



# DOCUMENTATION

# 1. Preamble

---

CERTitude is a network-based host scanner that enables searching for *Indicator of Compromise* (IOC) on a large scale from a single workstation.

The tool can be broken down into two main components:

- The Python-driven web interface that allow scan planning and result visualization;
- The IOC scanner, running in parallel, also written in Python.

## 2. Technology

---

### 2.1. Web interface operating and terminology

#### 2.1.1. Terminology

An **XMLIOC** represents an IOC under the XML-based OpenIOC format.

The **HostConfidential** setting prevents the target workstation from knowing what the investigator is searching for by retrieving the gathered data onto the investigator's workstation.

A **Configuration Profile** is the association of multiple **XMLIOC** and the **HostConfidential** setting set to *true* or *false*.

The **Windows Credential** element is an alias for the *{Domain, Username, Password}* triplet used to log onto remote targets.

A **Batch** is a set of targets sharing the same **Windows Credential** object, analyzed under a specific **Configuration Profile**. It is possible to achieve scan results history by duplicating a batch<sup>1</sup> and scanning its workstations again.

A **User** represents an investigator. Only users are able to log on the web interface or to launch an IOC scan<sup>2</sup>.

#### 2.1.2. Web interface role

The **web interface (frontend)** is designed so that the investigator can interact with its CERTitude instance's **database (backend)**. Though the database format may vary thanks to the use of an ORM, we will consider the case of a file-based SQLite database.

This interface is also used to display a graphical view of the scan results, under the *Visualization* tab:



---

<sup>1</sup> This feature does not exist for now: IP addresses must be manually entered again

<sup>2</sup> Due to the fact the Windows Credential sensitive data is encrypted within the database

### 2.1.3. Database structure

CERTitude uses the following SQL tables to store its data:

- **xmliocs** stores a Base-64 version of the OpenIOC structure;
- **configuration\_profiles**, **windows\_credential**, **batches** and **users** store the object described by the above terminology (§2.1.1);
- **queue** stores the metadata and state of the workstation scan requests posted from the web interface;
- **resultats** associates an element from the queue to a description of what the workstation was found to be;
- **ioc\_detections** links a result to the *IndicatorItem* element within an **XMLIOC** in order to display the indicators that have been discovered.

## 2.2. IOC scanner operating

The scanner operating can be summed up by the following steps:

- IOC parsing and host selection
- Connection and registration on the targets
- Data collection
- Data analysis
- Cleaning

### 2.2.1. IOC Parsing and host selection

The IOCs are retrieved by the scanner instance from the database where they are stored as a base64 version of the OpenIOC (XML-based) structure. They are parsed using the *ElementTree* Python library into a tree object and the trees are stored according to what *Configuration Profile* uses them.

Then, the scanner enters an infinite loop, performing the following actions.

A host is selected according to its priority (1 being the lowest and 99 the highest). The batch it belongs to is retrieved and so are the *Windows Credential* and *XMLIOC* objects.

### 2.2.2. Connection and registration on the targets

Using the a slightly modified Python version of the famous PsExec tool, an instance of the *RemoteCmd* class is created, attempting to perform a network SMB logon onto the remote workstation on port 445 using the provided credentials. Should it fail, the host would be temporarily abandoned and its priority as well as number of tries would be decreased.

Otherwise, *RemoteCmd* uses the previously built SMB connection to send an executable to the target to be registered as a service<sup>3</sup> using DCERPC. The executable is dropped in the first found writable share on the target, usually `Admin$`. This service is launched as SYSTEM for data collection to be as complete as possible and therefore the provided credentials must belong to a member of the target's *Administrators* group.

Finally, the new-born service opens three named pipes corresponding to the standard streams. It is now able to receive commands on the standard input and output their result to the two output streams. This remote session is maintained during the collection and analysis of the target.

### 2.2.3. Data collection

Python scripts have been written to gather all the elements of a specific type: services, processes, ARP entries (each one corresponding to an OpenIOC *IndicatorItem* category)...

These scripts are orchestrated through `collector.py` which is compiled with PyInstaller in a single .exe file. Some of the OpenIOC *IndicatorItem* categories could not be queried through Python interfaces or were too slow and were therefore written in C.

