CrimsonIAS: Listening for an 3v1l User

By ThreatConnect

Published: 2021-01-27 · Archived: 2026-04-06 01:01:01 UTC

Executive Summary

CrimsonIAS is a Delphi-written backdoor dating back to at least 2017 that enables operators to run command line tools, exfiltrate files, and upload files to the infected machine. CrimsonIAS is notable as it listens for incoming connections only; making it different from typical Windows backdoors that beacons out. The characteristics found in CrimsonIAS's execution flow suggest a connection to Mustang Panda (aka BRONZE PRESIDENT, RedDelta) PlugX samples. Based on those non-unique characteristics, ThreatConnect assesses with low confidence that CrimsonIAS is an additional tool in Mustang Panda's repertoire. Industry reporting assesses with varying levels of confidence that Mustang Panda is a Chinese espionage actor that has conducted operations in Mongolia, Vietnam, and Hong Kong among other locations. According to fellow researchers, Mustang Panda targets non-government organizations (NGOs), law enforcement organizations, and political entities.

Discovery

ThreatConnect identified CrimsonIAS while hunting for XOR encrypted PlugX binaries. The CrimsonIAS backdoor is encrypted similarly to recent MustangPanda PlugX samples, which piqued our interest.

Encrypted SHA256: acfd58369c0a7dbc866ad4ca9cb0fe69d017587af88297f1eaf62a9a8b1b74b4

Decrypted SHA256: 891ece4c40a7bf31f414200c8c2c31192fd159c1316012724f3013bd0ab2a68e

CrimsonIAS Analysis

The developers behind CrimsonIAS wrote this backdoor using Delphi. They also added some features that changed the normal execution flow starting with shell code embedded in the MZ header. Windows executables are not designed to execute code from the MZ header; they have a dedicated section (outside of the MZ header) for executable code. The actor is able to work around this design choice to start execution in the MZ header based on how we suspect the binary is loaded. This shell code calls a reflective loader function which resolves additional library functions needed before jumping to the malware's actual entrypoint. The binary was also XOR encrypted with a 10 byte XOR key prepended to the binary (T1140: Deobfuscate/Decode Files or Information). All of these are also seen in Mustang Panda's PlugX samples.

This backdoor spins up a listener and awaits the operator's commands to run command line tools, exfiltrate files, and upload files to the infected machine. Prior to spinning up the network listener, CrimsonIAS launches netsh.exe to open a port on the local machine (T1562.004: Impair Defenses: Disable or Modify System Firewall). This sample opens port 80.

Command and Control

When receiving network traffic, the listener's handler first checks for the presence of the marker 0x33669966 (T1205: Traffic Signaling). If it matches, then the handler proceeds to parse the first 24 bytes.

The first 24 bytes make up the command header; however, only the first three DWORDs appear to be used.

The command buffer starts at offset 24 (0x18) and its content differs based upon the command code specified.

The three command codes are:

- 0x6600 : Run Command (T1059.003: Command and Scripting Interpreter: Windows Command Shell)
- 0x7701 : Receive File (T1105: Ingress Tool Transfer)
- 0x7702 : Send File (T1041: Exfiltration Over C2 Channel)

A null preserving XOR encryption algorithm is used to hide the buffer's contents (T1573.001: Encrypted Channel: Symmetric Cryptography). All of the samples found use the same single byte XOR key 0x85.

Command and Control Recreation

After reversing the backdoor's network byte parser, we recreated parts of the command and control (C2) server to elicit responses from it. Here the backdoor responds to our command telling it to execute "*net user evil 3v1l /add*".

Notice both the sent and received responses start with 0x33669966 (0x66996633 on the network); enabling fingerprinting of the network traffic. The response buffer reads: "The command completed successfully."

Mustang Panda Connection

We assess with low confidence that CrimsonIAS is associated with Mustang Panda. The similarities with how the binary was packaged along with how it's launched are the basis for this connection.

