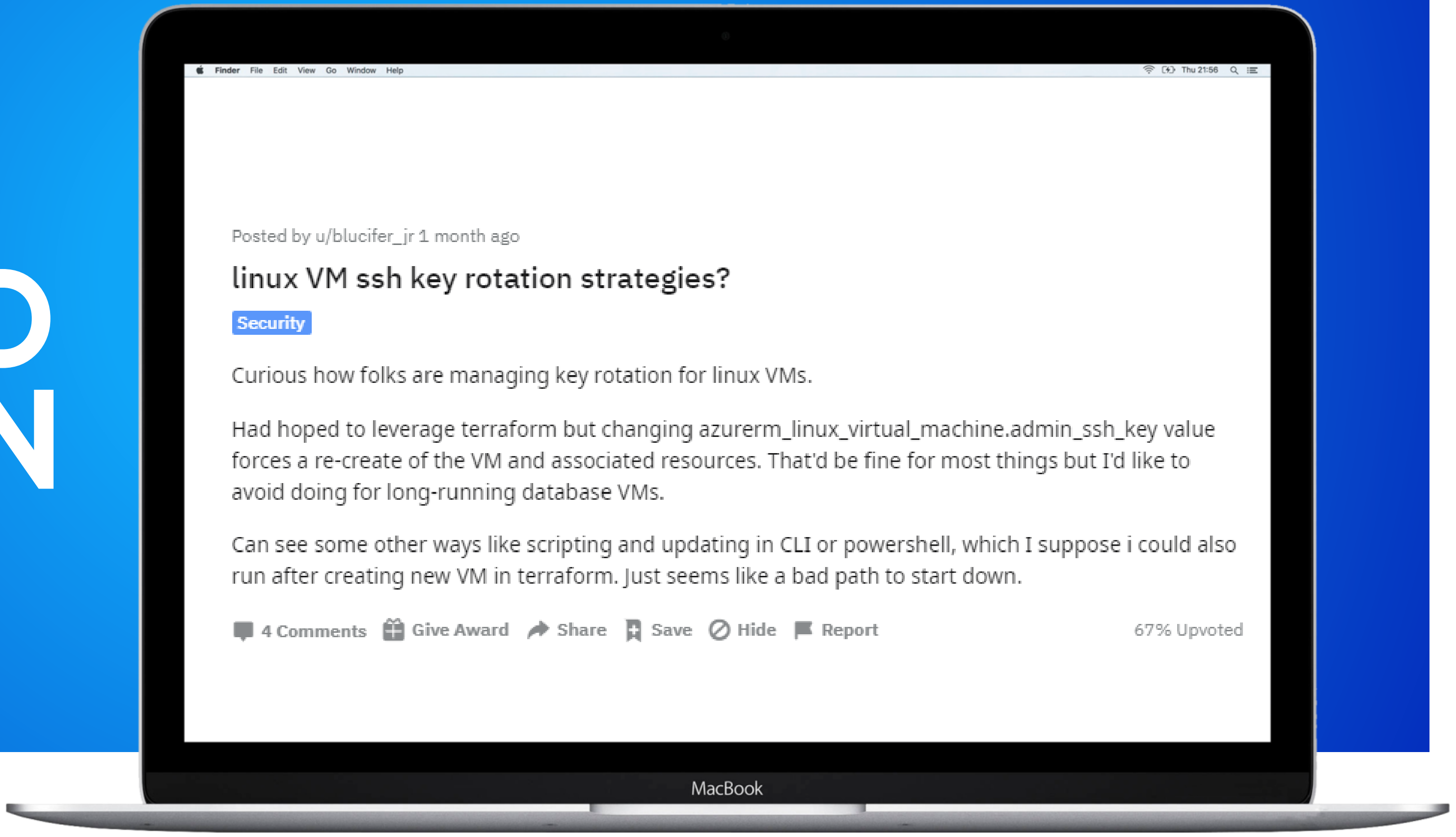
2010 SSH Certificates introduced

EZSSH

2021 EZSSH Makes SSH Certificates easy to use

KEYTOS

AZURE USERS NEED A SOLUTION



Posted by u/blucifer_jr 1 month ago

linux VM ssh key rotation strategies?

Security

Curious how folks are managing key rotation for linux VMs.

