Threat Spotlight: TeslaCrypt – Decrypt It Yourself

By Talos Group

Published: 2015-04-27 · Archived: 2026-04-05 21:39:59 UTC

This post was authored by: Andrea Allievi, Earl Carter & Emmanuel Tacheau

Update 4/28: Windows files recompiled with backward compatibility in Visual Studio 2008

Update 5/8: We've made the source code available via Github here

Update 6/9/2016: We've released a tool to decrypt any TeslaCrypt Version

After the takedown of Cryptolocker, we have seen the rise of Cryptowall. Cryptowall 2 introduced "features" such as advanced anti-debugging techniques, only to have many of those features removed in Cryptowall 3. Ransomware is becoming an extremely lucrative business, leading to many variants and campaigns targeting even localized regions in their own specific languages. Although it is possible that these multiple variants are sponsored by the same threat actor, the most likely conclusion is that multiple threat actors are jumping in to claim a portion of an ever increasing ransomware market. One of the latest variants is called TeslaCrypt and appears to be a derivative of the original Cryptolocker ransomware. Although it claims to be using asymmetric RSA-2048 to encrypt files, it is making use of symmetric AES instead. Talos was able to develop a tool which decrypts the files encrypted by the TeslaCrypt ransomware.

Click for Larger Image

At the first glance, the dropper appears to be related to the original CryptoLocker. The malware states that data files, such as photos, videos and documents on the victim's computer have been encrypted with the RSA-2048 asymmetric algorithm. As we shall see, that statement is not entirely accurate.

Targeting files that users value highly makes ransomware very effective at getting users to pay the ransom. TeslaCrypt is interesting because it also targets and encrypts computer games files, such as saved games and Steam activation keys. This means that TeslaCrypt is targeting many different types of users, including PC gamers. Just like irreplaceable photos, a game save, which is the product of countless hours of gaming, is extremely valuable and hard to replace.

We have analysed two samples of TeslaCrypt, the first dated March 2015 and the second dated April 2015. Their SHA256 are:

- 3372c1edab46837f1e973164fa2d726c5c5e17bcb888828ccd7c4dfcc234a370
- 6c6f88ebd42e3ef5ca6c77622176183414d318845f709591bc4117704f1c95f4

Both samples implement the following hashing algorithms:

- SHA1
- SHA256
- RIPEMD160
- BASE58

- BASE64

Infection Vector And Setup Function

This ransomware is usually distributed as an email attachment or through websites that redirect the victim to the Angler Exploit Kit. In our analysis, the exploit kit delivered a malicious Flash object containing an exploit against CVE-2015-0311. The payload for this exploit was a TeslaCrypt sample.

We are only going to give a quick introduction on the dropper's architecture and the setup function because this functionality has been widely covered.

Most TeslaCrypt samples use COM+ sandbox evasion techniques. For example, the dropper we analysed uses simple detection code that verifies if the "URLReader2" COM interface has been correctly installed in the DirectShow filter graph list:

```
hr = CoCreateInstance(CLSID_FilgraphManager, NULL,
CLSCTX_INPROC_SERVER,
      IID_IGraphBuilder, (void **)&pGraph);
hr = CoCreateInstance(CLSID_URLReader2, NULL, CLSCTX_INPROC_SERVER,
__uuidof(IBaseFilter),
       (void **)&pUrlReader);
pGraph->AddFilter((IBaseFilter*)pUrlReader, L"CLSID_URLReader2");

// Check the Emulation VM here:
hr = pGraph->FindFilterByName(L"CLSID_URLReader2",
(void**)&pUrlFilter);
if (!pUrlFilter) {
  LPVOID lpFuncPtr = NULL;
  // CALL a NULL routine and crash the dropper
  lpFuncPtr();
}
```

If the check passes, the real dropper is extracted and executed using a well-known method that makes use of the *ZwMap(Unmap)ViewOfSection* API functions to unmap the original PE memory image and re-map another image file. The final unpacked executable locates specific Windows directories such as the Application Data directory, and builds support files like the "key.dat" file, and files to store decryption instructions. The executable also adjusts its own privileges (adds "SeDebugPrivilege") and copies itself using a random file name to the user's Application Data directory. A new process is then spawned and execution is transferred to it. The original dropper file is deleted. The main malware window is created and five threads are spawned, followed by the window message dispatching cycle.

