Where is the Origin QAKBOT Uses Valid Code Signing

By By: Hitomi Kimura Oct 27, 2022 Read time: 10 min (2662 words)

Published: 2022-10-27 · Archived: 2026-04-05 15:09:58 UTC

A threat actor, QAKBOT, along with EMOTET, has been one of the most active threat actors over the past few years, with numerous reports regarding its actions since it was first observed in 2007. We have reported some of them in the past, however, there are two things that come to mind regarding this threat. Namely, how Black Basta ransomware operators have used QAKBOT as a means of entry, and how they used the vulnerability, CVE-2022-30190, called Follina.

We have recently observed their use of modules with valid code signing by valid certificates that were issued from public trusted certificate authorities. This indicates that the threat actor has the private key of the certain valid certificate that was used for the code signing.

For further information, here is one of the infection timelines that we have observed:

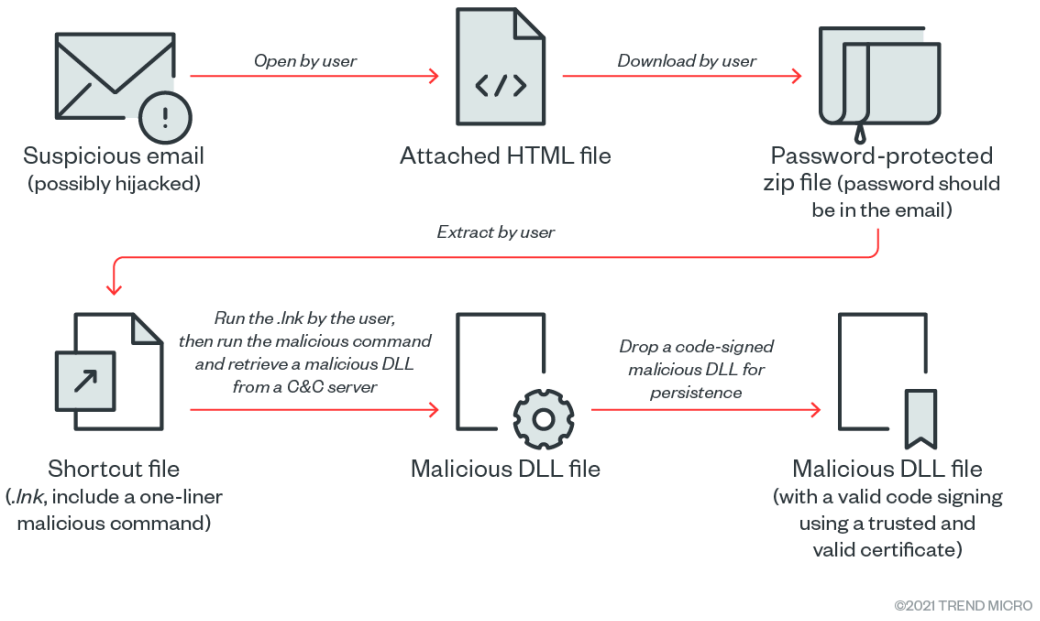

Figure 1. Infection Timeline

The following sections will discuss what we have found and how to counter this threat actor's method, which was actively observed during June-July 2022.

Key findings

Checking the modules related to QAKBOT shows multiple samples that have been signed with multiple valid code signing certificates. A look at the abused certificates also reveals that they were not issued to non-existent organizations for abuse, but rather valid certificates issued to real existent organizations through proper process.

This is not the first time we have observed the abuse of valid code signing certificates but is still uncommon and noteworthy that a single threat actor continues to obtain multiple valid code signing certificates and their private keys intermittently.

