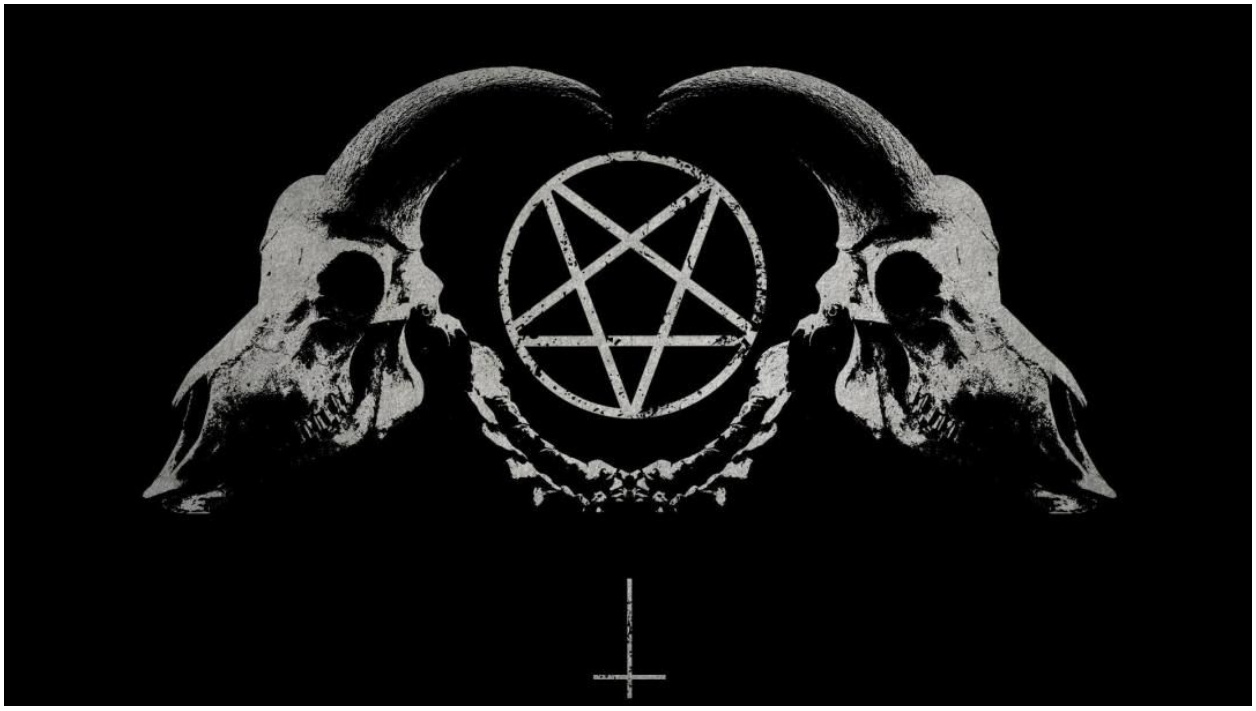
Covert Data Persistence with Windows Registry Keys

vx-underground.org collection // Jackson T.



Introduction

This article introduces a more subtle way to hide data within registry keys. It avoids detection from RegHide because it leverages class attributes instead of null characters.

Malware implants often need to persist data including configurations, encryption keys, modules, and so on. This can be done with varying degrees of sophistication, ranging from XOR-encrypted files to leveraging covert file systems, NVRAM variables, or secure enclaves.

The Poweliks and Kovter malware families would hide data by placing a null character in registry key value names which raises an error when viewed in RegEdit.

Technique

Storing Data:

The NtCreateKey routine is used below to create a new key with our secret buffer stored in the class name. This seems to work for buffers up to 64 KiB which is the maximum size of the Length member in UNICODE_STRING objects.

```
// Initialize OBJECT_ATTRIBUTES for the key.
UNICODE_STRING ObjectName = { 0 };
OBJECT_ATTRIBUTES ObjectAttributes = { 0 };
RtlCreateUnicodeString(&ObjectName, LR"(\REGISTRY\MACHINE\Software\Classes\{GUID})");
InitializeObjectAttributes(
        &ObjectAttributes,
        &ObjectName,
        OBJ_CASE_INSENSITIVE,
        NULL,
        NULL);

// Place the secret buffer (lpBuffer) in UNICODE_STRING.
UNICODE_STRING Class = { 0 };
Class.Length = dwBufferSize;     // buffer len
Class.Buffer = (PWSTR)lpBuffer; // buffer ptr

// Create the key with the secret buffer.
HANDLE KeyHandle = NULL;
NTSTATUS Status = NtCreateKey(
        &KeyHandle,
        KEY_ALL_ACCESS,
        &ObjectAttributes,
        NULL,
        &Class, // Supply UNICODE_STRING here.
        NULL,
        NULL);
```

Retrieving Data:

The Windows API appears to support retrieval of the key's class with the RegQueryInfoKey routine, however that failed to return the class value and length on tested systems. Instead, NtQueryKey can be used to retrieve the data:

```
// Retrieve the stored buffer via NtQueryKey.
ULONG ResultLength = 0;
NtQueryKey( // Get structure size first.
        KeyHandle,
        KeyNodeInformation,
        nullptr, NULL,
        &ResultLength);

// Allocate the PKEY_NODE_INFORMATION struct.
PKEY_NODE_INFORMATION KeyNodeInfo = (PKEY_NODE_INFORMATION)HeapAlloc(
        GetProcessHeap(),
        HEAP_ZERO_MEMORY,
        ResultLength);

// Populate structure with data.
NTSTATUS Status = NtQueryKey(
        KeyHandle,
        KeyNodeInformation,
        KeyNodeInfo,
        ResultLength,
        &ResultLength);

// Save data into its own buffer.
PBYTE Buffer = (PBYTE)HeapAlloc(
        GetProcessHeap(),
        HEAP_ZERO_MEMORY,
        KeyNodeInfo->ClassLength);

CopyMemory(
        Buffer,
        (PBYTE)((ULONGLONG)KeyNodeInfo + KeyNodeInfo->ClassOffset),
        KeyNodeInfo->ClassLength);
```

Tradecraft Considerations and Detection

1. Whereas key names with null characters can raise errors in RegEdit, this technique does not raise any. Key names, value names, and value data are not tampered.
2. A new key is created for each stored buffer in the above example. An example of blending in may involve creating GUID-named keys where many GUID-named sibling keys already exist. The keys should also share similar security descriptors, dates, and values as their siblings.
3. Backdooring an existing key with class data is possible, but does not appear to be as trivial as I thought because the data can only be provided upon the creation of a key.
4. Class names are usually UTF-16 strings, so while arbitrary byte buffers are possible, they may appear anomalous.
5. All data should be stored encrypted, where appropriate.

Acknowledgements

Thank you to @gsx0r1, @enigma0x3, and Lincoln for their reviews.