TeslaCrypt threads perform the following:

- Delete all system Volume Shadow Copies by executing "vssadmin.exe delete shadows /all /quiet" command
- Open the "key.dat" file and recover encryption keys. If "key.dat" file doesn't exist, create the keys and store them in an encrypted form in the "key.dat" file.
- Send the new master encryption key to the C&C server through POST request (the latest sample that we have analysed contains the following C&C server URLs:
    - 7tno4hib47vlep5o.63ghdye17.com

    - 7tno4hib47vlep5o.79fhdm16.com
    - 7tno4hib47vlep5o.tor2web.blutmagie.de
    - 7tno4hib47vlep5o.tor2web.fi
- Implement anti-tampering protection: every 200 milliseconds, TeslaCrypt enumerates all running processes and if a process with a filename that contains any of the words below is found, that process is terminated using the *TerminateProcess* Windows API function
    - taskmgr
    - procexp
    - regedit
    - msconfig
    - cmd.exe

File Encryption – Introduction

After the initialization routine and the deletion of the Volume Shadow copies, the sample creates the "key.dat" file where it stores all the encryption keys. The dropper from March 2015 calculates at least 2 different main keys: a payment key and a master encryption key. The other dropper implements the concept of an additional key known as the "Recovery key'.

"GetAndHashOsData" is the function responsible for creating the base buffer for the generation of all keys. At startup it acquires the following info:

- the global workstation's LAN network statistics, using the *NetStatisticsGet* API function
- 64 random bytes generated by Windows Crypto functions
- all heap descriptors of its own process
- all active process descriptors and the threads descriptors of each process
- all loaded modules in each process
- the workstation's physical memory information

Once the data is acquired, it generates a big array of SHA1 values, one for every 20 bytes of acquired data. At the end it calculates and stores a global SHA1 value for the entire array, in a symbol that we have called "g_lpGlobalOsDataSha1".

With these 2 items, the "FillBuffWithEncryptedOsData" routine is able to fill a generic buffer with the calculated data, in a pseudo-random manner. A master key and a payment key are generated using this function (each key is 32 bytes wide), their SHA256 is calculated and finally a custom algorithm is used to shift left and shift right the 2 keys. The two shifted SHA256 values are stored in the "key.dat" file.

The Key File

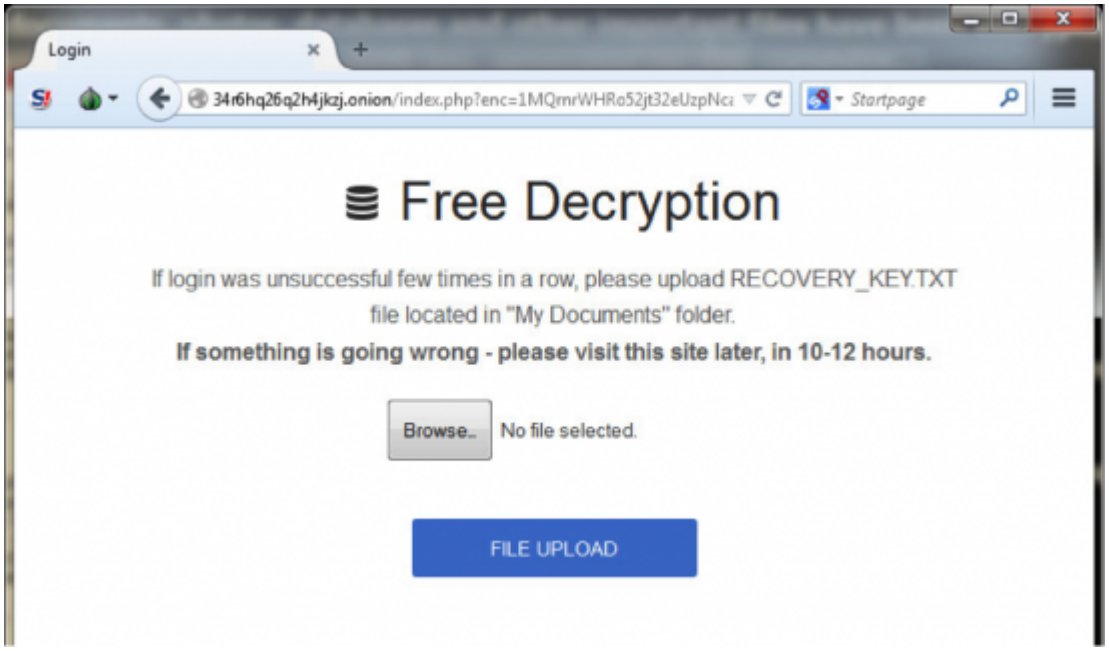
The "*OpenKeyFileAndWrite*" routine tries to open the "key.dat" file, located in the user's Application Data directory. If it doesn't exist, it generates the 2 master keys (3 in case of the most recent dropper) as well as other keys, and stores them in the key file.

