CounterCraft

Product Security

Committed to Continuous Security Excellence

Specific. Actionable. Threat Intelligence Powered by Deception.

Table of contents

ISO27001 Certification

CounterCraft was first awarded with the ISO27001 Certification in 2021 with the following scope:

> *"Information Security Management System that supports the product lifecycle process including design, development, sales, installation and support within the scope of Cyber Security"*.

During 2024 CounterCraft maintained the ISO27001 certification and updated it to the latest revision of the standard (i.e. ISO/IEC 27001:2022).

Product Security Features

In an era where digital threats are ever-evolving, ensuring robust cybersecurity measures is paramount. CounterCraft's The Platform™ stands at the forefront of cyber defense, offering cutting-edge cyber deception and threat intelligence solutions tailored to protect organizations from sophisticated cyber threats. This document outlines the key security features of The Platform, demonstrating our commitment to safeguarding your digital assets.

Users

Users' passwords are hashed using the PBKDF2 algorithm with a SHA256 hash, a password stretching mechanism recommended by NIST. This should be sufficient for most users: it's highly secure, requiring massive amounts of computing time to break. There is a minimum password length requirement of 8 characters.

Users can also protect their accounts enabling two factor authentication (2FA).

Users must change their password on their first login.

Both user valid and invalid login attempts are stored with their related information: date, IP address and user agent.

Users are disabled after a configurable number of invalid login attempts and they can only be enabled by the admin user.

Users can configure their own session timeouts.

Authentication

JWT Access

The application uses a signed Json Web Token (JWT) that includes information about the session.

Cookies

All cookies used in the application use the 'secure' and 'httponly' attributes.

API Access

API access uses a combination of API key and secret key. All API requests are HMAC signed requests and they use a TLS connection.

Deception Director Web Access

Web access uses TLS by default, but it can be further protected by adding the need of using client TLS certificates. This security feature can be enabled during the installation process.

Integration secrets

Any sensitive data belonging to an integration instance is securely stored in a secret vault (Hashicorp's Vault). No sensitive data is stored in clear text in the database. For more information about Vault's security please visit https://www.vaultproject.io/.

Certificate Authorities and Certificates

The product uses two different Certificate Authorities (CA): internal and external. The internal CA is used for authentication purposes between different components, and the external one for authenticating the deception hosts with the deception support nodes (DSN). This authentication is checked on both sides (from the DSN to the deception host, and from the deception host to the DSN).

Both CAs are stored in a secured vault (Hashicorp's Vault). For more information about Vault's security please visit https://www.vaultproject.io/.

Roles and permissions

There is a robust product role and permission model. Roles and permissions are explained in different sections in this User's manual. Please review the relevant sections.

Component updates

All the third-party components are updated and checked for vulnerabilities in each product release. For more information about third-party components please visit the Third Party Open Source Software section.

Containers security and hardening

We try to minimize the existence of vulnerabilities in our containers by using Alpine as base images and installing only the needed packages. We periodically check all the images and update them accordingly while containers always use a non-root user. The containers run in a private network and only the minimum ports are exposed to the outside.

LogBook and Audit Log

In order to log all user actions, there are two main product features: Campaign's LogBook and Audit logs. The Campaign's LogBook is designed to track who, when and what happened in a campaign configuration. On the other hand, the Audit log will track all changes to the database (only Administrators will be able to access the Audit Log).

Installation and upgrades

CounterCraft release artifacts are signed using GnuPG and our release signing key. Installations and upgrade processes are saved in a local file. The upgrade process can export the configuration and backup the database before making any changes. All external configuration changes (during the installation or the upgrade) are tracked using Git.

Network

Between the Deception Hosts and the Deception Director there is a component called Deception Support Node that takes care of gathering and forwarding telemetry logs and files from the DHs to the DD and commands from DD to the DHs. There is no direct connection from DHs to the Deception Director. Connections from the DD and the DHs to the DSNs are always inbound connections and the Deception Director is always set to never accept connections initiated from the networks in which the DSNs are located. All the information (logs, files) are pushed from the agents to the DSN and the Deception Director pulls it from there.

All the network services are written in Go, a programming language whose runtime has a very good track record with regard to vulnerabilities. It's a safe programming language by design.[1] The services are designed to only allow data to flow from the DHs to the DSNs and from the DSNs to the DD and never the other way around unless the user runs an action to send commands or files over. The DH never asks for information from the DSN. The DSN only asks for public keys to the DD in order to authenticate tokens.

The agents communicate with the DSN using TLS 1.3. Clients must authenticate either with client certificates or signed and encrypted tokens. Clients check that the server certificates are signed by the correct CA and with the correct CN. All the traffic is encrypted. There are no known vulnerabilities with TLS 1.3.[2] It provides forward secrecy for all connections. This means that, even if you have captured all the traffic, you won't be able to read it even if they get the private key.[3] All the traffic from the ServerHello message onwards is encrypted, that means an outside observer can't see any client certificate details.[4]

The communication between the Deception Director and the DSN is done via SSH. All the traffic is encrypted.[5] It provides forward secrecy for all connections. All traffic after the SSH server hello is encrypted.

