Progetto W24D4

Traccia:

EPICODE

Esercizio
Progetto

Malware Analysis

Il Malware da analizzare è nella cartella Build_Week_Unit_3 presente sul desktop della macchina virtuale dedicata.

Analisi statica

Con riferimento al file eseguibile Malware_Build_Week_U3, rispondere ai seguenti quesiti utilizzando i tool e le tecniche apprese nelle lezioni teoriche:

- Quanti parametri sono passati alla funzione Main()?
- Quante variabili sono dichiarate all'interno della funzione Main()?
- Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate
- Quali librerie importa il Malware? Per ognuna delle librerie importate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.

EPICODE

Esercizio
Progetto

Malware Analysis

Con riferimento al Malware in analisi, spiegare:

- Lo scopo della funzione chiamata alla locazione di memoria **00401021**
- Come vengono passati i parametri alla funzione alla locazione **00401021**;
- Che oggetto rappresenta il parametro alla locazione **00401017**
- Il significato delle istruzioni comprese tra gli indirizzi **00401027** e **00401029**.
- Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C.
- Valutate ora la chiamata alla locazione **00401047**, qual è il valore del parametro «ValueName»?

EPICODE

Esercizio
Progetto

Malware Analysis

Filtrate includendo solamente l'attività sul registro di Windows.

- Quale chiave di registro viene creata?
- Quale valore viene associato alla chiave di registro creata?

Passate ora alla visualizzazione dell'attività sul file system.

- Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?

Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

Soluzione:

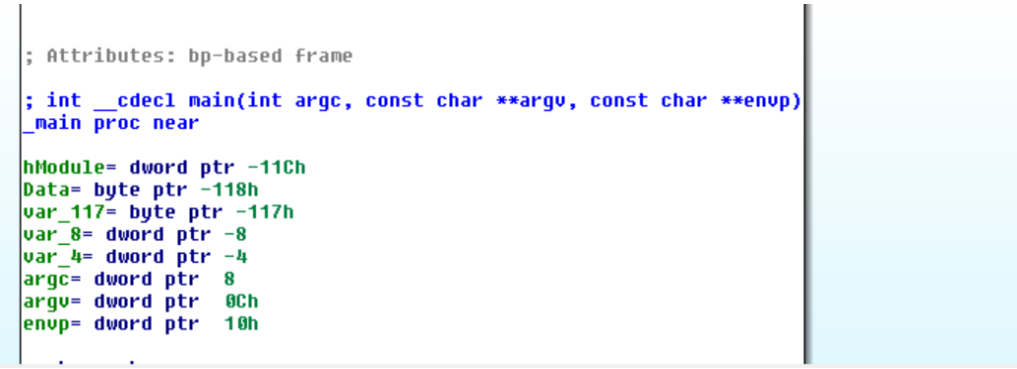
Analisi statica:

1. La funzione **main()** accetta **tre** parametri:

   -argc: un puntatore a un intero (**dword ptr 8**), che rappresenta il numero di argomenti passati al programma dalla riga di comando.

   -argv: un puntatore (**dword ptr 0Ch**) all'array di stringhe degli argomenti passati al programma.

   -envp: un puntatore (**dword ptr 10h**) all'array di stringhe delle variabili d'ambiente.



2. All'interno della funzione `Main()` sono dichiarate cinque variabili:

   - **hModule**: un puntatore a dword (dword ptr -11Ch), utilizzato per memorizzare l'handle del modulo corrente.
   - **Data**: un puntatore a byte (byte ptr -118h), utilizzato per memorizzare dati temporanei.
   - **var_117**: un puntatore a byte (byte ptr -117h), probabilmente utilizzato per l'allocazione di spazio temporaneo per dati.
   - **var_8:** un puntatore a dword (dword ptr -8), probabilmente utilizzato per l'allocazione di spazio temporaneo per dati.
   - **var_4**: un puntatore a dword (dword ptr -4), probabilmente utilizzato per l'allocazione di spazio temporaneo per dati.

3. Le sezioni presenti all'interno del file eseguibile sono:

   - .**text** (sezione del codice):

     Questa sezione contiene il codice eseguibile del programma, ossia le istruzioni in linguaggio macchina che vengono eseguite dalla CPU. È la sezione in cui risiedono le istruzioni del programma, comprese le funzioni principali come main().

     In questa sezione, il codice viene eseguito sequenzialmente, istruzione dopo istruzione, per svolgere le operazioni previste dal programma.

