GENETIC MALWARE

DESIGNING PAYLOADS FOR SPECIFIC TARGETS

@wired33
@midnite_runr

Ekoparty 2016

Who we are

- Travis Morrow
  - AppSec, Mobile, WebTesting, SecOps
- Josh Pitts
  - Author of BDF/BDFProxy
  - https://github.com/secretsquirrel
  - AppSec, RedTeaming, WebTesting, SecOPs

How we got here…


Dude,
I have this algo…


Awesome
Let's do it..

If you write Malware
you have four
enemies (besides LE)

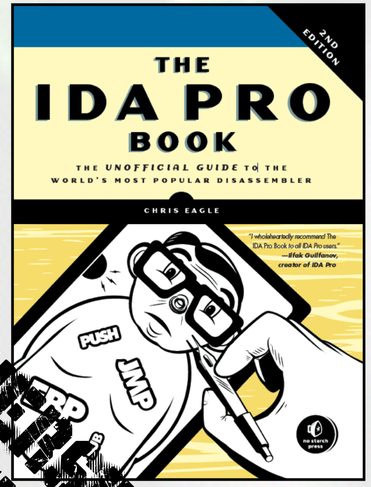
Conduct Operations

If you write ~~Malware~~ M̂alware you have four enemies (besides LE)

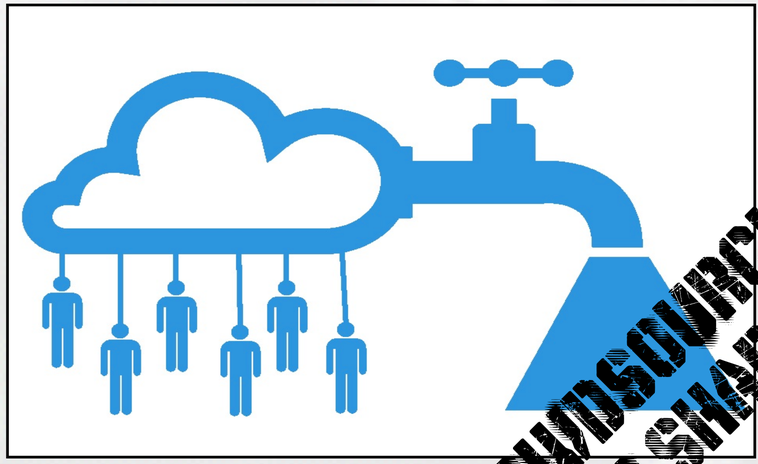
ANTI-VIRUS

AUTOMATED SANDBOX

REVERSE ENGINEER



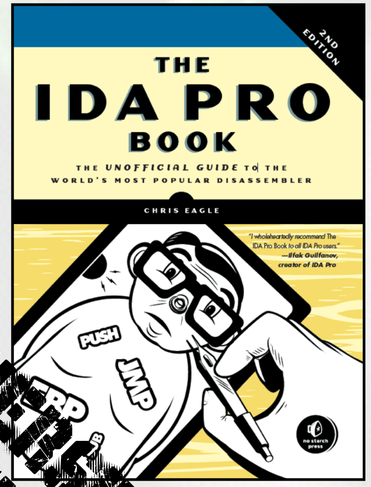
CROWDSOURCING INFO SHARING



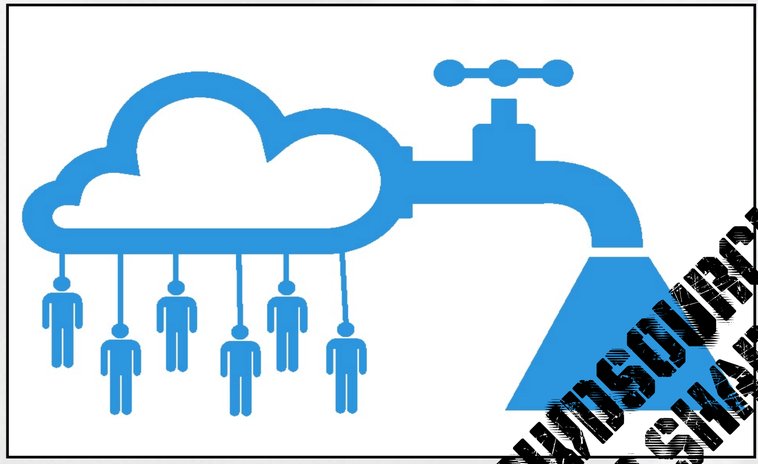
ANTI-VIRUS

AUTOMATED SANDBOX

REVERSE ENGINEER



CROWDSOURCING INFO SHARING



- Including Consumer Grade Products
- Founded by the Charlie Sheen of our industry
- Easy to bypass, not really a concern
- Can make you more vulnerable
- Respect for F-Secure and Kaspersky

- Including Consumer Grade Products
- Founded by the Charlie Sheen of our industry
- Easy to bypass, not really a concern
- Can make you more vulnerable
- Respect for F-Secure and Kaspersky

However…

Kaspersky is reporting that go1.6.2.windows-amd64.msi contains Trojan Ebowla #16292

Open **kaveh256** opened this issue on Jul 7 · 10 comments

kaveh256 commented on Jul 7 • edited

Kaspersky is reporting that there is a Trojan inside go1.6.2.windows-amd64.msi. See the report here. Looking further into it, it seems it considers api.exe to contain Trojan.Win32.Ebowla.

This seems to be a recurring issue. This also happens with version 1.5 with vet.exe and pprof.exe and also a few previously filed issues.

👍 1 😄 2

Kaspersky detecting Trojan.Win32.Ebowla.bn in docker.exe

Docker for Windows

neuneu2k Josselin Pujo — Jun 29

Expected behavior

Installation works without triggering antivirus

Actual behavior

Kaspersky IS detects malicious code in the docker.exe binary

VirusTotal has one corroborating detection:
https://www.virustotal.com/en/file/82e119c4d1c8b07719280c77acf3cab624362d6165e01781a2b193fe1f7bfa34/analysis/ 42

I'm looking at the Ebowla go code to see if there is a reason for docker to generate a false positive, not posting an issue on github until I have done more footwork...

Kaspersky detecting Trojan.Win32.Ebowla.bn in docker.exe

Docker for Windows

neuneu2k Josselin Pujo Jun 29

Full Version: Kaspersky deletes GO programming language executables

Kaspersky Lab Forum > English User Forum > Protection for Home Users > Kaspersky Internet Security & Anti-Virus for Windows

nick999999999

Kaspersky just zapped several of the executables from the Go programming language system claiming they contained variants of Win32.Ebola.kx.
Seems unlikely as Go has been installed for ages and not used recently so concerned this may be a false positive.
Is anyone else having problems with Kaspersky attacking Golang files? See log below.