Log flow

[1] Go vulnerability statistics
[2] Why use TLS 1.3
[3] The importance of forward secrecy TLS 1.3
[4] RFC 8446 - Major Differences from TLS 1.2
[5] SSH Protocol

The logs are serialised and deserialised safely and we check whether they are valid before forwarding to prevent attackers from injecting data that could cause issues down the line.

If an attacker tries to perform a DoS by generating a lot of events we have two measures in place to avoid it impacting the processing of other DH's logs:

1. If an event flood is detected, events are aggregated.
2. Logs are partitioned, a spike of logs in one DH doesn't affect the other ones.

The amount of resources each DH's logs takes in the DSN is limited, a DoS attempt wouldn't make the DSN go down by running out of memory or storage.

Regarding Logbook and Audit Logs there is no retention policy because we preserve the whole history (as those logs track user interaction with the product the amount of activity logged is not expected to grow over a few thousand entries).

Signing Keys

CounterCraft releases artifacts that are signed using GnuPG and our release signing key.

Before signatures can be verified, CounterCraft signing key must be downloaded. The key can be obtained from us, from keys.openpgp.org or from the SKS keyservers pool. The former method is recommended to avoid issues as most key servers are prone to overload, abuse and attacks:

```
curl -L https://ccservices.countercraftsec.com/static/countercraft-signing-key.asc --output countercraft-signing-key.asc
gpg --import countercraft-signing-key.asc
```

Verifying Signatures

To check signatures for the packages, download the CounterCraft signing key and a signature file. Signature files use the .asc extension that follows their artifact filename, e.g. the signature file of console-user-guide-2.9.14.zip.sha256 would be console-user-guide-2.9.14.zip.sha256.asc.

Then use gpg –verify:

```
gpg --verify [filename].asc [filename]
```

Here’s an example session, after having retrieved a CounterCraft release artifact and its associated detached signature from the download area:

```
gpg: assuming signed data in 'console-dist-2.9.14.zip.sha256'
gpg: Signature made Mon May  3 17:37:21 2021 CEST
```

```
gpg: using RSA key 06FBE614CC5F765BB7AD5FB4D63C6E3E8DB99621

gpg: Good signature from "CounterCraft Releases <support@countercraftsec.com>"
[ultimate]
```

If the signature is invalid, a “BAD signature” message will be emitted. If that’s the case, the origin of the package, the signature file and the signing key should be carefully verified. Packages that fail signature verification must not be used.

If the signature is valid, you should expect a “Good signature” message; if you’ve not signed our key, you will see a “Good signature” message along with a warning about our key being untrusted.

If you trust the CounterCraft signing key you avoid the warning output by GnuPG by signing it using your own key (to create your private key run gpg –gen-key):

```
gpg --sign-key 5057EF4CDD3EE429BDB43F7E3C187D4C2B6BB949
```

You can now verify its checksum using sha256sum:

```
sha256sum -c console-user-guide-2.9.14.zip.sha256

console-user-guide-2.9.14.zip: OK
```

Preventing cross-site scripting (XSS)

In order to avoid XSS vulnerabilities, the web app that loads in the browser (Director Console) handles all values as untrusted by default. When a value is inserted into the DOM from a template binding, or interpolation, the underlying webapp framework (Angular) sanitizes and escapes untrusted values. For further information visit https://angular.io/guide/security

CounterCraft’s product also includes different software mitigations against XSS and other similar vulnerabilities:

- Use of Content Security Policy (CSP) to mitigate certain types of attacks, including XSS and data injection attacks.
- Use of HTTP headers like ‘Strict-Transport-Security’ and some others to force the use of HTTPS.

Lazy loaded resources

Besides benefits for performance optimization, The Platform console downloads resources on premise depending on the user role. The app only downloads smaller chunks of the application to minimize data exposure and reduce attack surface. For example, if the user never performs a successful login, the rest of the application is never downloaded.

Product Security Audits

In today's rapidly evolving threat landscape, maintaining robust cybersecurity requires more than just static defenses. At CounterCraft, we understand the critical importance of proactive measures to ensure our products remain impenetrable. That’s why we regularly conduct rigorous penetration testing on our product (on every major product release), identifying and addressing vulnerabilities before they can be exploited by malicious actors.

Latest Penetration Testing

On 6 May, 2024, S2 Grupo carried out the security analysis on CounterCraft The Platform v3.5.0, within the framework of the project S24-0610.

Vulnerability analysis of the Deception Director Web Application and API, using different user roles.

The actions and tasks of the pentest service were carried out following internationally recognized OSSTMM[6] security audit methodology, along with the web audit methodology dictated by OWASP.

With respect to the first one, it defines a set of guidelines on which aspects should be audited according to the area of study (physical security, spectrum security and communications security) and security metrics that allow determining the level of risk and making comparisons over time.

Regarding the OWASP methodology, the audit team reviews, inter alia, the 10 most important types of security vulnerabilities according to the associated risk posed to a web service.

[6] http://www.isecom.org/research/

Secure Development and Continuous Integration

CounterCraft uses an agile development methodology where security is one of the main pillars of the methodology. Some of the security strategies include:

- Security tests (permissions, roles, etc.)
- Detection of security vulnerabilities in the code during the Continuous Integration process
- Detection of security vulnerabilities in the Docker containers during the Continuous Integration process
- Automated creation of a SBOM[7]

[7] https://www.cisa.gov/sbom

www.countercraftsec.com