   - .**data** (sezione dei dati inizializzati):

     Questa sezione contiene dati globali e statici inizializzati durante il caricamento del programma in memoria. Questi dati hanno un valore predefinito assegnato nel codice sorgente del programma.

     Ad esempio, variabili globali o statiche inizializzate con un valore specifico saranno allocate in questa sezione.

     I dati in questa sezione possono essere letti e modificati durante l'esecuzione del programma.

4. Il malware importa diverse librerie, tra cui **KERNEL32.dll**. Analizziamo le funzioni importate da questa libreria e facciamo delle ipotesi basate sulle loro funzionalità:

   - **SizeofResource**: Questa funzione viene utilizzata per ottenere la dimensione, in byte, di una risorsa all'interno di un modulo specifico. Potrebbe essere utilizzata dal

malware per ottenere informazioni sulle risorse incorporate all'interno del suo eseguibile, ad esempio per manipolare o estrarre risorse come file o dati incorporati.

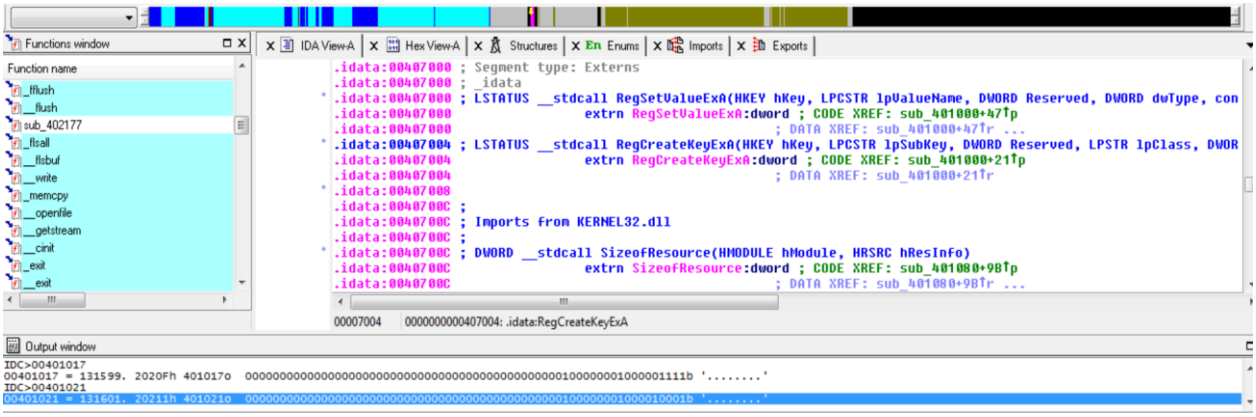
- **LockResource**: Utilizzata per ottenere un puntatore al primo byte di una risorsa di memoria. Questa funzione potrebbe essere impiegata per manipolare o eseguire operazioni sui dati delle risorse, come il caricamento di codice eseguibile o la decodifica di risorse crittografate.
- **LoadResource**: Serve per caricare una risorsa specificata in memoria. Questa funzione potrebbe essere utilizzata per caricare risorse dal file eseguibile del malware in memoria, ad esempio per eseguire codice incorporato o manipolare dati incorporati.
- **VirtualAlloc**: Viene utilizzata per allocare memoria nello spazio di indirizzamento virtuale di un processo. Potrebbe essere sfruttata dal malware per allocare spazio di memoria per l'esecuzione di codice malevolo o per nascondere parti del suo codice.
- **GetModuleFileNameA**: Utilizzata per ottenere il percorso completo del file eseguibile di un modulo specifico. Il malware potrebbe impiegarla per ottenere informazioni sul proprio percorso nel sistema, ad esempio per configurare percorsi di salvataggio o nascondersi all'interno del sistema.
- **GetModuleHandleA**: Serve per ottenere un handle al modulo specificato. Potrebbe essere usata dal malware per ottenere informazioni sui moduli caricati nel processo, ad esempio per identificare altre librerie caricate o ottenere informazioni sul sistema.
- **FreeResource**: Utilizzata per liberare le risorse caricate in memoria. Potrebbe essere sfruttata dal malware per liberare risorse caricate in memoria dopo averle utilizzate, ad esempio per nascondere tracce o ottimizzare l'allocazione di memoria.
- **FindResourceA**: Serve per individuare una risorsa all'interno di un modulo specifico. Potrebbe essere utilizzata dal malware per cercare risorse specifiche all'interno del suo file eseguibile o di altri moduli caricati, ad esempio per recuperare configurazioni o dati incorporati.
- **CloseHandle**: Utilizzata per chiudere un handle aperto. Potrebbe essere impiegata dal malware per chiudere handle di file o altri handle aperti durante le sue operazioni, ad esempio per liberare risorse o nascondere attività.
- **GetCommandLineA**: Serve per ottenere la riga di comando dell'applicazione corrente. Il malware potrebbe utilizzare questa funzione per ottenere informazioni sulle opzioni di avvio o sugli argomenti passati al processo.

