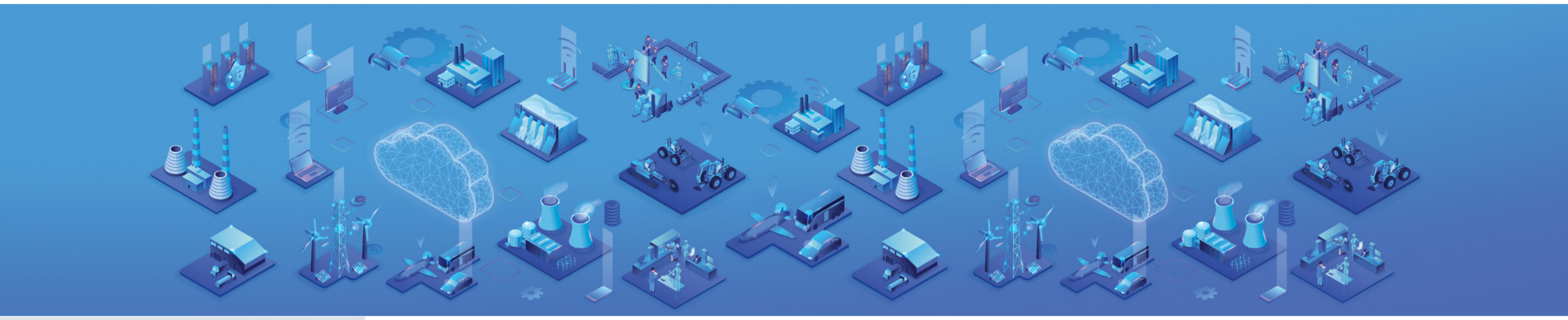


Dellfer's DICE – Rethinking Cybersecurity

HOW WE ARRIVED HERE

In the span of four decades, conventional firmware development has reached a tipping point, wrestling with outdated tools in the fight against the increase in cyber threats.

WHY IS IT BROKEN?

Depending on only static analyzers at the eleventh hour before product launch merely creates a perpetual loop of post-launch patches.

ARE YOU READY TO SHIFT

and get the upper hand against your attackers rather than merely defending and attempting to keep up?

It's essential to overturn the existing asymmetric advantage favoring attackers and transfer it to the defenders in cybersecurity. A strategy of ensuring protection and maintaining visibility of potential threats is the ideal course of action. The ubiquitous nature of interconnected devices can be harnessed as a vital tool for identifying cyber criminals and providing a significant edge in thwarting firmware attacks. When such a device functions as a threat sensor, attacks no longer remain undetected. This allows the defenders to identify and neutralize continuous threats within their networks, leading to the loss of the greatly valued Zero-Day vulnerability.

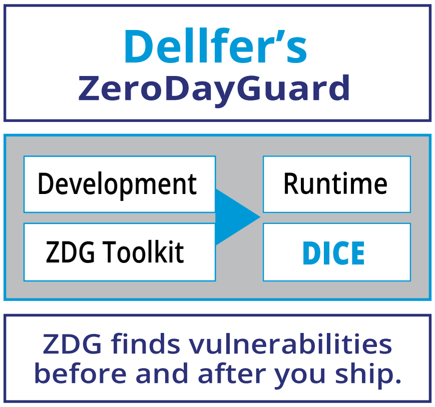
WHAT IS DICE?

Dellfer Incident Collection Engine (DICE) is a framework that collects and reports data in real-time from the ZeroDayGuard Forensic Collection module embedded on your target device. DICE is implemented using a containerized architecture, has a REST API and runs in a public cloud hosted by Dellfer.

Dellfer's ZeroDayGuard is innovatively shifting the balance of power away from cyber attackers in two steps:

1 Development: Eliminate harmful code by integrating ZDG into your source code and harden your runtimes at the highest level.