Inspecting Mustang Panda PlugX samples identified last year, we observed these three pertinent characteristics:

1 – Encrypted with a 10 byte XOR key prepended to the encrypted binary

Figure 7 and 8 show a prepended XOR key separated by a null byte from the encrypted payload. Over the past 8 months we've been monitoring this technique and, outside of this one CrimsonIAS sample, all the other uploaded samples have been the PlugX associated with Mustang Panda. We acknowledge during the last two months of 2020 that two samples deviated from the 10 byte XOR key length; however, the overwhelming majority used a 10 byte prepended XOR key.

2 – Matching shell code at the start of the MZ header (minus the offset value)

Shell code in the MZ header is not exclusive to Mustang Panda, as tools like Cobalt Strike make use of this; however this set of bytes (shown in figures 9 and 10) we've only seen in files related to Mustang Panda over the past few months.

3 – Exported Loader function

Both samples also make use of a reflective loader technique which they export under the name Loader. Searching VTI for this exported function leads to a fair number of results not tied to Mustang Panda samples; so the presence of this export is not unique enough.

Finally, we successfully launched CrimsonIAS by taking a Mustang Panda PlugX archive (complete with the DLL sideloading executable and the sideloaded DLL) and swapping out the encrypted PlugX DLL with the encrypted CrimsonIAS DLL (T1574.002: Hijack Execution Flow: DLL Side-Loading).

Grouping these characteristics together is the basis for the CrimsonIAS connection with Mustang Panda. The reasoning behind the low confidence is the fact that the first two techniques should be trivial to implement/copy and that the third is not unique to Mustang Panda; increasing the likelihood it could be a copycat, false flag, or inadvertent consistency. Additional information on organizations targeted, affected regions, the complete payload, inbound communications, or any associated incidents would potentially help us reassess our confidence in CrimsonIAS' association to Mustang Panda.

Evolution

VirusTotal shows that the first sample was uploaded back in 2018.

SHA256	Compile Time	VT First Submission	DLL Name	Exports
3bc96b4cce0dd550eeb3a563f7ef203614e36fbbbf990726e1afd5d3dcec33e1	1511835471 (2017-11-28 02:17:51)	2018-05-29 08:35:59	Dll.dll	0x42662c: dbkFCallWrapperA (1) 0x40bb34: __dbk_fcall_wrapp (2) 0x41d100: ServiceMain (3) 0x41d224: CPlApp (4)
bde63cd5c3aefed249d2610ca2ee834bde0c0ec06193119363972e3761fb3c63	1527218355 (2018-05-25 03:19:15)	2019-04-28 07:50:41	Dll.dll	0x42662c: dbkFCallWrapperA (1)

SHA256	Compile Time	VT First Submission	DLL Name	Exports
				0x40bb34: __dbk_fcall_wrapp (2) 0x41d108: ServiceMain (3) 0x41d22c: CPlApp (4)
194c0f6c5001b929080d700362e8d8e8009973c82d9409094af2a7ad33506228	1527218355 (2018-05-25 03:19:15)	2018-12-11 03:16:46	Dll.dll	0x42662c: dbkFCallWrapperA (1) 0x40bb34: __dbk_fcall_wrapp (2) 0x41d108: ServiceMain (3) 0x41d22c: CPlApp (4)
5021a19f439d31946e61b7529f8e930ebc9829b1ab1f2274b281b23124113cb1	1527218355 (2018-05-25 03:19:15)	2018-07-23 01:02:09	Dll.dll	0x42662c: dbkFCallWrapperA (1) 0x40bb34: __dbk_fcall_wrapp (2) 0x41d108: ServiceMain (3) 0x41d22c: CPlApp (4)
306175ffc59091515a8a0b211c356843f09fcb65395decd9fe72c9807c17288a	1528707090 (2018-06-11 08:51:30)	2019-05-16 10:21:16	Dll.dll	0x42662c: dbkFCallWrapperA (1) 0x40bb34: __dbk_fcall_wrapp (2) 0x41d108: ServiceMain (3) 0x41d22c: CPlApp (4)
63e144fbe0377e0c365c126d2c03ee5da215db275c5376e78187f0611234c9b0	1531455240 (2018-07-13 04:14:00)	2018-08-04 07:57:02	Dll.dll	0x42562c: dbkFCallWrapperA (1) 0x40bb34: __dbk_fcall_wrapp (2) 0x41cb80: CPlApp (3)
b19fea36cb7ea1cf1663d59b6dcf51a14e207918c228b8b76f9a79ff3a8de36c	1539848755 (2018-10-18 07:45:55)	2019-03-30 13:00:05	Dll.dll	0x42662c: dbkFCallWrapperA (1) 0x40bb34: __dbk_fcall_wrapp (2)

