HolyCat Ransomware Investigation

Nathan Stein

Paul Mellas

Ferris State University, Fall 2025

Published December 7, 2025

Investigating HolyCat Ransomware

Analysis Introduction

Objective

This post will walk through an analysis of HolyCat utilizing tools within the Any.run suite. We will leverage features of Any.run to bypass obfuscation to better understand how the ransomware operates and interacts with the local system and network. By creating this blog, we hope to spread awareness of the capabilities of this ransomware and recommend analysts a resource for malware analysis.

Why do we care

Information Security professionals deal with constantly changing variants of malware, which requires the learning of new behaviors to best protect systems from each variant. Any.run provides analysts a tool that allows for on-the-fly dynamic analysis of URLs and executables enabling rapid analysis and reaction to new threats. Each analysis takes place in the cloud, requiring few resources from the user while providing a feature rich environment.

Malware Introduction

HolyCat is a piece of ransomware that has been seen in a case where 2.7 million Indian car owners had their data stolen and sold on a hacker forum (Kaaviya, 2025). It has a feature rich executable, not requiring other pieces to be downloaded. From this analysis, it can only be said that it encrypts data, but it has been correlated to RustyStealer, a common information stealer bundled with ransomware. It utilizes modern encryption and defense evasion to deploy a difficult to remediate attack.

Analysis Report

Overview

HolyCat is a Rust-based ransomware that will encrypt victims files on their device when the executable is run. The ransomware encrypts the files with AES-256 which are then exfiltrated to a Discord Webhook as opposed to a normal Command-and-Control (C2) server. At the same time, HolyCat will drop two files into the folder the executable is in.

HolyCat also avoids encrypting system-critical files as to not hinder the Operating System from running, ensuring in the end that victims can still see the ransom instructions.

Summary

HolyCat is only invoked by the user or a threat actor who already has access to the system. It generates an AES-256 key, sends it through an encrypted channel to a discord webhook, and encrypts all files in the user profile. From here, it drops the decryption batch file, readme, and changes background to entice the user to communicate with the threat actor.