The origin of the certificate and private key is currently unknown, but there are at least two possibilities:

1. Using a traditional approach, which involves dumping and stealing private keys from infected computers
    - QAKBOT was interested in gathering data related to certificates as mentioned in our past research, and it had the capabilities to do so, allowing the threat actors to steal them if the victim does not use hardware tokens for key generation in the certificate issuance steps. Also, they sometimes use Mimikatz in their operation. It has built-in functions for dumping certificates and private keys.
    - However, it remains unclear if they were active in private key dumping in their operation, since the threat actors may have received the certificates directly through abuse of the certificate issuance process rather than through private key dumps.
2. Abusing the certificate issuance process, including possible identity theft
    - The certificates being abused were issued to micro companies, including some that are unrelated to technology like farmers.
    - We also found some unusual details in the information contained in the abuse certificates:
        - The same free email service was used with different certificates.
        - The domain name of the email address was assigned a new IP address right before the certificate was issued.

Code signing is used to build trust, but security teams need to be aware that an abuse of trust should be observed. Since a valid code signing is not enough to determine that a module is safe, it might be needed to check the certificates one step deeper, with the expectation that a malware may have a valid code signing.

Revisiting the history of code signing malware and how they activate

We have most recently published a report on a case in which a legitimate anti-cheat driver with a valid code signing was abused to bypass privileges. There have been other cases that involve the abuse of code signing certificates and private keys in the past.

One of the earliest high-profile cases would be back in 2010 in the form of Stuxnet, which used valid code signing certificates and private keys stolen from several real existing companies.

The Flameopen on a new tab malware is another notable case in which fakes were crafted used hash collision. In that case, an MD5 hash collision crafted one of the most trusted certificates, a Microsoft code signing certificate, which then was used to sign the APT malware. This technique was very sophisticated but could be identified because the details of the certificate differed from the original.

In addition to what was mentioned above, our research back in 2018 covered other ways to obtain valid trusted certificates. The research found that threat actors could mimic a legitimate organization to obtain a certificated issued by a trusted certificate authority (CA), and those certificates and private keys are sold on the black market.

Finally, a research by Kim Doowon of the University of Maryland, presented at the Conference on Computer and Communications Security Conference (CSS'17), "Certificate Malware: Measuring Breaches of Trust in the Windows Code-Signing PKIopen on a new tab," found 72 certificates with stolen private keys, 22 certificates issued by identity theft from legitimate companies, and five certificates issued to shell companies at that time. The report identified 188,421 malware cases and analyzed 14,221 code-signed malware cases from their dataset: 10+ positive samples that had been submitted to VirusTotal from at least the period including StuxNet – 2010 through 2017, excluding PUAs.

The number of abused certificates, which amounted to 72 in seven years, may seem fewer than expected. However, it shows that the QAKbot threat actor is even more actively retrieving certificates and private keys, as we have observed at least seven certificates in use by QAKbot in just two months (June to July 2022) when this activity was most active.

On [the MITRE websiteopen on a new tab](), we can review past operations where code signing abuse has been observed. We can also see a number of cases of APTs using specific few certificates. QAKbot was able to obtain access to multiple legitimate certificates on-and-off for several years.

The kinds of certificates QAKbot use

Looking at the sample observed in the timeline at the beginning of this article, it seems to be a micro-company that exists in the Czech Republic. The email address included in the certificate used a free email service. There was another abused certificate that used this free email service.