Here is a little schema of the layout of the "key.dat" file:

OFFSET	DESCRIPTION	SIZE
• 0x00	The user bitcoin address (represented in BASE58 encoding)	0x28
• 0x64	SHA1 derived from the OS info *	0x20
• 0x84	Shifted recovery key	0x40
• 0xE5	SHA1 derived from the OS info *	0x40
• 0x125	The local time acquired when the key file has been generated	0x10
• 0x136	Shifted SHA256 of the Payment Key	0x20
• 0x177	Shifted SHA256 of the Master key	0x20

* = We currently don't know precisely how this value is used by TeslaCrypt

The latest version of the dropper creates a "RECOVERY_KEY.TXT" file inside the user's document directory. It does this to achieve a particular goal: if the victim workstation is offline or if a firewall blocks the communication with the C&C server, the dropper will proceed with the destruction of the master key inside the "key.dat" file, after the encryption of all files has been completed. To recover the files, the user would have to connect to the threat actor's TOR website and provide the recovery key. The threat actors use a custom algorithm to to recover the master key from the recovery key:



Click for Larger Image

The recovery key file contains 3 pieces of information in an human-readable form, separated by a carriage return character:

- The Bitcoin address
- The payment key ID (32 hex digits)

- The recovery key (64 hex digits)

The File Encryption Algorithm

File encryption is performed in a dedicated thread. The code for the encryption thread takes the shifted master key, calculates its SHA256 hash and starts to enumerate all files of the victim workstation (filtering by extension type, Tesla Crypt supports over 170 different file extensions).

"EncryptFile" is the function that manages the entire file-encryption process. It:

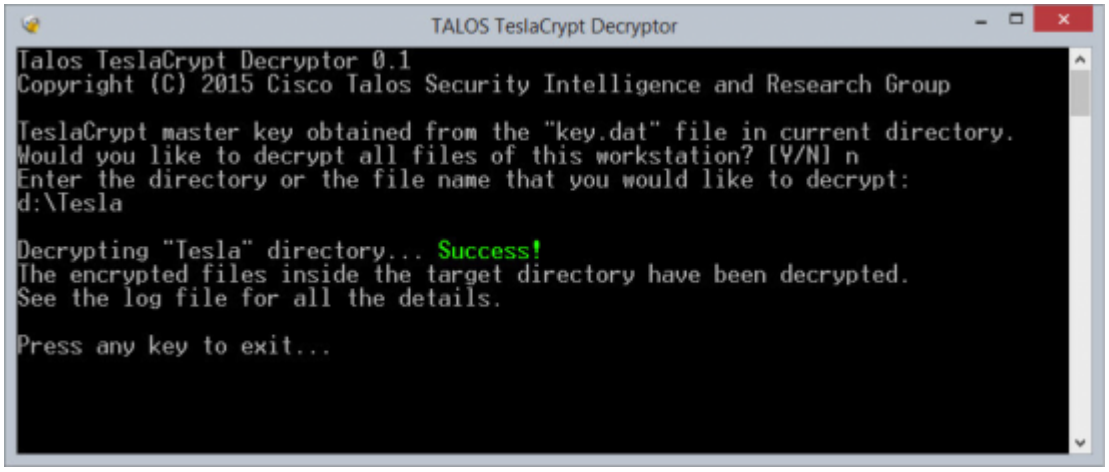
- generates a 16-bytes Initialization Vector for AES, using the *GetAndHashOsData* API function
- reads the target file
- initializes the AES encryption algorithm through the creation of the AES context data structure
- finally encrypts the contents of the file using an AES CBC 256-bit algorithm implemented in the "EncryptWithCbcAes" function.