Nick

https://www.virustotal.com/en/file/82e119c4d1c8b07719280c77acf3cab624362d6165e01781a2b193fe1f7bfa34/analysis/ 42

I'm looking at the Ebowla go code to see if there is a reason for docker to generate a false positive, not posting an issue on github until I have done more footwork...

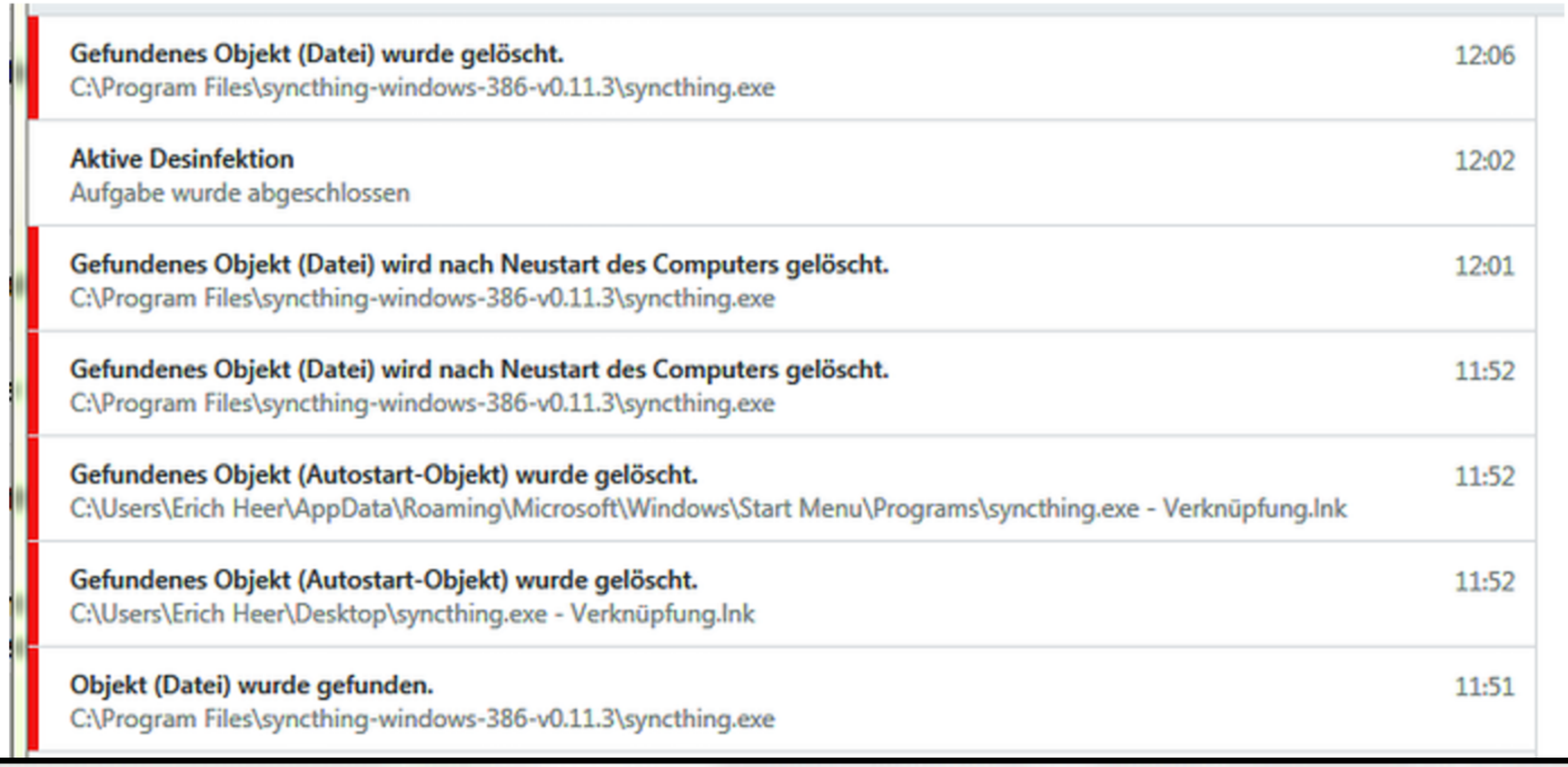
Kaspersky treats Syncthing as virus (false positive)

Support

Erich Heer Bug Reporter — 2 edits — Jun 19

This is a false positive.

// edit by @calmh



Kaspersky treats Syncthing as virus (false positive)