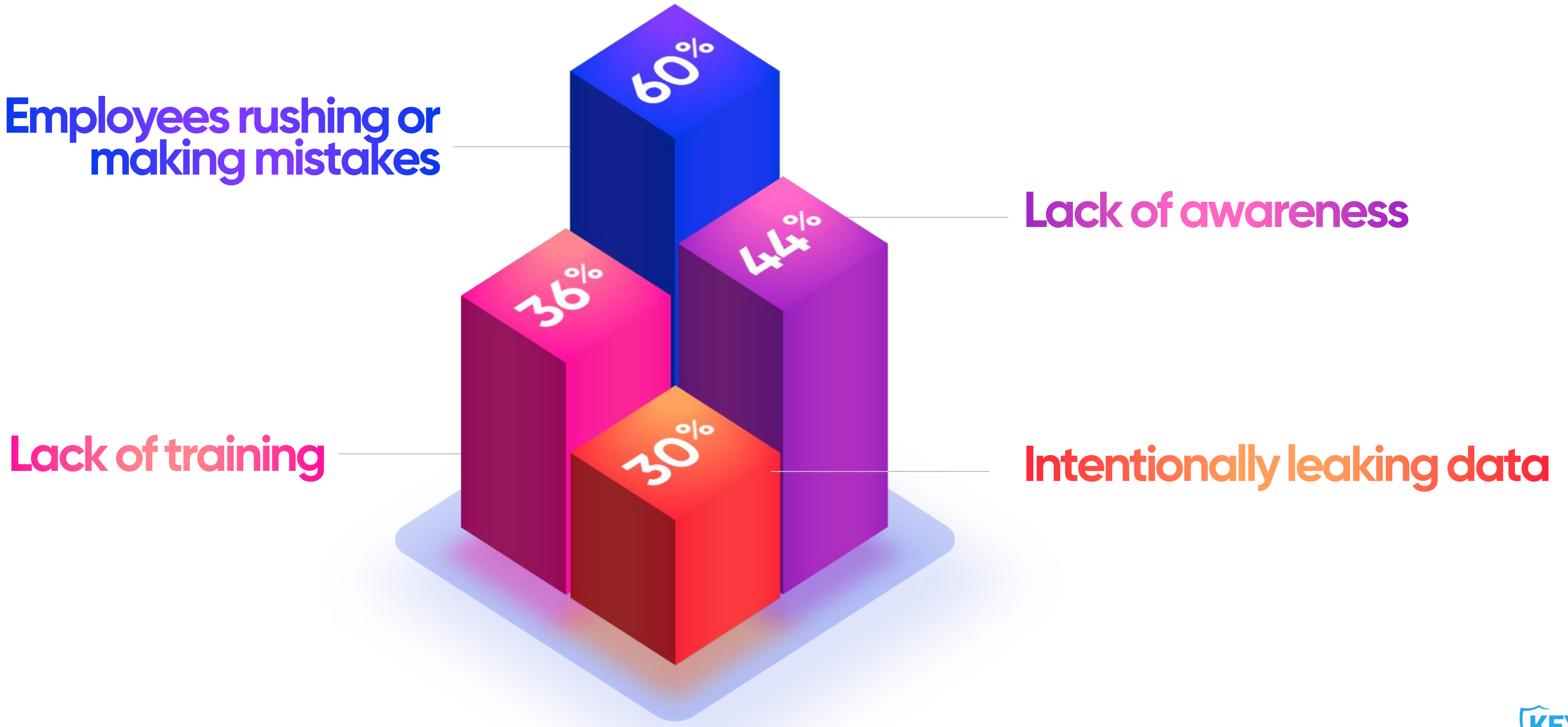
Had hoped to leverage terraform but changing azurerm_linux_virtual_machine.admin_ssh_key value forces a re-create of the VM and associated resources. That'd be fine for most things but I'd like to avoid doing for long-running database VMs.

Can see some other ways like scripting and updating in CLI or powershell, which I suppose i could also run after creating new VM in terraform. Just seems like a bad path to start down.