When the process is complete, the new encrypted file is created. The new file contains a small header (composed of the AES Initialization Vector in its first 16 bytes followed by the original file size in the next 4 bytes), and then the actual encrypted bytes.

The pop up window displays misleading information: the encryption method is a symmetric AES, and not an asymmetric RSA-2048 as stated by TeslaCrypt in the screenshot above. As proof that TeslaCrypt is truly using symmetric AES and not asymmetric RSA, we provide for a decryption utility capable of decrypting all the files encrypted by this ransomware (provided you have the master key).

The Talos TeslaCrypt Decryption Tool

Our decryption utility is a command line utility. It needs the "key.dat" file to properly recover the master key used for file encryption. Before it begins execution, it searches for "key.dat" in its original location (the user's Application Data directory), or in the current directory. If it isn't able to find and correctly parse the "key.dat" file, it will return an error and exit.



Click for Larger Image

To use this tool, just copy the “key.dat” file into the tool’s directory and then specify either the encrypted file or a directory containing encrypted files. That’s it! Files should be decrypted and returned to their original content.

Here is the list of command line options:

- /help – Show the help message
- /key – Manually specify the master key for the decryption (32 bytes/64 digits)
- /keyfile – Specify the path of the “key.dat” file used to recover the master key.
- /file – Decrypt an encrypted file
- /dir – Decrypt all the “.ecc” files in the target directory and its subdirs
- /scanEntirePc – Decrypt “.ecc” files on the entire computer
- /KeepOriginal – Keep the original file(s) in the encryption process
- /deleteTeslaCrypt – Automatically kill and delete the TeslaCrypt dropper (if found active in the target system)

Back up your encrypted files before you use this utility. Provided without any guarantees.

Link to the Tool

The TeslaCrypt Decryption Tool is provide as-is and is not officially supported. The user assumes all liability for the use of the tool.

Windows binary: https://github.com/vrtadmin/TeslaDecrypt/tree/master/Windows

IOCs

Hashes:

3372c1edab46837f1e973164fa2d726c5c5e17bcb888828ccd7c4dfcc234a370
6c6f88ebd42e3ef5ca6c77622176183414d318845f709591bc4117704f1c95f4

IP Addresses:

38.229.70.4
82.130.26.27
192.251.226.206

Domains Contacted:

7tno4hib47vlep5o.63ghdye17.com
7tno4hib47vlep5o.79fhdm16.com
7tno4hib47vlep5o.tor2web.blutmagie.de
7tno4hib47vlep5o.tor2web.fi

ThreatGrid has also added a behavioral indicator to identify TeslaCrypt.

Click for Larger Image

Conclusion

Analysing TeslaCrypt ransomware was a challenge. All the encryption and hashing algorithms in the dropper made the analysis pretty difficult. As we have seen, sometimes the threat actors authors even lie. Nevertheless, ransomware continues to plague users. Incorporating a layered defense is critical to combating this type of threat before it has the chance to encrypt files. A good system backup policy is the best way to recover files that have been hijacked.

Product	Protection
AMP	✔
CWS	✔
ESA	✔
Network Security	✔
WSA	✔

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.
CWS or WSA web scanning prevents access to malicious websites and detects malware used in these attacks.
The Network Security protection of IPS and NGFW have up-to-date signatures to detect malicious network activity by threat actors.

ESA can block malicious emails including phishing and malicious attachments sent by threat actors as part of their campaign.