Support

Erich Heer Bug Reporter

2 Jun 19

/r/ethereum COMMENTS

8

Kaspersky detected Ebowla trojan from geth? i.gyazo.com

Submitted 3 months ago by DashHex

4 comments share

all 4 comments

sorted by: top (suggested)▼

[–] **hbhades** **4 points** 3 months ago

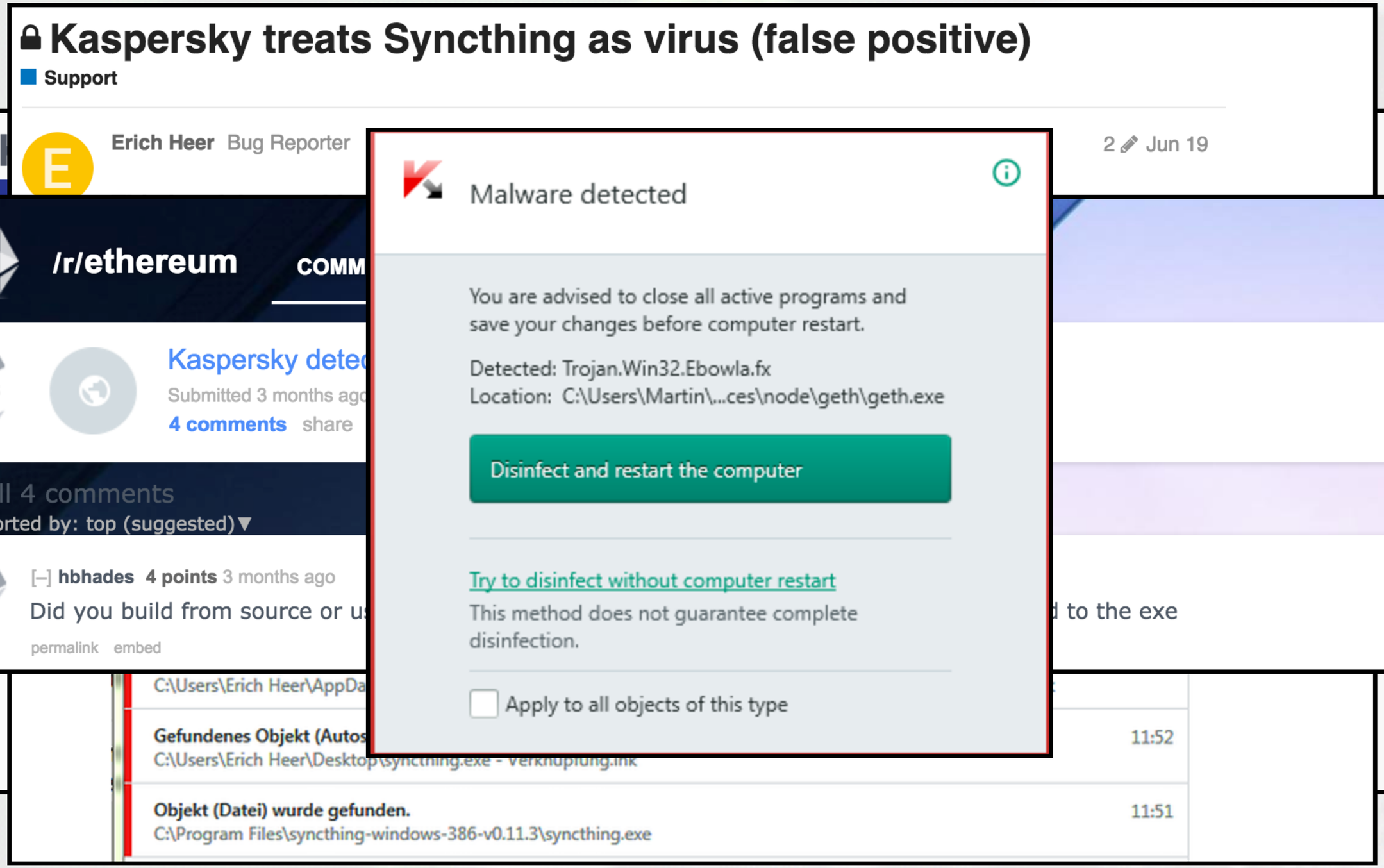
Did you build from source or use official binaries from github? If not, anything could be added to the exe

permalink embed

C:\Users\Erich Heer\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\syncthing.exe - Verknüpfung.lnk

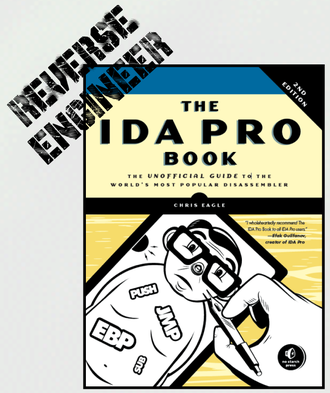
Gefundenes Objekt (Autostart-Objekt) wurde gelöscht. 11:52
C:\Users\Erich Heer\Desktop\syncthing.exe - Verknüpfung.lnk

Objekt (Datei) wurde gefunden. 11:51
C:\Program Files\syncthing-windows-386-v0.11.3\syncthing.exe

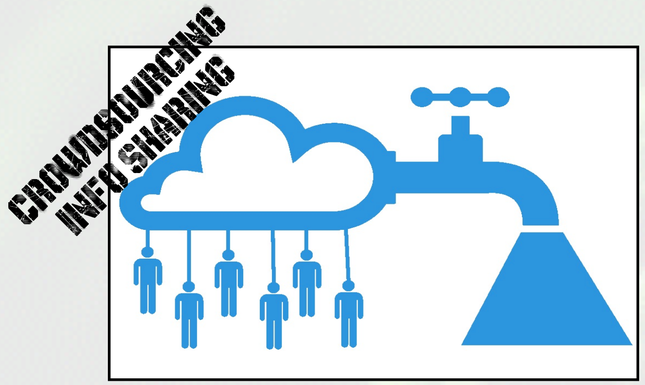

Kaspersky treats Syncthing as virus (false positive)
Support
Erich Heer Bug Reporter
2 Jun 19
/r/ethereum
COMM
Kaspersky detec
Submitted 3 months ago
4 comments share
all 4 comments
sorted by: top (suggested)
[–] hbhades 4 points 3 months ago
Did you build from source or u
to the exe
permalink embed
Malware detected
You are advised to close all active programs and save your changes before computer restart.
Detected: Trojan.Win32.Ebowla.fx
Location: C:\Users\Martin\...ces\node\geth\geth.exe
Disinfect and restart the computer
Try to disinfect without computer restart
This method does not guarantee complete disinfection.
Apply to all objects of this type
C:\Users\Erich Heer\AppDa
Gefundenes Objekt (Autos
C:\Users\Erich Heer\Desktop\syncthing.exe - Verknüpfung.lnk
11:52
Objekt (Datei) wurde gefunden.
C:\Program Files\syncthing-windows-386-v0.11.3\syncthing.exe
11:51

- Easy to bypass analysis
- A lot of machines are still XP
- They often:
  - Have unique ENV vars
  - Rarely change external IP
  - Have analysis timeouts

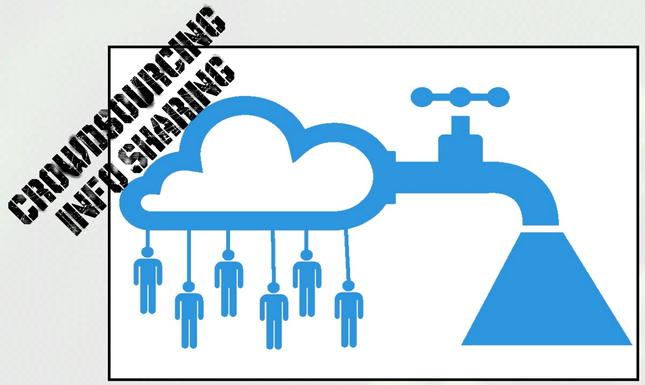
- Easy to bypass analysis
- A lot of machines are still XP
- They often:
  - Have unique ENV vars
  - Rarely change external IP
  - Have analysis timeouts



- Hard to defeat the Reverse Engineer (RE)
- Tricks that defeat AV and Automated Sandboxes != work on an experienced RE
- If malware payloads decrypt in memory on the RE’s machine, it can be analyzed
- At best you can only slow down the RE
- Turn RE into a password cracker and you win



