Canary - Operation and Security Technology

The security of our Canary system largely relies on three factors:

- Choice of sane technologies;
- Applying the principle of least privilege and;
- Defensive coding techniques.

Security Technology

Canary services are written in Python, a language with built-in memory management, so memory corruption bugs aren't an issue from the start. The core is written using the Twisted framework, a well-tested and widespread tool for writing Python network applications. Each service is intentionally low-interaction to reduce the exposed code. For example, the SSH service reports login attempts but will never allow logins to succeed; knowing that an attacker is trying to login is sufficient to trigger an investigation without tracking what they're doing.

Console data is stored in a NoSQL database (Redis), dramatically limiting the potential for query injection bugs. The Redis access library removes the possibility of injection bugs by design, and we don't use two Redis features that potentially could introduce injection bugs.

For cryptography we've relied on Dan Bernstein's NaCl library which provides authenticated asymmetric and symmetric encryption. We do not use any custom cryptography at all.

The web UIs are built with a modern web framework (Flask), which handles escaping automatically.

Data transmitted between Canaries and its console is always encrypted, using keys agreed on at registration, which is also done over encrypted channels. Once registration has been successfully completed, the Canary device communicates with its console using encrypted DNS packets.

Key management has been carefully crafted, and multiple keys exist for different actions rather than reusing a single key.

We run a common Linux distribution so updates are forthcoming and aren’t reliant on us creating our own packages for common services. Unneeded packages have been stripped.

Principle of least privilege

In terms of least privilege, none of our services run as the superuser. Each has its own regular user.

There is a separate configuration and production mode. The Canary can only be configured when running in config mode, which requires pressing a physical button. This means that when the Canary is running in production, the configuration state cannot be altered. No configuration changes or updates can be pushed down to the Canary devices from the Console.

Defensive Techniques

We purposely limit the exposure of the Canaries. No real information is stored on them. A compromised Canary yields the ability to transmit events to your console, but does not provide information about other Canaries on the network. We think of them as moderately dumb sensors, reporting events to a console which can correlate.

The events captured by Canaries are pre-determined and have a high signal-to-noise ratio. They don't report every packet seen, and never report entire packet contents. For each event, only data of interest is sent to the console (e.g. credentials used in a brute-force).

Consoles treat Canaries with distrust. Each new Canary registration must be confirmed at the console before the Canary can report new events.

Each console is an isolated instance, there is no sharing of instances between different users. Consoles do not expose SSH, access is via a jump-box. Consoles cannot access each other's private services (e.g. can't SSH from one console to another). Data is stored on the same instance as the console web server, there is no network access to the data. This means access to the console is a prerequisite for reading its data.

END