---

<sup>3</sup> CERTitude-RCS

The data collection executables are sent to the remote workstation along with some utilities<sup>4</sup> through the existing SMB connection and each category will be assigned a remote SQLite database to store its elements into. For each item to be searched, if its category's database does not yet exist, all of the elements of this category are gathered and output in a single tab-delimited file. The file is then imported in the SQLite database and adjustments are made in some cases<sup>5</sup>.

#### 2.2.4. Data analysis

Depending on the HostConfidential setting, this phase can be performed on the remote workstation or on the investigator's workstation.

##### 1) HostConfidential = false

Queries are crafted on the investigator's computer and sent to the remote SQLite3 utility. Results are sent back to the scanner instance and detected items are stored in CERTitude's database.

##### 2) HostConfidential = true

Remote SQLite databases created during the data collection phase are retrieved by the scanner instance and locally queried. Results are analyzed and detected items are store in CERTitude's database

#### 2.2.5. Cleaning

When the host scan is completed, every dropped file is deleted. The service is unregistered, and the SMB session gracefully closed.

---

<sup>4</sup> e.g. sqlite3.exe

<sup>5</sup> e.g. the FileItem category

## 3. Using CERTitude

---

### 3.1. Installation prerequisites

Because most of CERTitude components are written in **Python**, you must have a **recent enough installation** (at least 2.7.9), preferably **32-bits**<sup>6</sup>.

All the libraries required to **run** CERTitude may be found under the dist directory. If you don't know how Python libraries work, you can copy/paste the following snippet:

```
> pip install -r requirements.txt
> pycrypto-2.6.1.win32-py2.7.exe
> install-pyopenssl.bat
```

**If you are using an authenticated proxy, *pip* won't be able to contact the *PyPi* website to download the modules. First setup an anonymous relay proxy using tools such as *Cntlm*.**

**If you followed the previous step or you are behind an anonymous proxy, set the `http_proxy` and `https_proxy` environment variable to your proxy address. Then, use the `easy_install` command to download and install each of the modules listed in the `requirements.txt` file.**

If you wish to add your own data collection modules, you will need *PyInstaller*:

```
> easy_install pywin32
> easy_install pyinstaller
```

### 3.2. Installing CERTitude

Before anything, you need to initialize the database that will be used by your CERTitude instance.

Double click on the **init.bat** script to perform this action. You will be asked to choose the "seeker" password that will be used to login on the web interface. The master key<sup>7</sup> is also generated during this step.

---

<sup>6</sup> Any new module will be compiled using Python. Win32 Python installation ensures maximal compatibility with target workstations.

<sup>7</sup> See Annexe 1.4.1.2

### 3.3. Scanning prerequisites

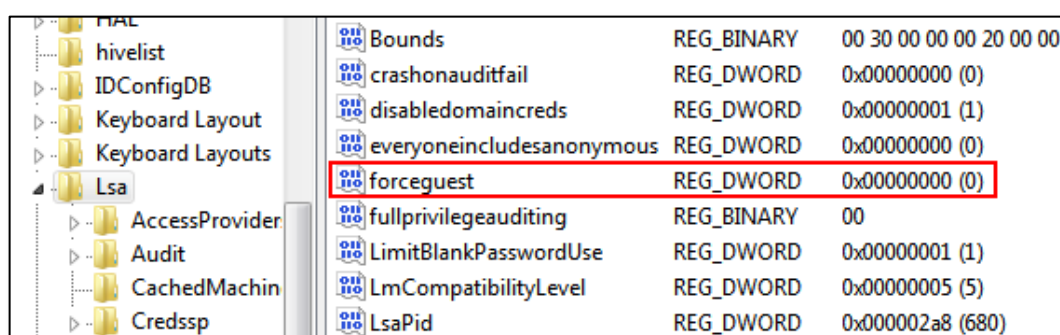
#### 1) Administrator account

In order to be able to connect to the remote workstations and to create a service that will be launched as SYSTEM, one must ensure that the credentials used to log onto the target belong to a member of the remote Administrators group.

This can be done in many ways, including:

- Using a local Administrator account deployed with the same password on all the targets<sup>8</sup>;
- Using a Domain Administrator account, whether one that already exists or one crafted for the occasion<sup>9</sup>.