- Kind of a MMO of Whack-A-Mole
- Magnifies the outcome of easy to fingerprint malware
- Defeat the RE and this becomes less effective



- Kind of a MMO of Whack-A-Mole
- Magnifies the outcome of easy to fingerprint malware
- Defeat the RE and this becomes less effective

Enter Environmental Keying

Enter Environmental Keying … a short primer

Clueless Agents

- Environmental Key Generation towards Clueless Agents (1998) – J. Riordan, B. Schneier
- Several methods for key sources:
  - Server required
    - Usenet
    - Web pages
    - (Forward|Backwards)-Time Hash Function
  - Host specific
    - Mail messages
    - File System
    - Local network

Clueless Agents

- Environmental Key Generation towards Clueless Agents (199[illegible] – J. Riordan, B. Schneier
- Several methods for key sources:
  - Server required
    - Usenet
    - Web pages
    - (Forward|Backwards) Time Hash Function
  - Hardware specific
    - Mouse movements
    - File system
    - Local network

NO POC

Secure Triggers

- Foundations for Secure Triggers (2003), Corelabs
  - Did not reference Clueless Agents
- Defeat REs and analysis
- Makes mention of OTP
- Lots of Math (too much)

Secure Triggers

- Foundations for Secure Triggers (2003), Corelabs
  - Did not reference Clueless Agents
- Defeat RAT analysis
- Makes mention of OTP
- Lots of Math (too much)

NO POC

Bradley Virus

- Strong Cryptography Armoured Computer Virus Forbidding Code Analysis (2004), Eric Filiol
  - References Clueless Agents
- Nested encrypted enclaves/payloads
- “Complete source code is not available”
- “[…]cause great concern among the antiviral community**. This is the reason why will not give any detailed code.**

Bradley Virus

- Strong Cryptography Armoured Computer Virus Forbidding Code Analysis (2004), Eric Filiol
  - References Clueless Agents
- Nested encrypted enclaves/payloads
- “Complete source code is not available”
- “[…] caused great concern among the antiviral community. **This is the reason why will not give any detailed code.**

NO POC

Hash and Decrypt

- Mesh design pattern: hash-and-decrypt (2007), Nate Lawson
- Application of secure triggers to gaming

Hash and Decrypt

- Mesh design pattern: hash-and-decrypt (2007), Matt Lawson
- Application of secure triggers to gaming

NO POC

Über-Malware

- Malicious Cryptography… Reloaded (CanSecWest 2008) – E.Filiol, F.Raynal
- New: Plausible Deniability!
  - Via OTP
- POC was a XOR

Über-Malware

- Malicious Cryptography Reloaded (CanSecWest 2008) – E.Filiol, F.Raynal
- New: Plausible Deniability!
  - Via GPG
- POC was just XOR

NO POC

Impeding Automation

- Impeding Automated Malware Analysis with Environmental-sensitive Malware (2012), Usenix,(C.Song, et al)
  - Did not reference Clueless Agents or the Bradley Virus
- Rediscovers Environmental Keying..
- Examples of Environmental keys
- Great Quotes:
  - “Due to time constraints..”
  - “[…]exceeds the scope of this paper,[…]
  - “At the inception of this paper, concerns were raised[…]”

Impeding Automation

- Impeding Automated Malware Analysis with Environmental Keying Malware (2012), Usenix (C.Song, et al)
  - Did not reference Clueless Agents or the Bradley Virus
- Rediscovers Environmental Keying.
- Examples of environmental keys
- Great Quotes
  - “[…] time constraints..”
  - “[…] exceeds the scope of this paper,[…]
  - “At the inception of this paper, concerns were raised[…]”

NO POC

Researchers have not released an open source environmental keying POC

Flashback (2011)

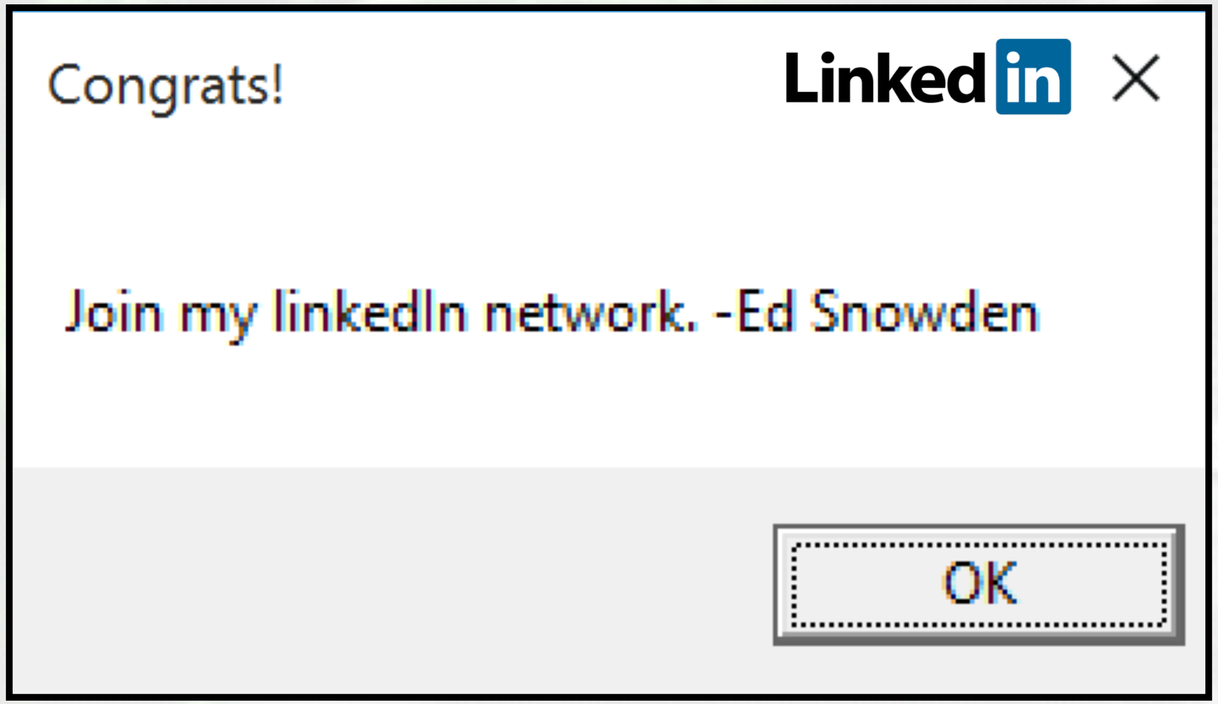
Flashback (2011)