2 Runtime: Establish a real-time, active monitoring portal that preemptively seeks out potential threats. Transform your network of devices into an advanced identification system of cybercriminals, providing you a significant advantage in fending off firmware attacks.



Dellfer's ZeroDayGuard

Development → Runtime

ZDG Toolkit → DICE

ZDG finds vulnerabilities before and after you ship.

VSOC CORNER

Cybersecurity Technician

"Am I to believe that, because you are actually embedded in the source code and are monitoring the device and application for any abnormal behavior that you deliver no false positives and report all known and unknown Zero Day attacks?"

Answer

"Yes"

FROM CUSTOMERS

"We are excited to be working with Dellfer to protect our automotive electronic control units. As we continue to increase performance in the fields of connectivity and automated driving, solutions like ZeroDayGuard become essential for a vehicle's security against cyber attacks."

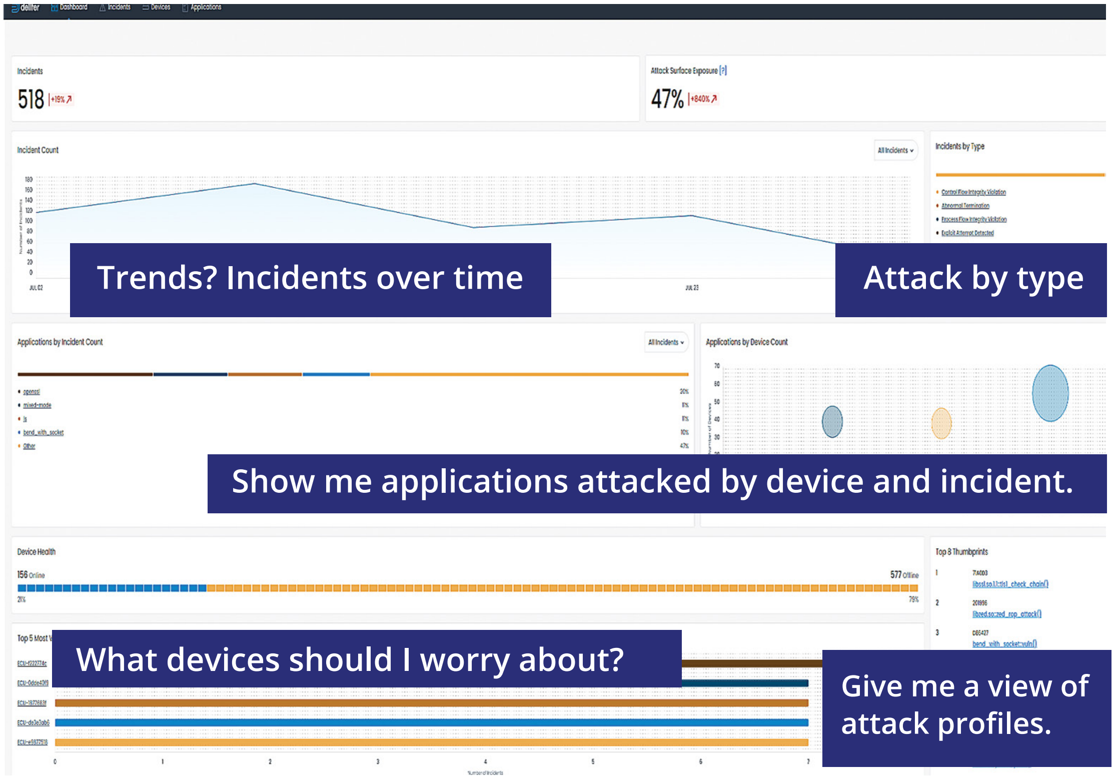
Tony Cannestra, DENSO

For a copy of a third party evaluation of Dellfer for DENSO visit us on the Web at: https://hellohub.dellfer.com/denso-technical-paper