If you intend to use a local account, please make sure that the following registry key is set to 0:



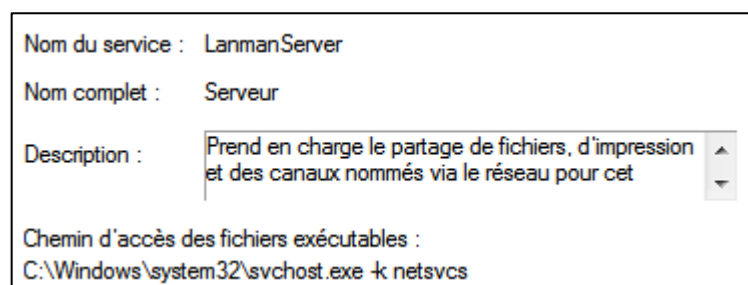
Name	Type	Value
Bounds	REG_BINARY	00 30 00 00 00 20 00 00
crashonauditfail	REG_DWORD	0x00000000 (0)
disabledomaincreds	REG_DWORD	0x00000001 (1)
everyoneincludesanonymous	REG_DWORD	0x00000000 (0)
<b>forcequest</b>	REG_DWORD	<b>0x00000000 (0)</b>
fullprivilegeauditing	REG_BINARY	00
LimitBlankPasswordUse	REG_DWORD	0x00000001 (1)
LmCompatibilityLevel	REG_DWORD	0x00000005 (5)
LsaPid	REG_DWORD	0x000002a8 (680)

KEY = HKLM\SYSTEM\ControlSet001\Control\Lsa\forcequest

#### 2) Administrative shares

The communication channel establishment requires being able to register a dropped executable as a service. Since the Administrator account takes care of the service part, you must provide a place for Python PsExec to drop the executable.

This can be done by enabling (if not already done) the use of administrative shares (c\$ and admin\$) through the following service:



Nom du service :	LanmanServer
Nom complet :	Serveur
Description :	Prend en charge le partage de fichiers, d'impression et des canaux nommés via le réseau pour cet
Chemin d'accès des fichiers exécutables :	C:\Windows\system32\svchost.exe -k netsvcs

<sup>8</sup> Highly not recommended since *Pass-the-hash* attacks cannot get any easier in that case

<sup>9</sup> In this case, it is advised to design a Domain group that is added *via* GPO to the local Administrators group instead of adding the crafted account by GPO



### 3) Remote firewall configuration

The analyst workstation must be able to establish an SMB connection to the remote target. Therefore, please make sure that the following ports are not filtered from the CERTitude host:

- TCP/139
- TCP/445

## 3.4. Searching for IOC with CERTitude

### 1) Interface

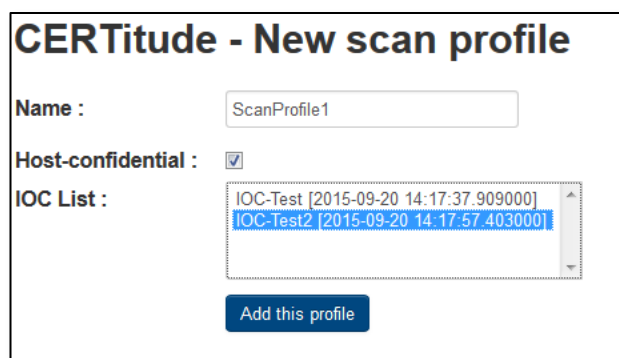
- Launch the interface using “interface.bat”
- Add IOCs



**CERTitude - New IOC**

Name :

- Add a scan profile



**CERTitude - New scan profile**

Name :

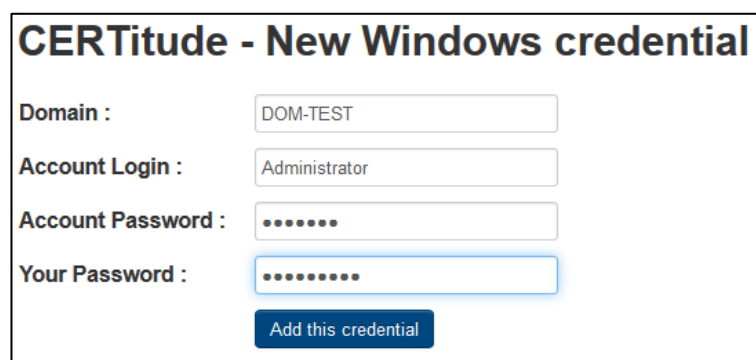
Host-confidential : ☒

IOC List : 