- Mac OS X only malware
- Initial agent sent back UUID of OS to server
- Server used MD5 of UUID to encrypt payload
- Sent back to user and deployed

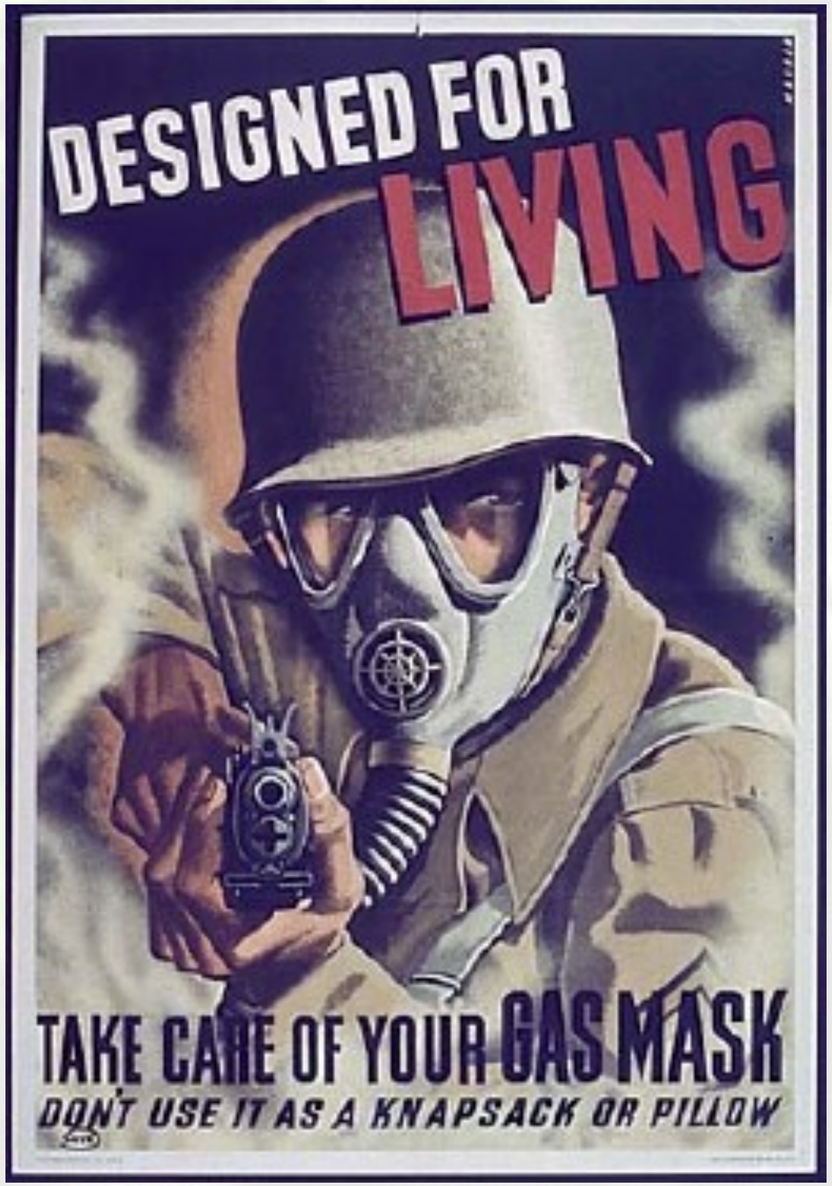
Gauss (2012)

Gauss (2012)

- Discovered by Kaspersky
- Encrypted Payload “Godel”
- Key derived from directory path in program files, MD5 hashed for 10k rounds
- Not publicly decrypted to date


Congrats!
LinkedIn
Join my linkedIn network. -Ed Snowden
OK

Targeted Malware Compared to Biological/ Chemical Agents


DESIGNED FOR
LIVING
TAKE CARE OF YOUR GAS MASK
DON'T USE IT AS A KNAPSACK OR PILLOW

Chemical Agents

- Area effect weapons
- Effective for days to weeks
- For targeting systems:
  - Domain specific env vars
  - External IP address
  - Check system time

Biological Agents

- Viral
- Genetic Targeting
- “Ethnic Weapons”
- For systems targeting:
  - Path
  - Particular file (OTP)
  - See Jacob Torrey’s Work at HITB 2016 on PUFs (https://conference.hitb.org/hitbsecconf2016ams/wp-content/uploads/2015/11/D1T1-Jacob-Torrey-Using-the-Observer-Effect-and-Cyber-Fengshui.pdf)