5. La locazione di memoria **00401021** si riferisce al codice di una specifica istruzione all'interno del file eseguibile. In questo caso, sembra che l'indirizzo sia associato a una chiamata alla funzione **RegCreateKeyExA()** proveniente dalla sezione **.idata**

   – La funzione **RegCreateKeyExA()** è utilizzata per creare o aprire una chiave di registro e restituisce lo stato dell'operazione (**LSTATUS**). Prende diversi parametri come argomenti, tra cui **hKey** (l'handle della chiave di registro padre), **lpSubKey** (il nome della

chiave di registro da creare o aprire), **samDesired** (i diritti di accesso desiderati per la chiave) e altri.

- L'indirizzo **00407004** sembra essere l'import address table (IAT) entry per la funzione **RegCreateKeyExA()**, il che significa che questa istruzione si riferisce all'importazione della funzione dalla libreria di sistema **advapi32.dll** nell'eseguibile. Quando il programma viene eseguito, questo indirizzo verrà sovrascritto con l'indirizzo reale della funzione **RegCreateKeyExA()** caricata in memoria.
- In breve, **00401021** è l'indirizzo nel codice del programma in cui viene chiamata la funzione **RegCreateKeyExA()**, mentre **00407004** è l'indirizzo nella sezione **.idata** che contiene l'indirizzo della funzione importata.



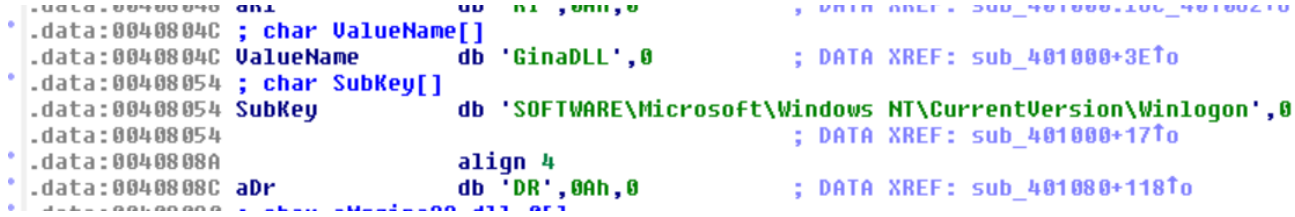
6. La funzione **RegCreateKeyExA** è una funzione API di Windows utilizzata per creare o aprire una chiave di registro. La convenzione di chiamata standard per le funzioni API di Windows è la convenzione di chiamata **__stdcall**, che prevede il passaggio dei parametri attraverso lo stack in un ordine specifico.

Ora, analizzando la firma della funzione **RegCreateKeyExA**, possiamo dedurre come i parametri vengono passati alla funzione:

1. **HKEY hKey**: Questo parametro è passato attraverso il registro ECX.
2. **LPCSTR lpSubKey**: Viene passato attraverso il registro EDX.
3. **DWORD Reserved**: Questo parametro è passato attraverso lo stack.
4. **LPSTR lpClass**: Viene passato attraverso lo stack.
5. **DWORD dwOptions**: Questo parametro è passato attraverso lo stack.
6. **REGSAM samDesired**: Viene passato attraverso lo stack.
7. **const LPSECURITY_ATTRIBUTES lpSecurityAttributes**: Viene passato attraverso lo stack.
8. **PHKEY phkResult**: Viene passato attraverso lo stack.
9. **LPDWORD lpdwDisposition**: Viene passato attraverso lo stack.

Quindi, alla locazione 00401021, la funzione **RegCreateKeyExA** viene chiamata con i relativi parametri, passati come descritto sopra, utilizzando la convenzione di chiamata **__stdcall**.

7. Il parametro alla locazione 00401017 rappresenta l'indirizzo dell'inizio della stringa che specifica il sottochiave del registro da creare o aprire. La stringa è "SOFTWARE\Microsoft\Windows NT\CurrentVersion", che rappresenta il percorso della chiave di registro nel formato richiesto dalla funzione **RegCreateKeyExA**.