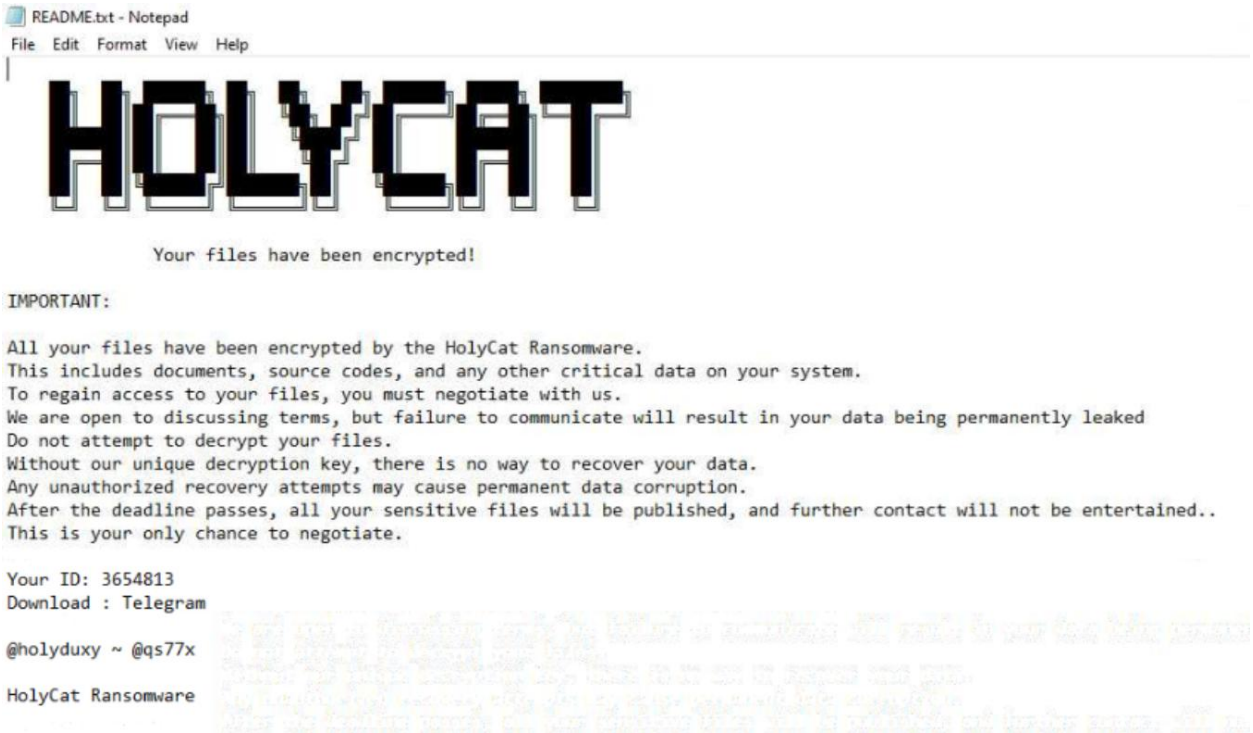
File Activity

HolyCat creates three files, two of which are made locally and the other is an image downloaded from the internet.

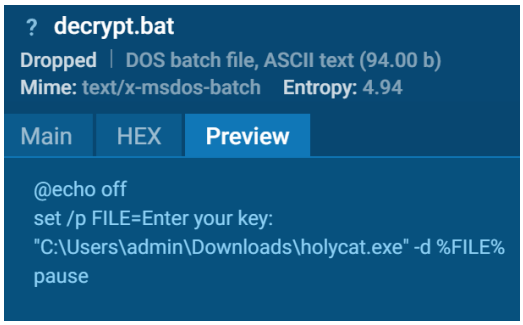
+14479 ms	image	40 Kb	C:\Users\admin\AppData\Local\wallpaper ec9d467eceedf08766273aeb6efaf99c
+14713 ms	text	2 Kb	C:\Users\admin\Downloads\README.txt bfb848aba42fc2dd2ef2d1b5e55c4b80
+14713 ms	text	94 b	C:\Users\admin\Downloads\decrypt.bat 0df52d9de9691ca27a75c4fe43c77148

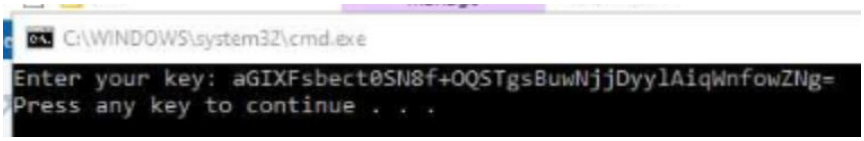
As well as this, HolyCat encrypts files in the users' Desktop, Documents, Music, Videos, and Images folders with the .HC extension.

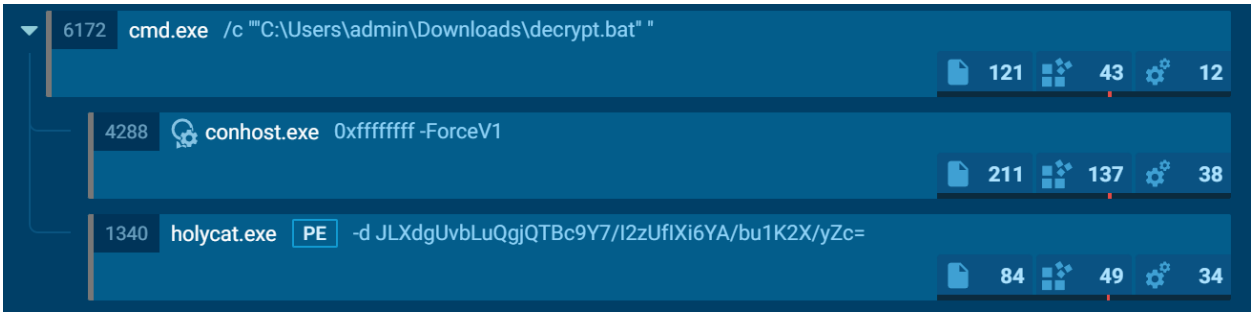
The README.txt details how to reach the operators through telegram and provides the victim ID.