Targeted Malware and its use in Operations

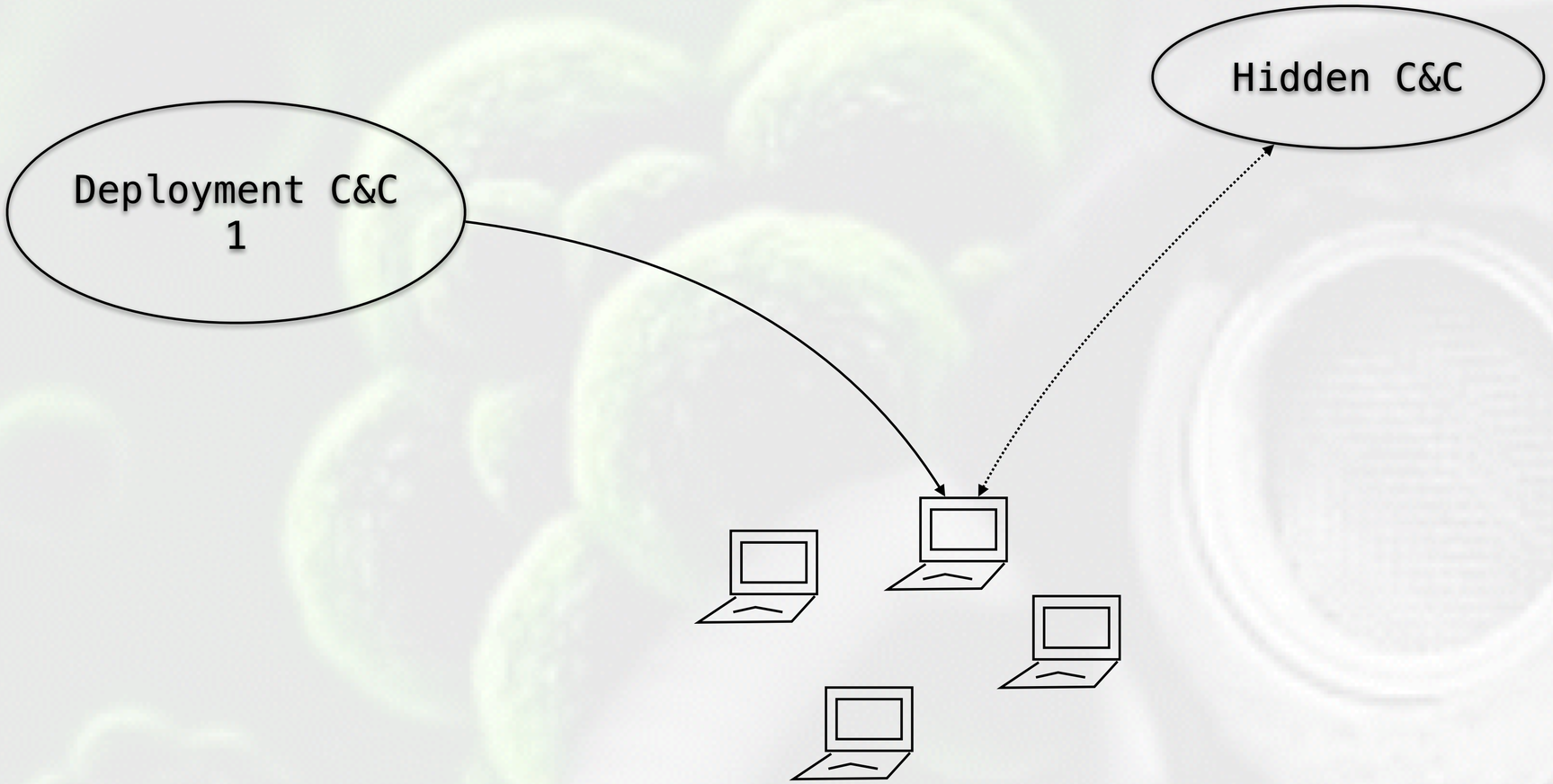
Deploy everywhere
work somewhere