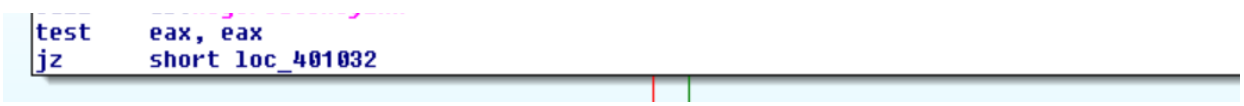


Quindi, l'oggetto rappresentato da **push offset SubKey** è un puntatore alla stringa che specifica il percorso della sottochiave del registro.

8. Le istruzioni comprese tra gli indirizzi 00401027 e 00401029 sono:

   - **test eax, eax**: Questa istruzione effettua un'operazione logica AND bit a bit tra il registro EAX e se stesso. Il risultato viene impostato a zero se il risultato dell'AND è zero, altrimenti viene impostato l'indicatore di zero (ZF) se il risultato è diverso da zero. In sostanza, questa istruzione controlla se il valore di EAX è zero.

   - **jz short loc_401032**: Questa è un'istruzione di salto condizionale. Se il flag ZF (zero flag) è impostato, il programma salta all'indirizzo specificato da **loc_401032**, altrimenti prosegue normalmente con l'istruzione successiva.



In breve, queste istruzioni controllano se il valore di EAX è zero e, in caso affermativo, eseguono un salto corto (short jump) all'indirizzo 00401032.

10. Il valore del parametro "ValueName" è **'GinaDLL'**, come indicato dalla riga:

cssCopy code

.data:0040804C ValueName db 'GinaDLL',0 ; DATA XREF: sub_401000+3Eo

(vedi 2 immagini precedenti)

Quindi, il parametro "ValueName" è una stringa che contiene il valore 'GinaDLL'.

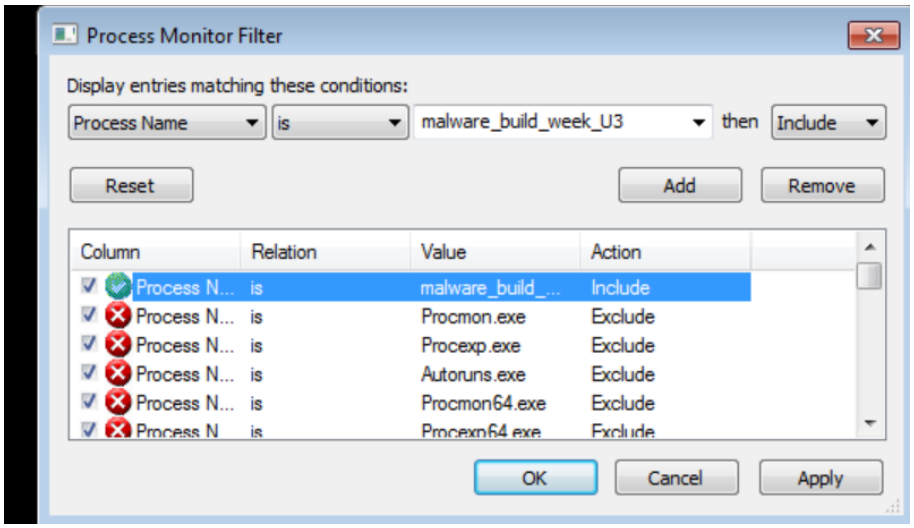
11. Conversione in codice C:

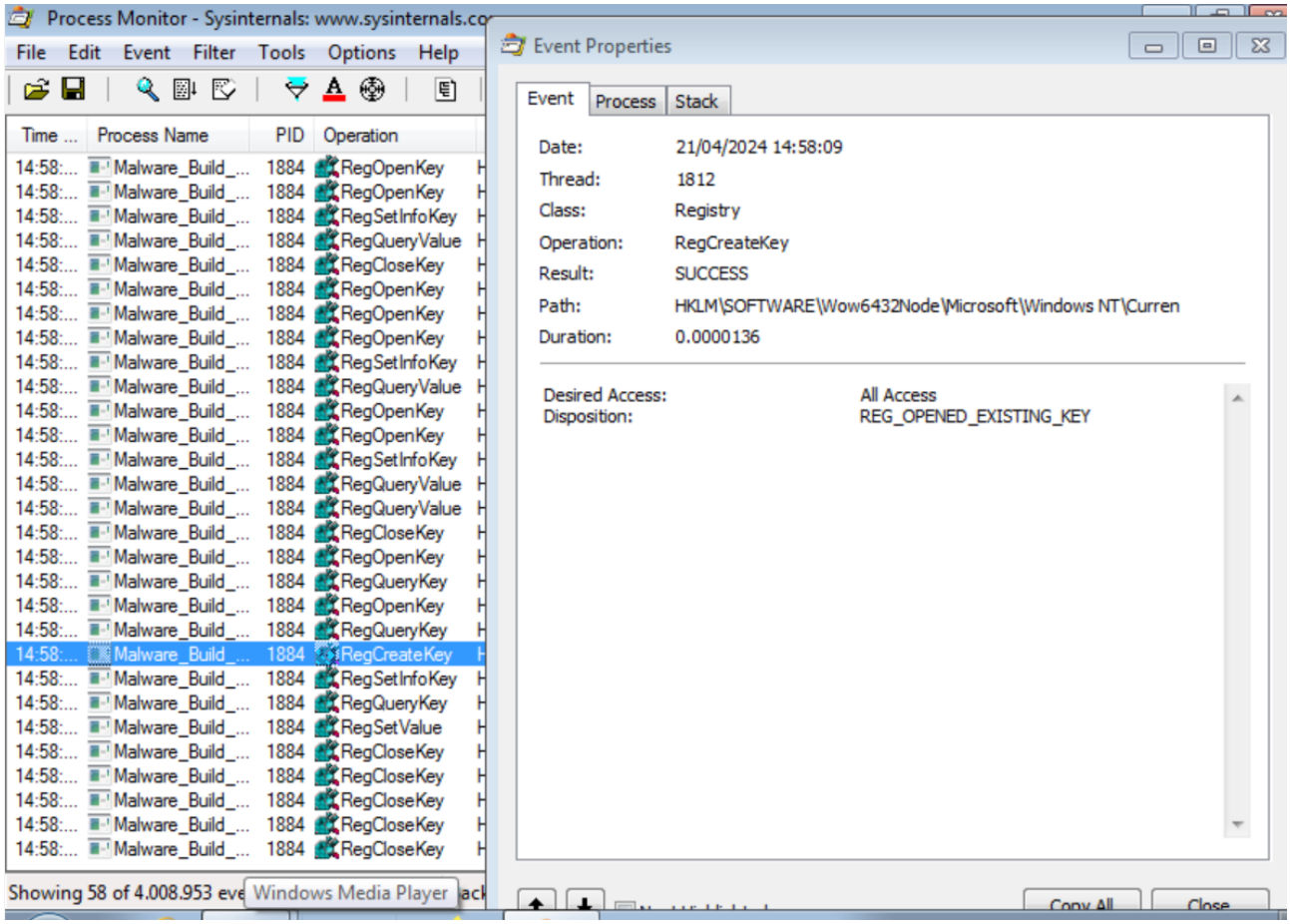
```
#include <windows.h>

int main() {
    HMODULE hModule = GetModuleHandleA(NULL);
    char Data[MAX_PATH];

    GetModuleFileNameA(NULL, Data, MAX_PATH);

    char* found = strrchr(Data, '\\');
    if (found != NULL) {
        *found = '\0';
        if (strstr(Data, "\\msgina32.dll") != NULL) {
            // Fai qualcosa qui...
        }
    }

    return 0;
}
```

