Ungilded Secrets: A New Paradigm for Key Security

By Tal Be'ery

Published: 2021-01-25 · Archived: 2026-04-05 18:44:57 UTC

TLDR: SUNBURST attackers recently used stolen private keys to compromise several high profile targets in the US. To significantly reduce the chances of this happening again, we suggest a radical approach based on the cryptographic technology and security architecture that power our cryptocurrency wallet.

Russian Intelligence hackers recently pulled off a major cyberattack on the US. What is now known as the SUNBURST attack compromised the computer networks of several US government agencies and leading IT and security companies, including Microsoft and FireEye.

As authorities come to terms with the attack and learn of its true extent, most of the attention continues to focus on the initial access vector, a supply chain attack on a leading IT vendor. However, we believe the attackers' modus operandi of abusing golden secrets also needs to be scrutinized.

Golden secrets are at the heart of most current authentication systems. These long-lasting secrets are used to cryptographically secure the issuance of shorter-term access tokens and protect their integrity. Consequently, they are also the most lucrative target for attackers. When a golden secret is captured, it allows attackers to issue golden access tokens in an offline manner to take full control over their victims' environment.

In this post, we'll present a novel solution based on the same advanced cryptographic technology and security architecture used in our cryptocurrency wallet, which we believe can substantially improve organizations' ability to defend against such attacks.

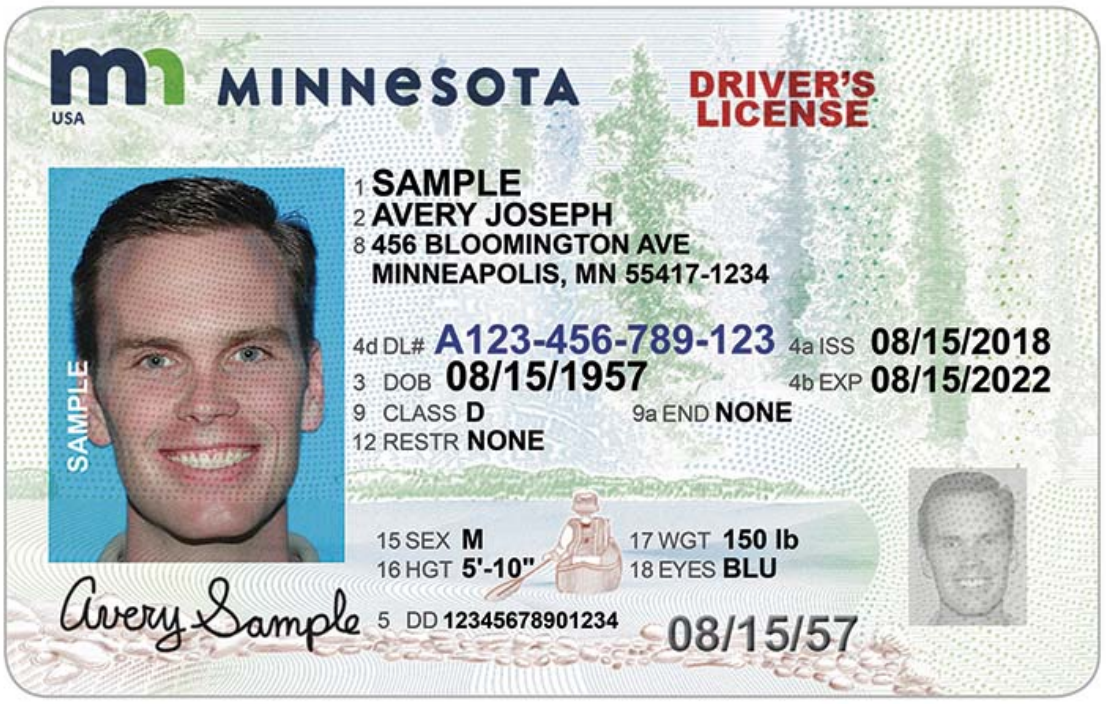
SAML in SUNBURST

The SUNBURST attackers were initially able to enter their victims' environments by infecting the software updates of a widely used IT product called SolarWinds Orion (which is why this attack is named "SUNBURST").

Once attackers were inside, their goal was to gain access to the authentication system's golden secrets, mainly (but not limited to) the SAML private key.

SAML tokens can be likened to a permit, such as a driver's license. When someone wants to obtain a driver's license, they must visit the Department of Motor Vehicles (DMV) **once** to present the relevant papers (e.g., photo ID and driving test results) and get a permit.

The permit details the individual's identity and contains authorizations (i.e., to drive a certain type of vehicle). This permit can be verified by **multiple** other parties (e.g., policemen) by checking the security measures embodied within.



Driver license: contains driver's authentication (name) and authorization (class D) information; security measures embodied within the permit. (source: mn.gov)

Similarly, SAML tokens are issued by Identity Providers (IdP) within an organization (e.g., Microsoft ADFS). Users authenticate **once** (in a while) to the IdP by showing their credentials: usernames, passwords, and Multi-Factor Authentication (MFA) and get a SAML token in return, signed by the IdP private key.

The SAML token details the person's identity and authorization level (e.g., admin or regular user). When users wish to connect to **multiple** Service Providers (SPs), either on cloud or on-premises, they present their SAML token to the relevant SP. The SP verifies the token by validating the signature using the public key.