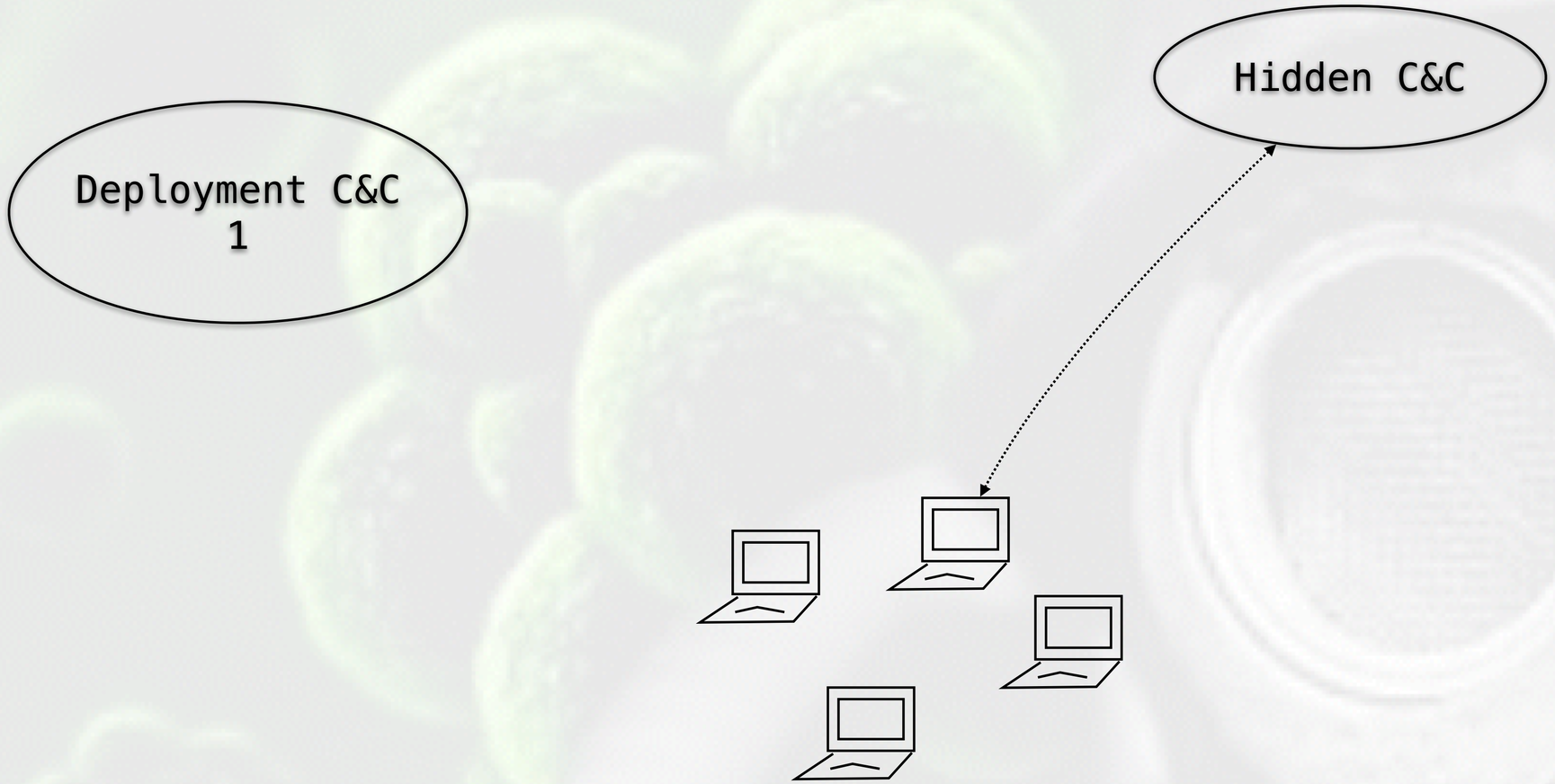
Operational plausible deniability

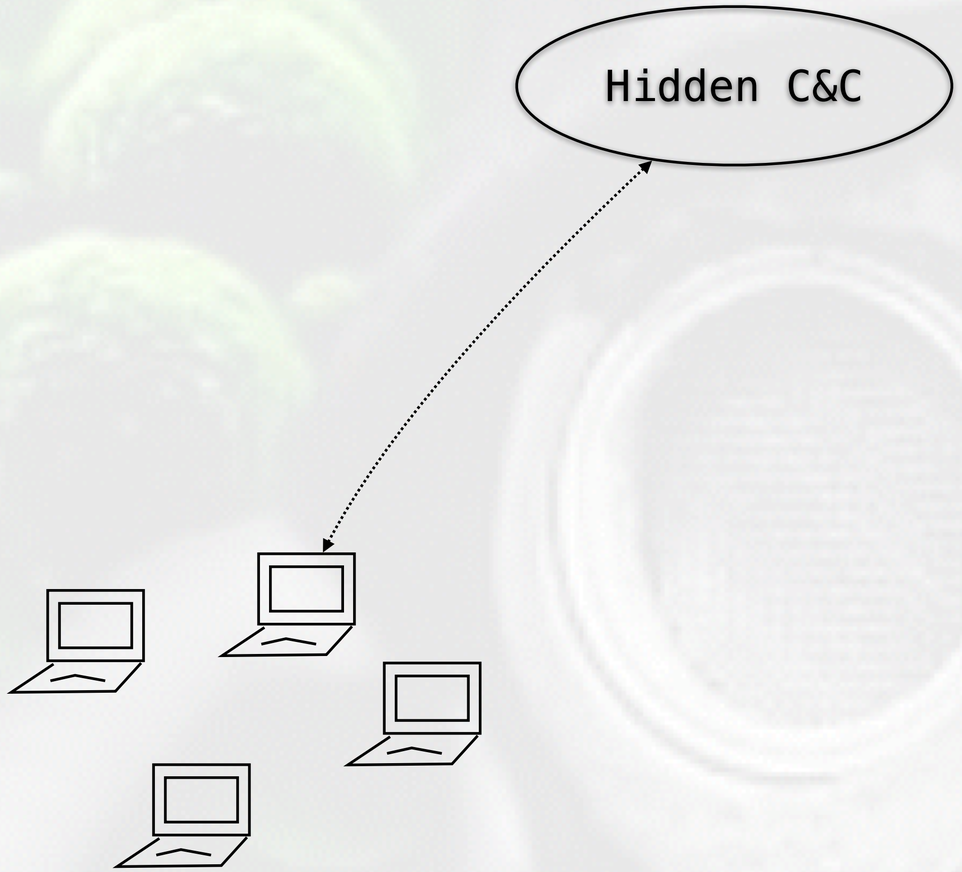
Hidden Command and Control (C&C)