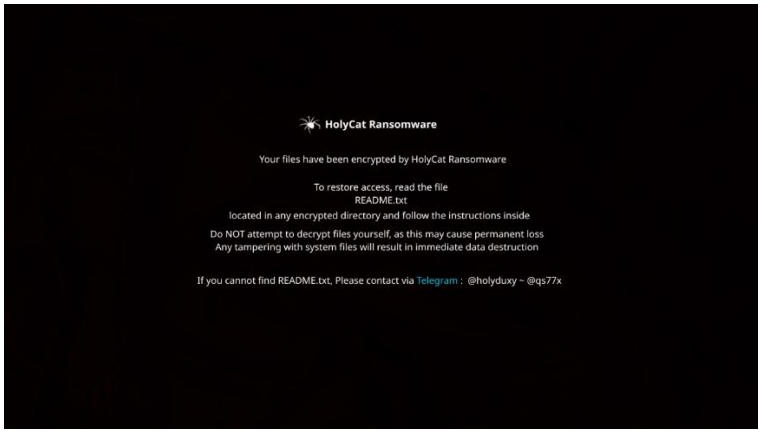
The decrypt.bat file simply runs the holycat program with the -d flag followed by the encryption key provided by the user. This has been proven to reverse the encryption that HolyCat used during this investigation.







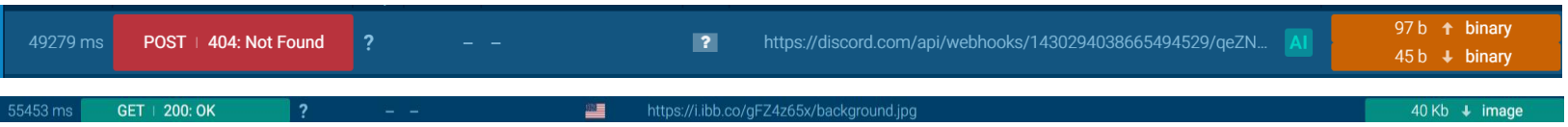
Finally, the wallpaper file is what HolyCat sets the desktop background to and displays some instructions related to communication. This file is downloaded from the internet, with specifics in the next section.



Network Activity

The network traffic included mostly TLS traffic, accounting for 63% of packets sent and 90% of the Bytes observed. Because of this, most organizations would be unable to have observability into the URL paths and the data transmitted.

Utilizing Any.Run's HTTPS MiTM function, we were able to bypass TLS to extract the IOC's for C2 and resource hosting. The following two URLs were found:



The first URL is a Discord webhook, which sends a message to a user containing the victim ID and the encryption key. This can be seen in the below screenshot:

We also identified an HTTPS request to fetch an image file hosted on ibb.co which is the image set as the user's background post compromise.

Processes

HolyCat.exe is a standalone process that does not utilize external executables or subprocesses. The process sends the victim ID and encryption key through a discord webhook, utilizes bcrypt.dll to encrypt the files specified in the File Activity section with AES-256 encryption, drops the supporting files, changes the background, and then exits.