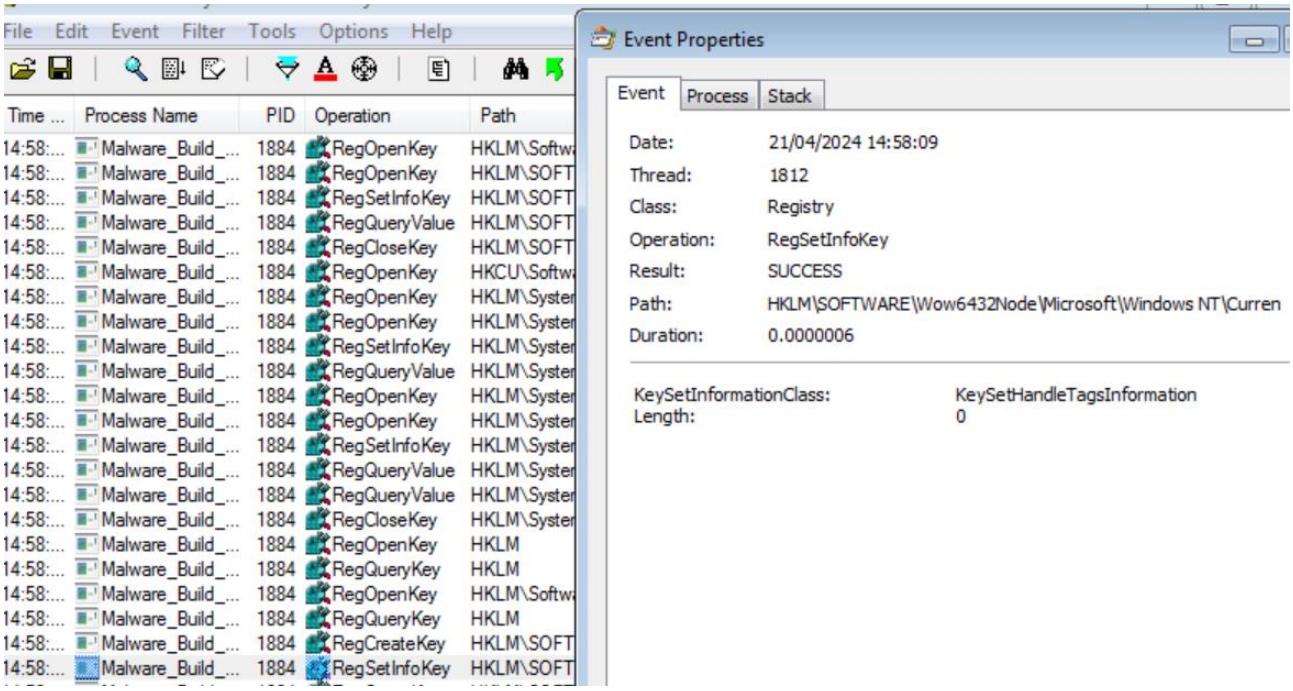
Analisi dinamica:

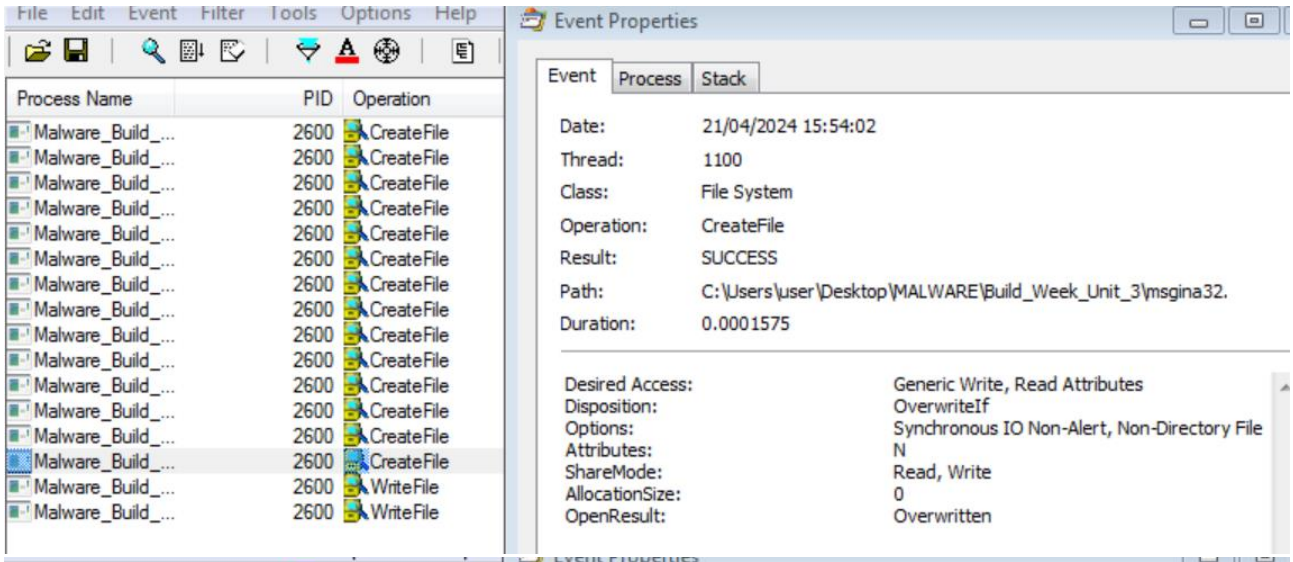
Procmon:

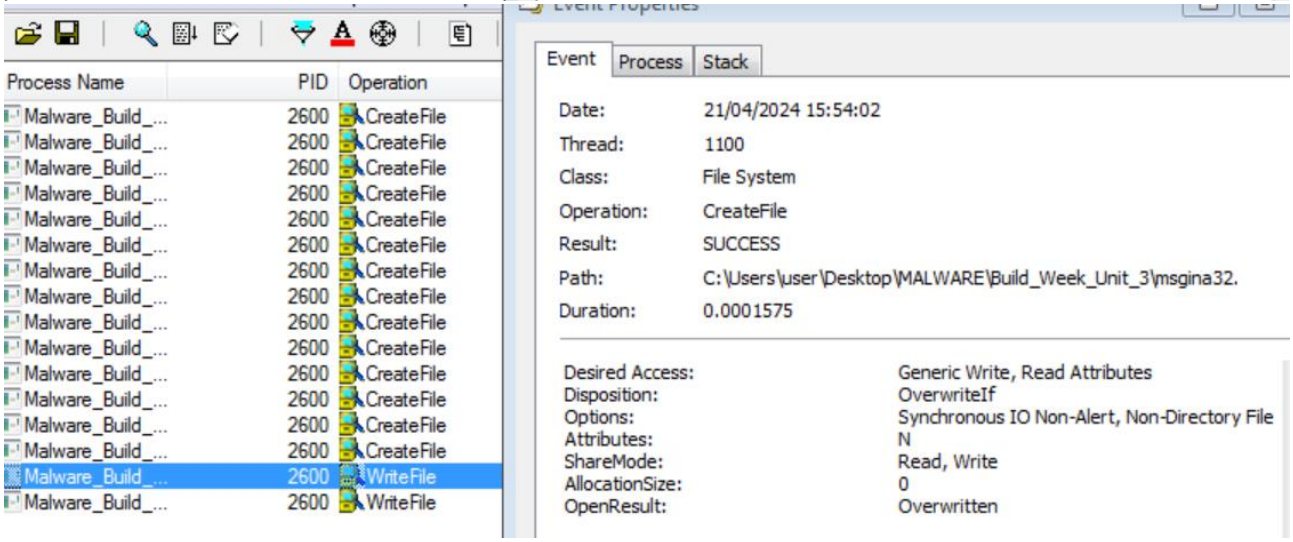
Grazie a Procmon siamo in grado di rilevare la creazione/modifica di eventuali chiavi di registro. Nel dettaglio grazie a Procmon siamo in grado di visualizzare la creazione di una nuova chiave di registro 'RegCreateKey'.
Oltretutto siamo anche in grado di individuare la chiamata di sistema che ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware (Msgina32):


Process Monitor Filter
Display entries matching these conditions:
Process Name
is
malware_build_week_U3
then
Include
Reset
Add
Remove
Column
Relation
Value
Action
Process N... is malware_build_... Include
Process N... is Procmon.exe Exclude
Process N... is Procexp.exe Exclude
Process N... is Autoruns.exe Exclude
Process N... is Procmon64.exe Exclude
Process N... is Procexp64.exe Exclude
OK
Cancel
Apply


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help
Time ... Process Name PID Operation
14:58:... Malware_Build_... 1884 RegOpenKey
14:58:... Malware_Build_... 1884 RegOpenKey
14:58:... Malware_Build_... 1884 RegSetInfoKey
14:58:... Malware_Build_... 1884 RegQueryValue
14:58:... Malware_Build_... 1884 RegCloseKey
14:58:... Malware_Build_... 1884 RegOpenKey
14:58:... Malware_Build_... 1884 RegOpenKey
14:58:... Malware_Build_... 1884 RegOpenKey
14:58:... Malware_Build_... 1884 RegSetInfoKey
14:58:... Malware_Build_... 1884 RegQueryValue
14:58:... Malware_Build_... 1884 RegOpenKey
14:58:... Malware_Build_... 1884 RegOpenKey
14:58:... Malware_Build_... 1884 RegSetInfoKey
14:58:... Malware_Build_... 1884 RegQueryValue
14:58:... Malware_Build_... 1884 RegQueryValue
14:58:... Malware_Build_... 1884 RegCloseKey
14:58:... Malware_Build_... 1884 RegOpenKey
14:58:... Malware_Build_... 1884 RegQueryKey
14:58:... Malware_Build_... 1884 RegOpenKey
14:58:... Malware_Build_... 1884 RegQueryKey
14:58:... Malware_Build_... 1884 RegCreateKey
14:58:... Malware_Build_... 1884 RegSetInfoKey
14:58:... Malware_Build_... 1884 RegQueryKey
14:58:... Malware_Build_... 1884 RegSetValue
14:58:... Malware_Build_... 1884 RegCloseKey
14:58:... Malware_Build_... 1884 RegCloseKey
14:58:... Malware_Build_... 1884 RegCloseKey
14:58:... Malware_Build_... 1884 RegCloseKey
14:58:... Malware_Build_... 1884 RegCloseKey
Showing 58 of 4.008.953 eve
Windows Media Player
Event Properties
Event Process Stack
Date: 21/04/2024 14:58:09
Thread: 1812
Class: Registry
Operation: RegCreateKey
Result: SUCCESS
Path: HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\Curren
Duration: 0.0000136
Desired Access: All Access
Disposition: REG_OPENED_EXISTING_KEY
Copy All
Close