One critical difference between a SAML token and a driver's license is that a SAML token is not a "photo ID." There is no way for the SPs to make sure the holder is the same person as the SAML token claims. SPs can only verify the token is signed correctly.

By stealing the SAML private key, the attackers could create their own tokens **offline** (i.e., without interacting with any of the victims' systems, thus leaving no traces in audit logs). **They could also masquerade as any user with different access privileges and totally bypass Multi-Factor Authentication (MFA), thus gaining unlimited access to all systems.**

What's more, even when victims' were able to supposedly clean their environment by removing the malware and resetting passwords (but not the SAML private key), attackers could keep returning and accessing the victims' services with their forged SAML tokens.

The knee-jerk reaction: build another wall around private Keys

The automatic reaction when something we own is stolen is to add more layers of defense. This was precisely how the Cybersecurity and Infrastructure Security Agency (CISA) reacted when they advised victims' to store their

private keys in a Hardware Security Module (HSM).

″Strongly consider deploying a FIPS validated Hardware Security Module (HSM) to store on-premises token signing certificate private keys. An HSM, aggressively updated, makes it very difficult for actors who have compromised the system to steal the private keys and use them outside of the network″

But inherent weaknesses exist within this HSM recommendation. For a start, hardware solutions are much more complicated and expensive to deploy compared to software solutions. This meant CISA could only “strongly recommend” this action and not mandate it as they did in other situations.

Additionally, HSM solutions need to be “aggressively updated,” as HSM vulnerabilities can allow attackers to gain access to private keys and steal them as before.

Given these limitations and the attackers’ demonstrated capability to penetrate highly protected systems, adding additional security layers within the infected environment will likely prove ineffective. More radical thinking is required to solve this problem.

What if we didn’t need PRIVATE KEYS?

System administrators don’t require the private key itself. They just need to get SAML tokens signed. Instead of protecting private keys within an infected environment, what if keys are stored in another domain to which the victims have no access?

Having an external service that signs tokens but keeps the private key out of reach of system admins and potential hackers can be a step forward. However, this solution is far from perfect.

The external signing service must be trusted to keep private keys safe from theft and ensure only valid requests from the customers’ are signed and not from attackers.

The good news is we can use Secure Multi-Party Computation (MPC) and specifically Threshold Signatures (TSS) to solve this issue.

Using these state of the art cryptographic constructions, we can “split the atom” and create a private key in a distributed manner. Half the key is created in the customers’ organization environment, while the other half is created in the external service environment.

Both parties must engage in a protocol that signs the token without revealing their part of the key to other parties to sign. The signature itself is the same as if it had been created with a single private key and validated with a public key. Therefore, there is no need to adjust existing solutions to support this architecture.

When a private key is distributed with TSS, attackers must compromise these two independent environments. As a result, the SAML private key is protected, even if the customer’s environment is compromised.

Additionally, a customer’s need to trust the external service is eliminated. Even if the service becomes entirely compromised or turns rogue, it cannot forge tokens independently.

The solution consists of two mandatory parts, TSS cryptography and the security architecture of an external service. It is the combination of these two parts that makes this solution effective.

Using TSS on its own and distributing the secret within the customer's environment without an external service is insufficient. This is because when a customer's environment is fully compromised, attackers can put their hands on all the distributed parties and extract the private key. On the other hand, using an external service to store keys without TSS is vulnerable to external service attacks, as described above.

It's important to note that these TSS and MPC algorithms are not just theoretical but actually very practical. In fact, we at Zengo have been successfully using them for the last two years in our cryptocurrency wallet to protect our customers' private keys – by splitting them between our servers and customers' devices. We've made our implementation code publicly available for the benefit of the community.

Applying TSS to SAML is even easier than for cryptocurrency, as availability is not a problem. In cryptocurrency, users' funds are bounded to a specific private key, and therefore losing even a single part of that distributed key can lead to loss of funds. As a result, we needed to develop dedicated solutions to prevent this from happening.

This is not a problem for SAML. If a key, or some part of it, is lost, a new one can be issued, as unlike cryptocurrency, the specific private key has no special meaning. Additionally, older tokens do not become obsolete as they can still be verified with the older public key, even when the private key is no longer available.

Of course, while TSS with external service for SAML signing is a significant step forward, it's not a panacea. Attackers can still generate requests to sign tokens online from the customers' environment to the external signing service.

However, this is restrictive and requires attackers to work against the system.

- Attackers cannot persistently return to the environment after they've been kicked out.
- Online attacks leave traces in logs, making the attack much more visible. In fact, splitting the signing between two environments adds more security, as there can be an audit log for both environments. Attackers need to compromise both to tamper them.

Attackers may still add an additional pair of a private and a public key as further SAML authenticators to the environment (a la Skeleton Key attack). But this attack must be carried out in an online manner that's highly visible.

Parting thoughts

By abusing golden secrets, attackers were able to take full control over the networks of some of the world's most protected organizations. This clearly indicates our current defense methods are insufficient and that new thinking is required.

In this post, we presented a radical but practical solution that can significantly reduce the chances of an attack like SUNBURST happening again.

Instead of trying to keep golden secrets safe by adding additional protective security layers, organizations should strive not to keep them exclusively in the first place.

Securely distributing the responsibility for these secrets to external services using Threshold Signatures is simply a no-brainer. But it’s just the tip of the iceberg when it comes to how modern cryptography can significantly upgrade the security of organizations worldwide.