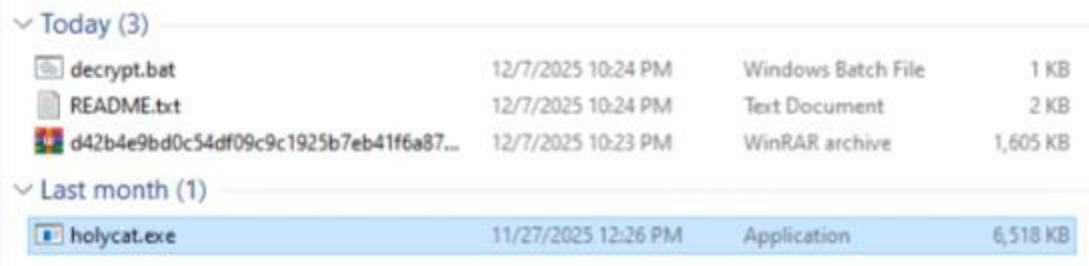
MITRE ATT&CK

Tactic	ID	Technique	Observed Behavior in HolyCat
Discovery	T1083	File/Directory Discovery	Enumerated user folders (Documents, Music, etc.) to target files for encryption.
C2	T1071.001	Application Layer Protocol: Web Protocols	Communicates over standard HTTP/HTTPS ports (80/443) to avoid detection.
C2	T1105	Ingress Tool Transfer	Downloaded the wallpaper image payload from ibb.co.
Impact	T1486	Data Encrypted for Impact	Encrypted user files (AES-256) and appended. HC extension.
Exfiltration	T1567.004	Exfil Over Web Service: Exfiltration Over Webhook	Sent victim ID and encryption keys to a Discord Webhook.

Extracting Indicators of Compromise (IOCs)

The best ways to identify potential Indicators of Compromise (IOCs) would be a Security Information and Event Management system (SIEM). The SIEM would ingest network logs which would alert on C2 communications or traffic to malicious external resources. Additionally, IOCs such as the encrypting of all the files on the system drive can be identified by changes in file name to the identified .HC extension.

The files created by HolyCat also provide a method of detection. Both “README.txt” and “decrypt.bat” are the files that are created when holycat.exe runs. Their hashes can be found in the IOC CSV file.



Endpoint Detection and Response (EDR) software can detect the holycat.exe process due to known hashes. These programs can block the program from running through known file hashes as well as dynamic executable analysis.

Exporting IOCs

The identified IOCs related to HolyCat have been exported to a CSV file that can be opened in Excel or ingested into a threat intelligence platform for detection. These include the URLs of C2 communication and the downloaded resources as well as the hashes of the dropped files.

The URLs and hashes can be added to a blocklist in an EDR solution. Alternatively, the URLs can be blocked at the firewall level. If a compromise is suspected to use HolyCat, the IOCs provided can be used in a threat hunt to detect its presence.

Conclusion

Overall, Holycat proves to be an effective piece of ransomware. It can evade detection due to it utilizing Discord as its method of C2, which uses TLS for session encryption. While Holycat at this point has not shown signs of exfiltrating files themselves, it has been correlated to RustyStealer which suggests it may be on its way to implementing data exfiltration. (kernelv0id, 2025)

Future research on this ransomware should include analyzing the Discord webhook and IBB image platform to potentially identify the maker of the ransomware. Additional considerations should be placed into identifying its ties to RustyStealer.

References

Cryakl. (n.d.). *Ransomware-database/HOLYCAT/d42b4e9bd0c54df09c9c1925b7eb41f6a877963ededdd0dd463e3d16dfe6cd5d.7z at main · Cryakl/ransomware-database*. GitHub. https://github.com/Cryakl/Ransomware-Database/blob/main/HolyCat/d42b4e9bd0c54df09c9c1925b7eb41f6a877963ededdd0dd463e3d16dfe6cd5d.7z

Kaaviya. (2025, April 7). *Hackers allegedly uploaded 2.7m Indian car owners data on Hacker Forums*. Cyber Security News. https://cyberpress.org/hackers-allegedly-uploaded-2-7m/

kernelv0id. (2025). *Checking your browser*. MalwareBazaar. https://bazaar.abuse.ch/browse/tag/HolyCat/

Stein, N. (2025, December 7). *Any.run HolyCat Report*. Any.run. https://any.run/report/091c498d92b620b7b1ba6d254864b97176438c9f32790b987aab6a87bac9efe1/1bed07c4-32eb-40a6-806c-a7a9be8a4ce5