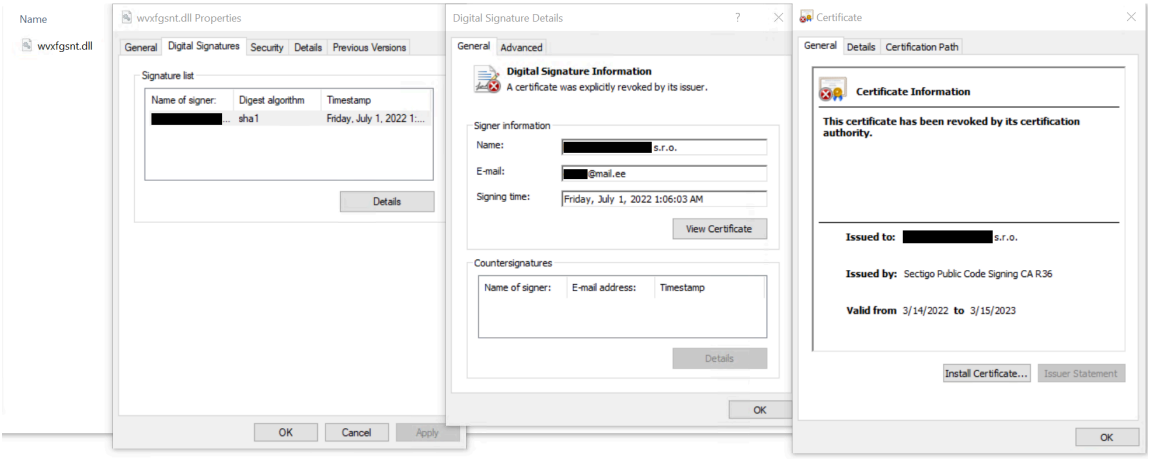

Figure 2. An example of an abused certificate that used a free email service

One of the peculiar details is the way the domain name would be assigned a new IP address the same time the certificate was issued. At the same time, the domain name does not host any contents related to the company. Those can be indicators that make us suspect that the certificate was issued through an unusual procedure.

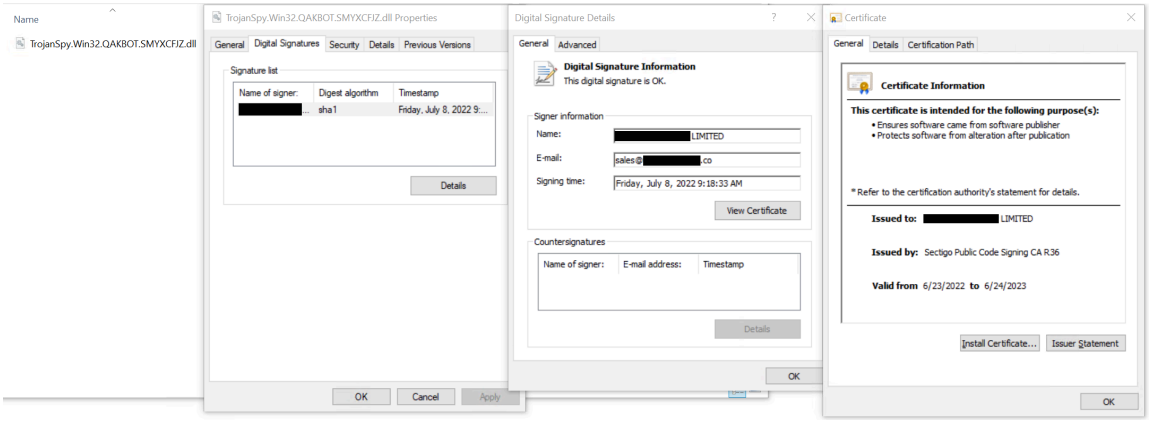

Figure 3. An example of an abused certificate using an email address with a domain name similar to their company name, but with a “co” top-level domain (TLD)

It is not a single issuer problem. A certificate for a company registered in the UK as a micro-company from an issuer other than those listed is also abused.