dellfer

INFO@DELLFER.COM
WWW.DELLFER.COM

DICE Dashboard



Trends? Incidents over time

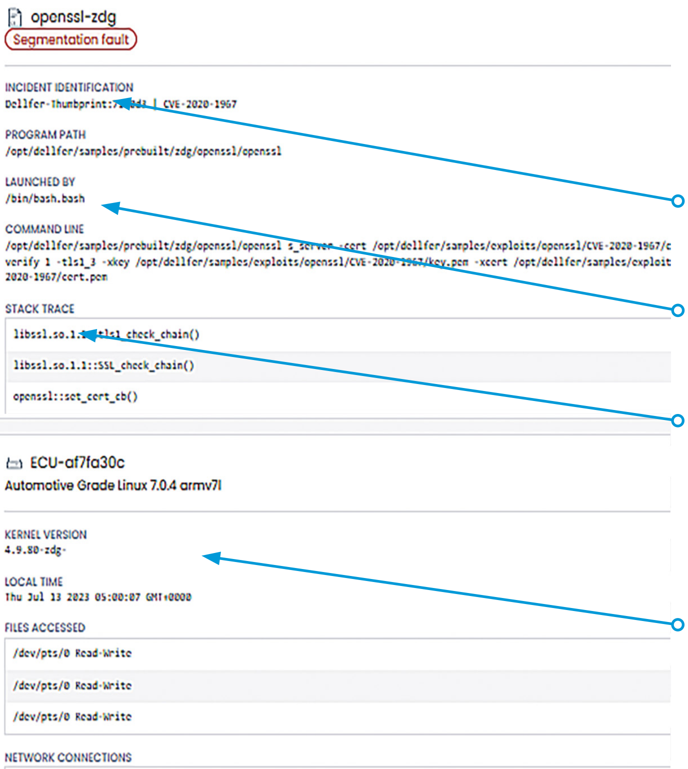
Attack by type

Show me applications attacked by device and incident.

What devices should I worry about?

Give me a view of attack profiles.

How DICE Hunts the Bad Guys



openssl-zdg
Segmentation fault

INCIDENT IDENTIFICATION
```
Dellfer-Thumbprint:7[illegible]3d3 | CVE-2020-1967
```

PROGRAM PATH
```
/opt/dellfer/samples/prebuilt/zdg/openssl/openssl
```

LAUNCHED BY
```
/bin/bash.bash
```

COMMAND LINE
```
/opt/dellfer/samples/prebuilt/zdg/openssl/openssl s_server -cert /opt/dellfer/samples/exploits/openssl/CVE-2020-1967/c
verify 1 -tls1_3 -xkey /opt/dellfer/samples/exploits/openssl/CVE-2020-1967/key.pem -xcert /opt/dellfer/samples/exploit
2020-1967/cert.pem
```

STACK TRACE
```
libssl.so.1.1::tls1_check_chain()
libssl.so.1.1::SSL_check_chain()
openssl::set_cert_cb()
```

ECU-af7fa30c
Automotive Grade Linux 7.0.4 armv7l

KERNEL VERSION
```
4.9.80-zdg-
```

LOCAL TIME
```
Thu Jul 13 2023 05:00:07 GMT+0000
```

FILES ACCESSED
```
/dev/pts/0 Read-Write
/dev/pts/0 Read-Write
/dev/pts/0 Read-Write
```

NETWORK CONNECTIONS

The Anatomy of an Attack.

This is an example of a crash caused by a bad hash ID in OpenSSL, resulting in a segmentation fault crashing the code.

A unique Dellfer thumbprint is created, noting any confirmed CVE information available.

Program path, parent process and the command line preceding the failed execution.

With a built-in stack trace , there is no delay in providing a fix. This is proactive security, and Dellfer delivers, allowing fixes to be made quickly because we know the exact line of offending code.

What the endpoint is and what files and sockets were open at the time of the attack Lastly, what was the IP or source of the attack was vulnerability or bug.