IOC-Test [2015-09-20 14:17:37.909000]

IOC-Test2 [2015-09-20 14:17:57.403000]

- Add new credentials



**CERTitude - New Windows credential**

Domain :

Account Login :

Account Password :

Your Password :

- Add a new batch

### CERTitude - Scan planification: Add a new batch

Name:

Profile:

Credentials:

- Add targets

### CERTitude - Scan planification - Batch1

Target:  OR

Subnet information:

Profile:

Credentials:

Priority:

Number of tries:

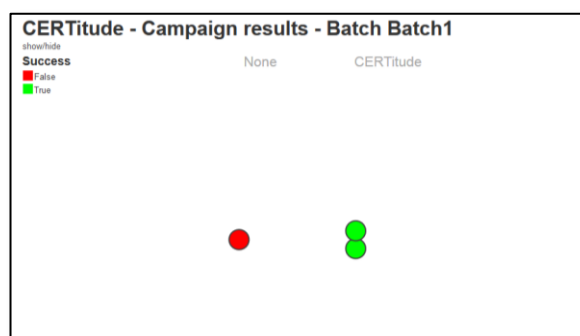
Force scan: ☐

## 2) IOCscan

- Launch "iocscan.bat" to start a scanner instance
- Repeat the previous step as many times as you want for multiple scanner instances
- Log onto each scanner instance to start scanning
- Close scanner instances when you want

## 3) Interface

- Visualize results



## 4. Security considerations

---

### 4.1. Retrieving the administrator account passwords...

#### 4.1.1. ... from one of the endpoints?

Because CERTitude uses only NTLMv2 network logon (challenge/response) for all the registration/connection process, one cannot use tools such as *Mimikatz* to collect the password from memory from one of the endpoints.

#### 4.1.2. ... from the database?

It is an established fact that whatever the encrypted form the Administrators accounts' password are stored in the database in, they must be available in clear text form during the scan.

In order to prevent non-users from accessing the credentials, we considered the following encryption/decryption functions:

- `ENC = IV || AES_256_CBC ( PAD ( CLEAR ), KEY, IV )`
- `CLEAR = UNPAD( AES_256_CBC-1( ENC_WO_IV10, KEY, IV ) )`

A 256-bit Master Key is generated during the installation process and encrypted using the first user's password. While the latter form will be stored in the database, the cleartext form of the Master Key will only exist as a variable during a short period of time.

When adding new credentials, though already logged in, the user is asked to provide his password again. If correct<sup>11</sup>, it will be used to decrypt the Master Key, which in turn will be used to encrypt the credentials.

When adding a new user, the current user must provide his password so that the Master Key can be decrypted and encrypted with the new user's password.

Finally, when launching an IOC scan, a valid user/password combination is required. The Master Key is decrypted using the password and the credentials are decrypted using the Master Key.

This encryption scheme ensures the two following points:

- The cleartext Master Key will only exist in memory during a brief time period;
- Only registered users are able to decrypt the administrative credentials.

---

<sup>10</sup> Encrypted message without Initialization Vector (IV)

<sup>11</sup> Password hashed is checked against its salted SHA256 value

## 4.2. Network-level security

While the HostConfidential parameter ensures confidentiality towards the remote host (no IOC query is sent to that target), there is no protection from network eavesdroppers.

They can achieve:

- Real-time packet modification, since SMB is not signed by default;
- Interception of the collected data;
- ...

However, CERTitude is fully compatible with the IPSec encryption protocol which adds the desired encryption and integrity layers.

We recommend that you only add an IPSec rule that will cover traffic:

- Point-to-point (no IPSec tunnel)
- From the analyst host, from any port
- To any host in the target subnet, to ports TCP/139 and TCP/445
- Mirrored
- Using the best security parameters available compatible for both the analyst and the targets (e.g. Windows XP does not support high-level DH groups).

**Our test cases used Pre-Shared Keys to authenticate each party. While this is not the best method to use, it remains efficient if you cannot afford to create your PKI (or it has been compromised) / you do not want to include the analyst workstation in the Active Directory.**