SHA256	Compile Time	VT First Submission	DLL Name	Exports
				0x41ccc8: ServiceMain (3) 0x41cdec: CPlAppl (4)
891ece4c40a7bf31f414200c8c2c31192fd159c1316012724f3013bd0ab2a68e	1582792200 (2020-02-27 08:30:00)	2020-08-09 06:29:04	Dll.dll	0x42662c: dbkFCallWrapperA (1) 0x40bb34: __dbk_fcall_wrapp (2) 0x41cbc4: Loader (

The earliest compile time found suggests this backdoor has been around since at least 2017. The three samples with the compile time 2018-05-25 03:19:15 are the same sample with just the listen on port number changed; probably after compilation.

Overall CrimsonIAS's primary functionality remains consistent over the years. The primary difference lies in how the malicious code is executed. The earlier samples relied on calling the exported function CPlApplet that would register and launch a Windows service that then executes the backdoor functionality (T1569.002: System Services: Service Execution). The file 63e144fbe0377e0c365c126d2c03ee5da215db275c5376e78187f0611234c9b0 is an exception as the exported function CPIApplet does not create a service but it immediately starts up the backdoor.

The latest sample moved away from this technique and it now makes use of a reflective loader technique that is seen across different malware families. This change along with the prepended XOR key and the shellcode provide hints to who might be behind this backdoor.

Conclusion

ThreatConnect believes that Mustang Panda will continue to be active and adapt their toolset as needed to meet their objectives against largely near-abroad targets. The backdoor, CrimsonIAS, passively awaits commands, implying that the actor has some means of proxying/accessing the target's network or, more likely, that the machine targeted is exposed to the public Internet. We encourage entities who think they might have been targeted by Mustang Panda to check for the presence of programs listening for external inbound connections and inspect further if something is found. Verify carefully as the presence of programs listening for inbound connections does not necessarily mean that the machine is compromised.

Naming Convention Note

We generally abstain from adding a new name for malware and threats where other industry reporting has already done so. Two exceptions where we may use our own naming convention:

- When we are unsure whether our findings are consistent with activity sets other organizations have described and named. In this case, we will attempt to describe the overlap and differences between those previously named sets and the activity we're describing.
- When we are describing a previously unnamed malware or threat.

The latter is the case here. Our naming convention is intended to help describe the assessed origin of the threat or malware, along with another identifier that is specific to the entity. In this case, we are using *Crimson* to refer to the malware's assessed Chinese origin, and *IAS* to refer to the Internet Authentication Service (IAS) binary string previously described.

About the Author

ThreatConnect

By operationalizing threat and cyber risk intelligence, The ThreatConnect Platform changes the security operations battlefield, giving your team the advantage over the attackers. It enables you to maximize the efficacy and value of your threat intelligence and human knowledge, leveraging the native machine intelligence in the ThreatConnect Platform. Your

team will maximize their impact, efficiency, and collaboration to become a proactive force in protecting the enterprise. Learn more at www.threatconnect.com.