KEYTOS

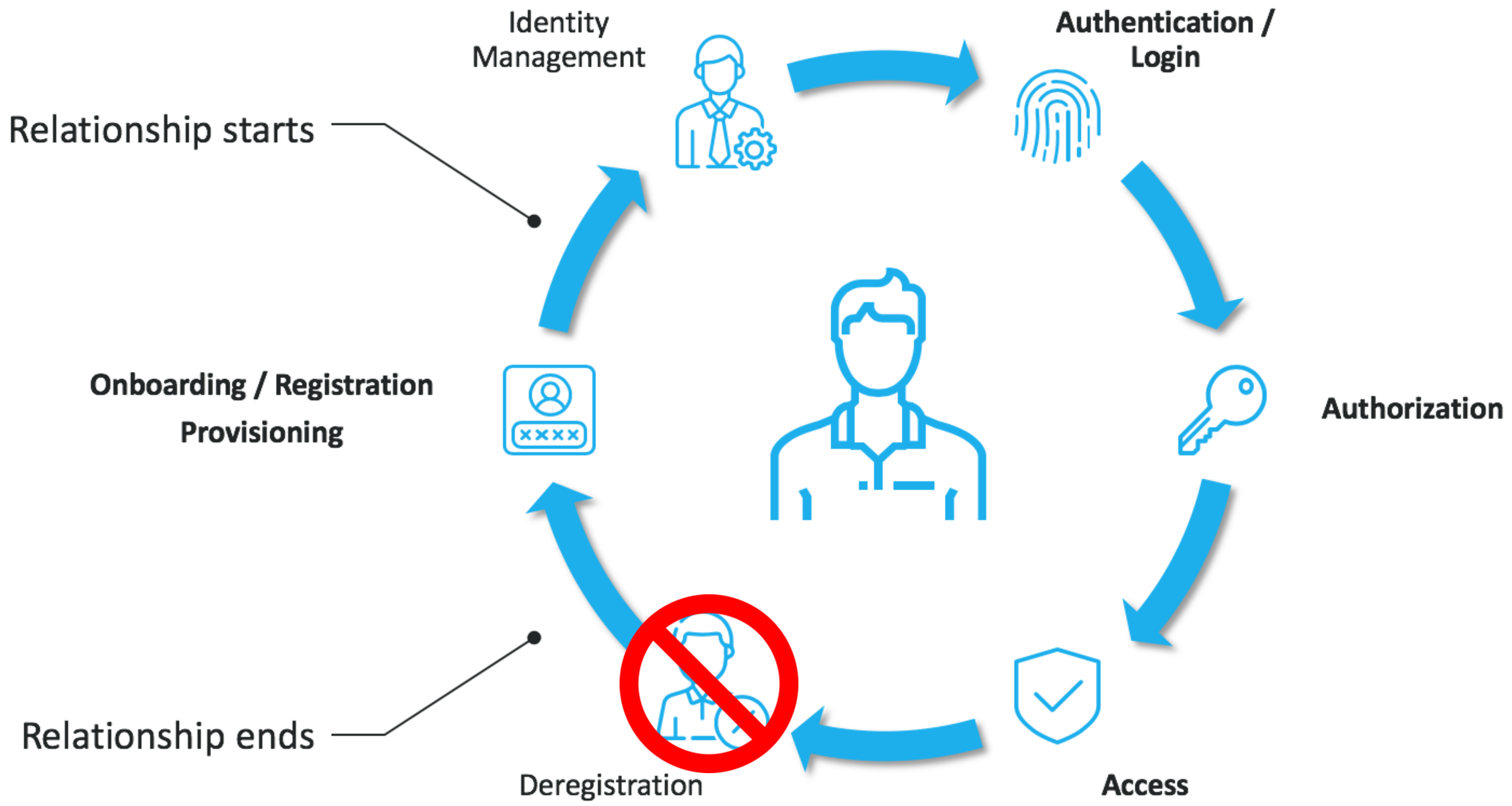
ENGINEERS MAKE MISTAKES

The bulk of insider data breaches



KEYTOS

IDENTITY LIFE CYCLE



KEYTOS

SSH UNDER ATTACK

- https://blog.ssh.com/ssh-key-scan-attack-honeypot
- https://www.zdnet.com/article/linux-under-attack-compromised-ssh-keys-lead-to-rootkit/
- https://securityaffairs.co/wordpress/37459/cyber-crime/compromised-ssh-keys.html
- https://www.beckershospitalreview.com/cybersecurity/github-leaks-exposed-up-to-200-000-medical-records-4-details.html
- https://thehackernews.com/2021/08/how-companies-can-protect-themselves.html
- https://www.lightreading.com/security/t-mobile-admits-breach-after-epic-hacking-claims/d/d-id/771524

2021 had the highest average cost in 17 years

Data breach costs rose from USD 3.86 million to USD 4.24 million, the highest average total cost in the 17-year history of this report.

Remote work due to COVID-19 increased cost

The average cost was USD 1.07 million higher in breaches where remote work was a factor in causing the breach, compared to those where remote work was not a factor.

Compromised credentials caused the most breaches

The most common initial attack vector, compromised credentials, was responsible for 20% of breaches at an average breach cost of USD 4.37 million.

Security AI had the biggest cost-mitigating effect

Automation and security artificial intelligence (AI), when fully deployed, provided the biggest cost mitigation, up to USD 3.81 million less than organizations without it.

A zero trust approach helped reduce cost

The average cost of a breach was USD 1.76 million less at organizations with a mature zero trust approach, compared to organizations without zero trust.

Cloud migration impacted costs and containment

Organizations further along in their cloud modernization strategy contained the breach on average 77 days faster than those in the early stage of their modernization journey.

OTHER TOOLS

Tool Name	How It Works	Key Drawbacks
Thycotic Secret Server	It is a shared password manager that allows teams to centralize their password manager.	- Requires an admin account with password to run as a high privilege user to rotate the passwords and keys.
Hashicorp Vault	Hashicorp vault is a vault service that allows you to store and create secrets for your endpoints. It also has an SSH CA feature that allows you to create SSH certificates.	- While vault offers SSH Certificates that is the same tech that we use, the process for the user is still manual (they must go to vault, create the certificate and then install it on their PCs). - Vault also lacks the advance access management that EZSSH offers.
Key Factor	Key factor allows companies to centralize their SSH key management into one portal.	- Requires admin privileges to manage SSH credentials. - While key management is centralized, input from administrators is required for lifecycle credentials.

KEYTOS

UNIQUE EZSSH FEATURES

- ✓ Designed for Zero Trust (No agent or high privilege account)
- ✓ Connection with Azure security tools: Networking JIT, Azure PIM, Sentinel
- ✓ Transparent security for users, with easy-to-use tools
- ✓ Reduce insider threat by having Just In Time Access with appropriate approval workflows.
- ✓ Reduce audit costs with easy to Audit access logs
- ✓ Centralized management for hybrid and multi-cloud environments
- ✓ Native Linux Authentication no custom PAM module or code runs on your servers.
- ✓ Bring your own Certificate Authority support

KEYTOS