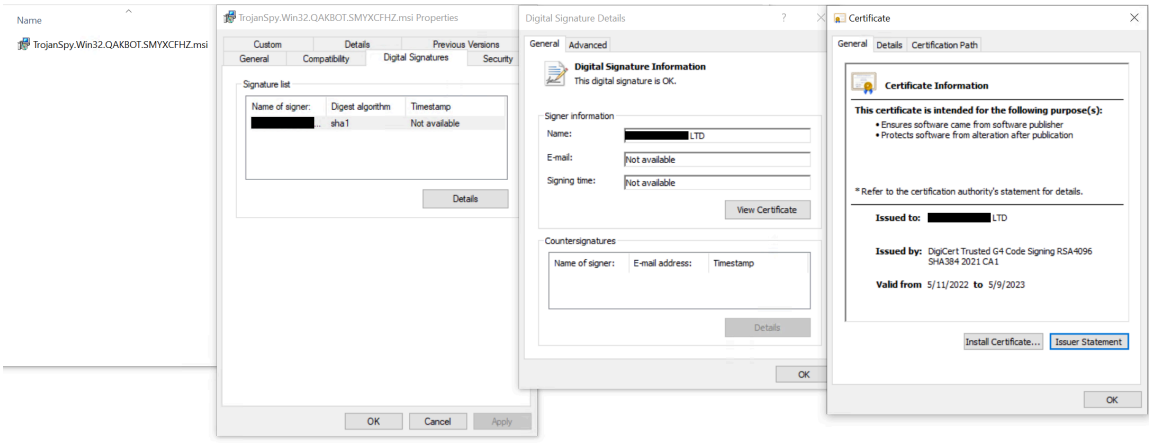

Figure 4. An example of an abused certificate issued by another certificate authority

Origins of the abused certificates

The origins of these certificates are currently unknown, but we can suspect two possibilities:

QAKbot has a function for enumerating and dumping certificates and private keys, thus having the capability to steal them if the victim does not use hardware tokens for key generation in the certificate issuance steps. Our analysis shows that they are using APIs like *PFXExportCertStore()*, which is used for dumping private keys.

```
CertAddCertificateContextToStore_0(v32, CERT_CONTEXT, 3, 0);
CertAddCertificateContextToStore(v32, CERT_CONTEXT, 3, 0);
CertAddCertificateContextToStore_1(v32, CERT_CONTEXT, 3, 0);
v28 = 0;
if ( PFXExportCertStore(v32, &v27, a3, EXPORT_PRIVATE_KEYS_|REPORT_NOT_ABLE_TO_EXPORT_PRIVATE_KEY) )
{
  v27 = 0;
  v28 = 0;
  PFXExportCertStore(v32, &v27, a3, EXPORT_PRIVATE_KEYS_|REPORT_NOT_ABLE_TO_EXPORT_PRIVATE_KEY);
  v28 = HeapAlloc_1F10D0(v27);
  if ( !v28 || !PFXExportCertStore(v32, &v27, a3, EXPORT_PRIVATE_KEYS_|REPORT_NOT_ABLE_TO_EXPORT_PRIVATE_KEY) )
    return CertFreeCertificateContext(CERT_CONTEXT);
  if ( !fn_1E320E )
    goto LABEL_31;
  fn_1E320E(v28, v27, v30, v31);
}
else
{
  fn_1E320E(0, 0, v30, v31);
}
```

Figure 5. QAKbot calls PFXExportCertStore() for dumping private keys

Cobalt Strike, which they sometimes use for their operations, includes a built-in Mimikatz, which provides certificate and private key dumping capabilities. However, it still remains unclear if they were actually active for private key dumping in their operation.

The certificates being abused were then issued to micro-companies, including some unrelated to technology. The possibility of the abuse of certificate issuance process could include possible identity theft. From the observed abused certificates, it seems as if the attacker is impersonating a legitimate organization and is directly issued certificates by public trusted CAs.

Another certificate that was issued to a company registered to farmers was also observed. Although it is not unthinkable for them to have been issued the certificate for their business, it is also possible that the threat actor stole their identity, used it, and/or abused the issuance process to be directly issued the valid certificates by the public trusted CAs.