File Edit Event Filter Tools Options Help
Time ... Process Name PID Operation Path
14:58:... Malware_Build_... 1884 RegOpenKey HKLM\Softw
14:58:... Malware_Build_... 1884 RegOpenKey HKLM\SOFT
14:58:... Malware_Build_... 1884 RegSetInfoKey HKLM\SOFT
14:58:... Malware_Build_... 1884 RegQueryValue HKLM\SOFT
14:58:... Malware_Build_... 1884 RegCloseKey HKLM\SOFT
14:58:... Malware_Build_... 1884 RegOpenKey HKCU\Softw
14:58:... Malware_Build_... 1884 RegOpenKey HKLM\Syste
14:58:... Malware_Build_... 1884 RegOpenKey HKLM\Syste
14:58:... Malware_Build_... 1884 RegSetInfoKey HKLM\Syste
14:58:... Malware_Build_... 1884 RegQueryValue HKLM\Syste
14:58:... Malware_Build_... 1884 RegOpenKey HKLM\Syste
14:58:... Malware_Build_... 1884 RegOpenKey HKLM\Syste
14:58:... Malware_Build_... 1884 RegSetInfoKey HKLM\Syste
14:58:... Malware_Build_... 1884 RegQueryValue HKLM\Syste
14:58:... Malware_Build_... 1884 RegQueryValue HKLM\Syste
14:58:... Malware_Build_... 1884 RegCloseKey HKLM\Syste
14:58:... Malware_Build_... 1884 RegOpenKey HKLM
14:58:... Malware_Build_... 1884 RegQueryKey HKLM
14:58:... Malware_Build_... 1884 RegOpenKey HKLM\Softw
14:58:... Malware_Build_... 1884 RegQueryKey HKLM
14:58:... Malware_Build_... 1884 RegCreateKey HKLM\SOFT
14:58:... Malware_Build_... 1884 RegSetInfoKey HKLM\SOFT
Event Properties
Event Process Stack
Date: 21/04/2024 14:58:09
Thread: 1812
Class: Registry
Operation: RegSetInfoKey
Result: SUCCESS
Path: HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\Curren
Duration: 0.0000006
KeySetInformationClass: KeySetHandleTagsInformation
Length: 0


File Edit Event Filter Tools Options Help
Process Name PID Operation
Malware_Build_... 2600 CreateFile
Malware_Build_... 2600 WriteFile
Event Properties
Event Process Stack
Date: 21/04/2024 15:54:02
Thread: 1100
Class: File System
Operation: CreateFile
Result: SUCCESS
Path: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.
Duration: 0.0001575
Desired Access: Generic Write, Read Attributes
Disposition: OverwriteIf
Options: Synchronous IO Non-Alert, Non-Directory File
Attributes: N
ShareMode: Read, Write
AllocationSize: 0
OpenResult: Overwritten


Process Name PID Operation
Malware_Build_... 2600 CreateFile
Malware_Build_... 2600 WriteFile
Event Process Stack
Date: 21/04/2024 15:54:02
Thread: 1100
Class: File System
Operation: CreateFile
Result: SUCCESS
Path: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.
Duration: 0.0001575
Desired Access: Generic Write, Read Attributes
Disposition: OverwriteIf
Options: Synchronous IO Non-Alert, Non-Directory File
Attributes: N
ShareMode: Read, Write
AllocationSize: 0
OpenResult: Overwritten

Possibile funzionamento del malware:

- Il malware si avvia all'orario specificato.
- Modifica il contenuto della cartella in cui è situato per nascondere tracce o per installare componenti aggiuntivi.
- Crea una nuova chiave di registro per mantenere persistenza o per configurare il sistema secondo le sue necessità.
- Utilizza la libreria DLL msgina32.dll per svolgere funzioni specifiche, il che potrebbe indicare che il malware sta cercando di interagire con il processo di autenticazione di Windows, ad esempio per ottenere credenziali o per manipolare il processo di accesso.
- Utilizza la chiave di registro creata per memorizzare informazioni critiche, come configurazioni, parametri o istruzioni da eseguire.