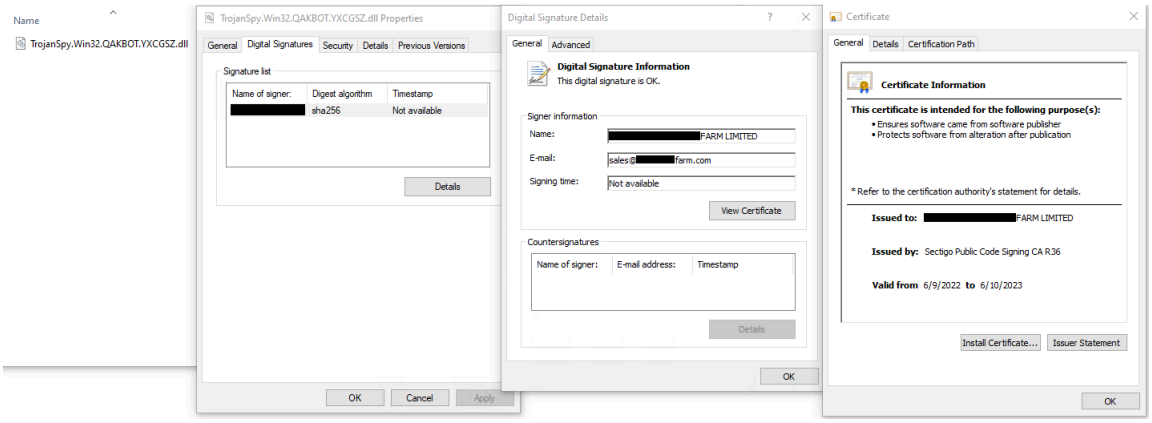

Figure 6. An example of an abused certificate issued for a farmer's company

Protecting users' private keys

In this case, the assumption that the threat actor was directly issued certificates through abuse of the certificate issuance process is more strongly suspected than the stealing of the private key, but the protection of private keys on the user side is still a challenge.

In the use of code signing certificates, private key protection on the user side has been enhanced over time, but it still has a long way to go before it can be classified as foolproof.

As mentioned above, we can see in QAKbot a function to retrieve victim's private keys using the *PFXExportCertStore()* API, which can only export private keys if they are stored in the Windows Certificate Store with an exportable flag. In other words, if key generation was performed using a hardware token that meets certain criteria (IC card, HSM, TPM, etc.), the private key is not stored in the Windows Certificate Store, so no method is provided to export and reuse the private key since it is stored within the token.

There is a document, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Code Signing Certificates(v.3.1)open on a new tab" that defines things related to code signing certificates. This document is followed by CAs that issue code signing certificates and require CAs to encourage users to use tokens that meet the criteria to protect their private keys. According to this, prior to November 15, 2022, CAs must recommend to users that they protect their private keys with hardware tokens that meet the criteria and must obtain a representation from the user that they will protect their private key by using certified hard tokens or at least in a way to keep the key physically separate from computers until a signing session can begin.

So far, we have confirmed that many companies that sell certificates for code signing issue them in a way that the key is stored on a smart card or other hardware device basically in which case the private key cannot be exported remotely.

Even though everyone knows that hardware key generation provides stronger protection, software key generation that allows easy backup in the type of PKCS#12 files without the use of hardware tokens has remained popular on the user side. Some certificate issuers continue to provide this method as an option by to this day. As the last mile of protection, if the user actually imports it into the token and keeps it offline, it is left to the user themselves.

The scenario in which active abuses by threat actors are continuously observed is required to rethink the key generation and certificate issuance methods. The baseline requirement will have stronger requirements for private key protection starting November 15, 2022. According to the Baseline Requirements, the CAs "shall" ensure the user's private key generation with

a suitable hardware after the date. While this is a step forward compared to the scenario in which the last mile is left to the user, the day when private key storage in software is no longer common may still be a little further away.

How is WebPKI doing?

In the previous sections, we mentioned the possibility of abuse of the certificate issuance process, which includes identity theft. The problem that a threat actor may impersonate a real organization then being issued a certificate directly can also happen in WebPKI, which gives TLS certificates used for HTTPS communications.

WebPKI has introduced Certificate Transparency(CT) since around 2015, which means that the issuance of all public trusted server certificates used for HTTPS is logged in the CT logs in public. This allows the CT logs to be verified by anyone from the internet and helps detect the issuance of certificates that the domain name owner is unaware of.

Thus, while WebPKI can use the CT logs to check for unexpected issues of certificates for organizations, however, code signing certificates cannot be checked in the same way because the CT mechanism has not been introduced for code signing certificates.

Regarding certificate revocation

To counter the abuse of certificates that have already been issued, the one of the options is to revoke the certificates.

According to research made by Doowon Kim and Bum Jun Kwon in 2018 titled "The Broken Shield: Measuring Revocation Effectiveness in the Windows Code-Signing PKIopen on a new tab," even if a code signing certificate is revoked, the signature may remain valid due to the specifics of the revocation and verification method, which is a mechanistic challenge.

All certificates observed to have been abused in this case have been revoked at this time.

How to counter abused code signing certificates

For users:

First, protect your private keys. The people who perform code signing store their private keys for code signing certificates using modern best practices. The private key is stored with the appropriate hard token and stays offline until just before the signing process begins. Modern practice also provides a service to store private keys and sign in the cloud.

Second, protect your identity. While the origin of the abused certificate is currently unknown, victims may have been impersonated in the name of the company completely unknown to them, and the certificates issued to them may have been abused. Thus, it would be imperative if we could carefully monitor for strange things happening around us related to identity theft. It will take some time for things to unravel but protecting yourself while the certificate issuers figure out why this is happening will be key to avoiding abused certificate issuance.

It is possible that the threat actor has acquired a domain name similar to the company name of a real company and is using the fact that they own the domain name to get a certificate issued. Unfortunately, there is currently no effective way to check this case since unlike WebPKI, CT has not been implemented for code signing certificates.

Detection and monitoring of abuse of trust

Security teams need to be aware that an abuse of trust has been observed. Since a valid code signing is not enough to determine that a module is benign, it might be needed to check things closer, to be assured that the one with a valid code signing is not a malware impersonating a legitimate module.

The certificates confirmed to have been abused in this case had the following attributes:

- **Had company names unrelated to technology**
- **Issued to micro-companies around the world**
- **Used a free email service for its email address**
- **Used an email address that contains a domain name that looks like a company name, but the domain name does not host any content**
- **The top-level domain of the domain name used is not common for enterprise, like "co"**
- **Assigned a new IP address just before the certificate was issued**

Fortunately, these certificates can be figured to be unusual if they are carefully checked.

Conclusion and Trend Micro solutions

QAKbot is trying to evolve into a much more intrusive malware. Its use of valid code signatures can make it hard to determine which files are real, and its persistence can cause further danger to machines and the network that it is connected to while the certificate issuers find a way to counteract what the trojan is doing.

Since QAKbot can originate from an Excel file, make sure that macros are disabled in Microsoft applications. In addition, check the sender and the format of the email address to prevent email spoofing. For further protection, install Trend Micro Deep Discoveryproducts, which contains an email inspection tool. For endpoints, Trend Micro Vision Oneproducts can detect possible persistence since QAKbot drops in via .DLL files.

With additional insights by Byron Gelera.

Indicators of Compromise

Trojan.Win32.QAKBOT.DRSO	adadda4d61188c53c25323a3561db52d14a5dbb2585a53e18b33882f1013b9ee
TrojanSpy.Win32.QAKBOT.YXCGSZ	78bc13074087f93fcc8f11ae013995f9a366b6943330c3d02f0b50c4ae96c8a7
TrojanSpy.Win32.QAKBOT.SMYXCFJZ	42bc9b623f70e46d6aab4910d8c75221aecf89a00756a61b21f952eea13a446c
TrojanSpy.Win32.QAKBOT.YXCCBZ	37e973699f119ce5a2047281aa6f52429bc15164abdfe110f3340ee02d4c21b5
TrojanSpy.Win32.QAKBOT.SMYXCFHZ	362e56855844fb2be3dfae4b566ab676f6ec681fad1c1a2e8eb6d245d56b83f5
TrojanSpy.Win32.QAKBOT.YXCCQZ	20839bc89ca241bcd77ea69a2e36e40d7c1bd0dd91952502de8cd1db6fe771e1
TrojanSpy.Win32.QAKBOT.SMYXCFJZ	1beb6f15f403fe31392012b506ce1bb38424482d3e8123f0f80ae439484fffe7

Tags