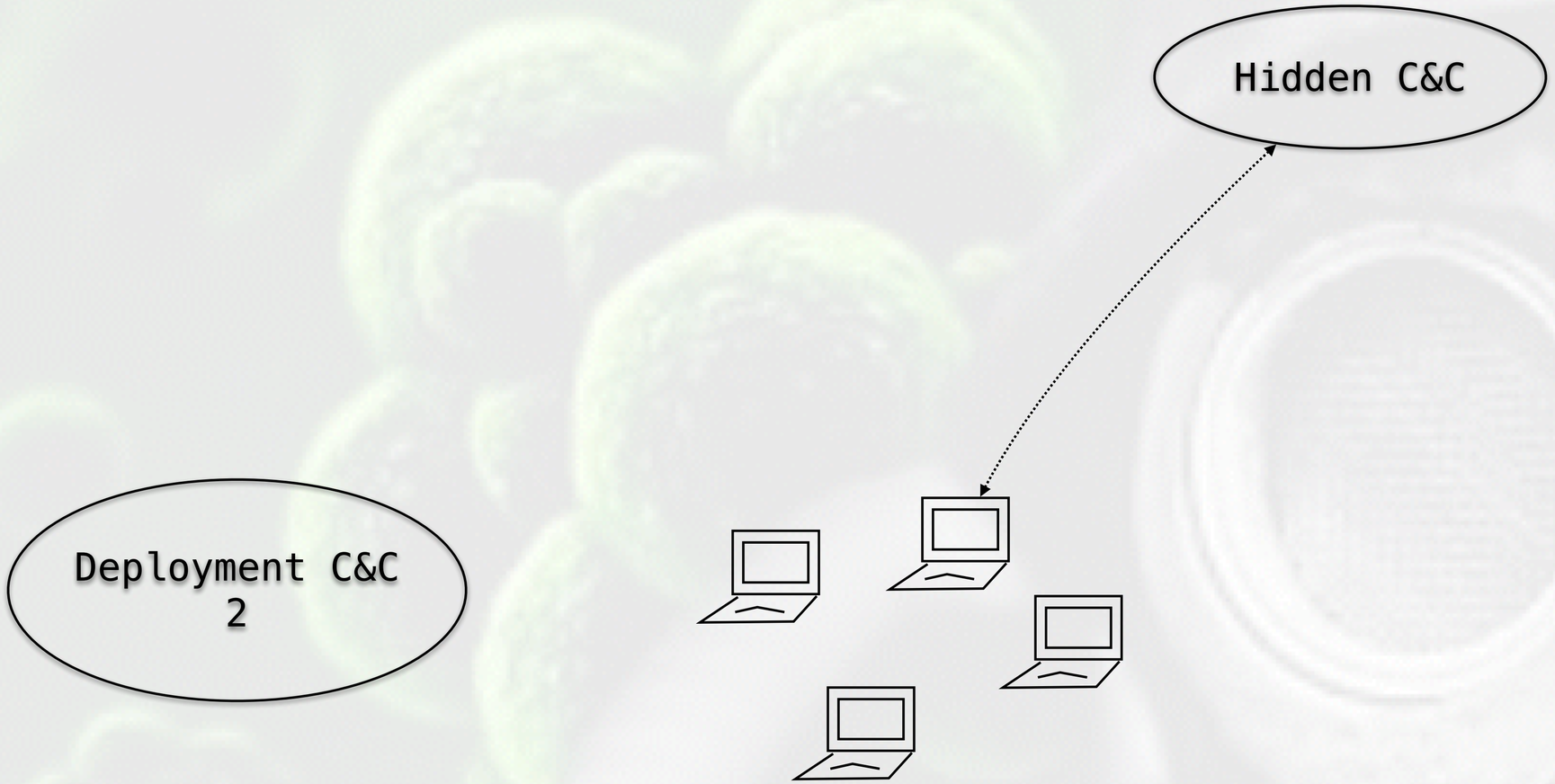
Hidden C&C

Deployment C&C
1


Hidden C&C
Deployment C&C
1

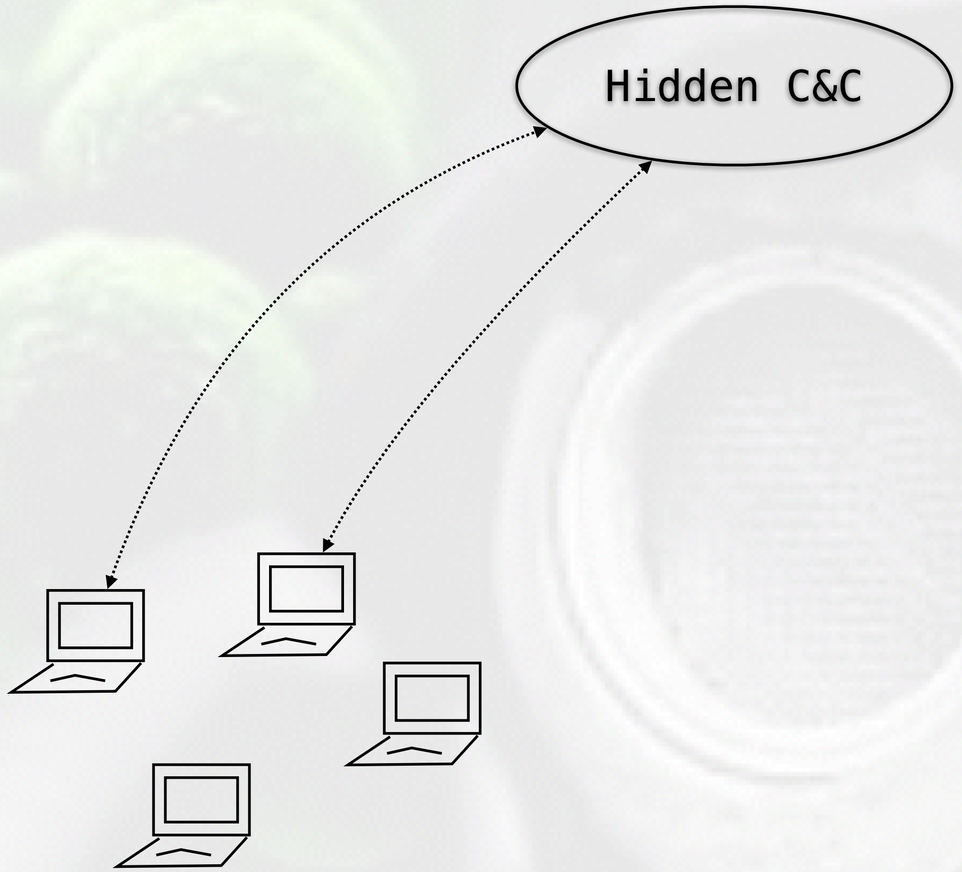

Hidden C&C
Deployment C&C
1

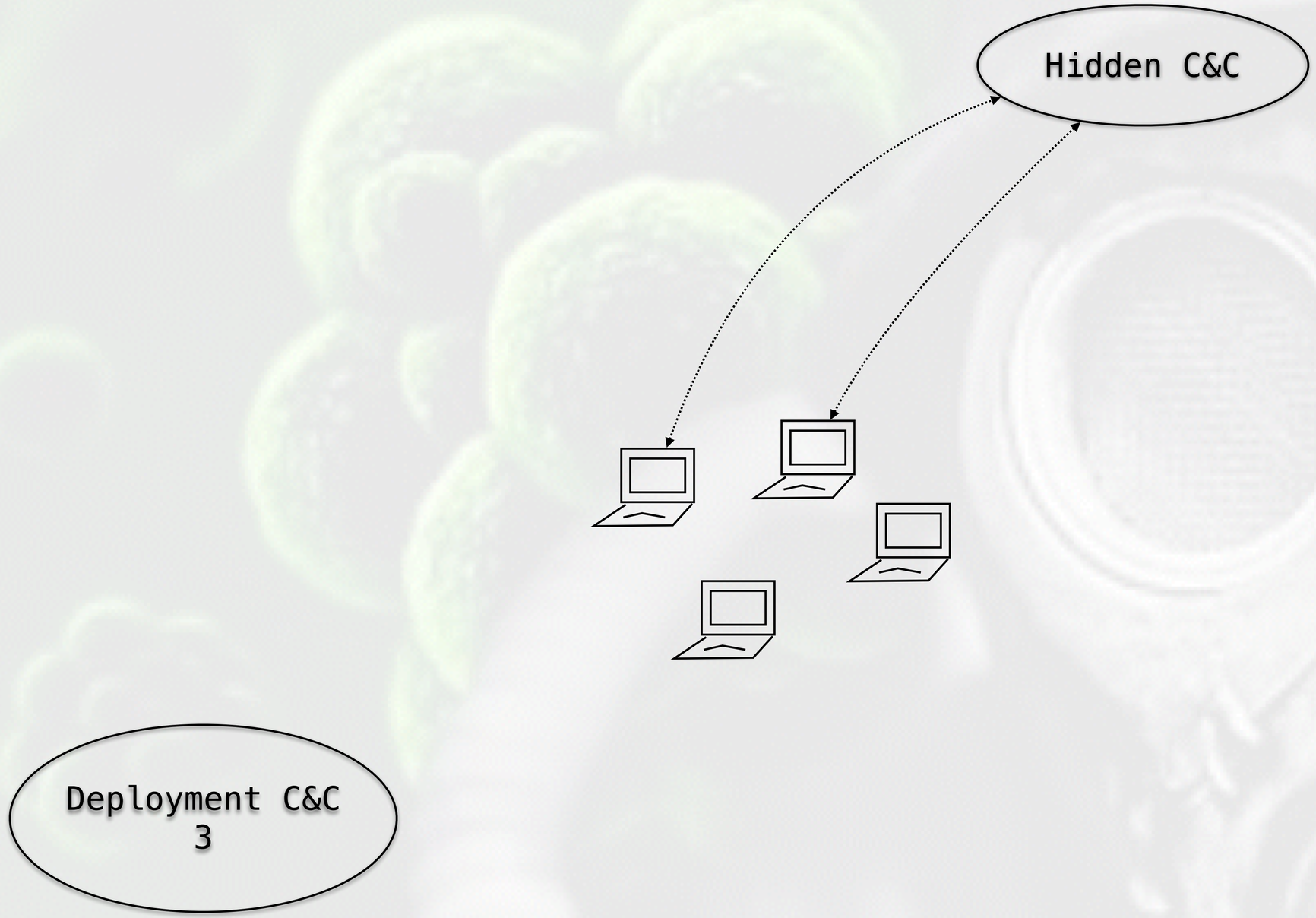

Hidden C&C

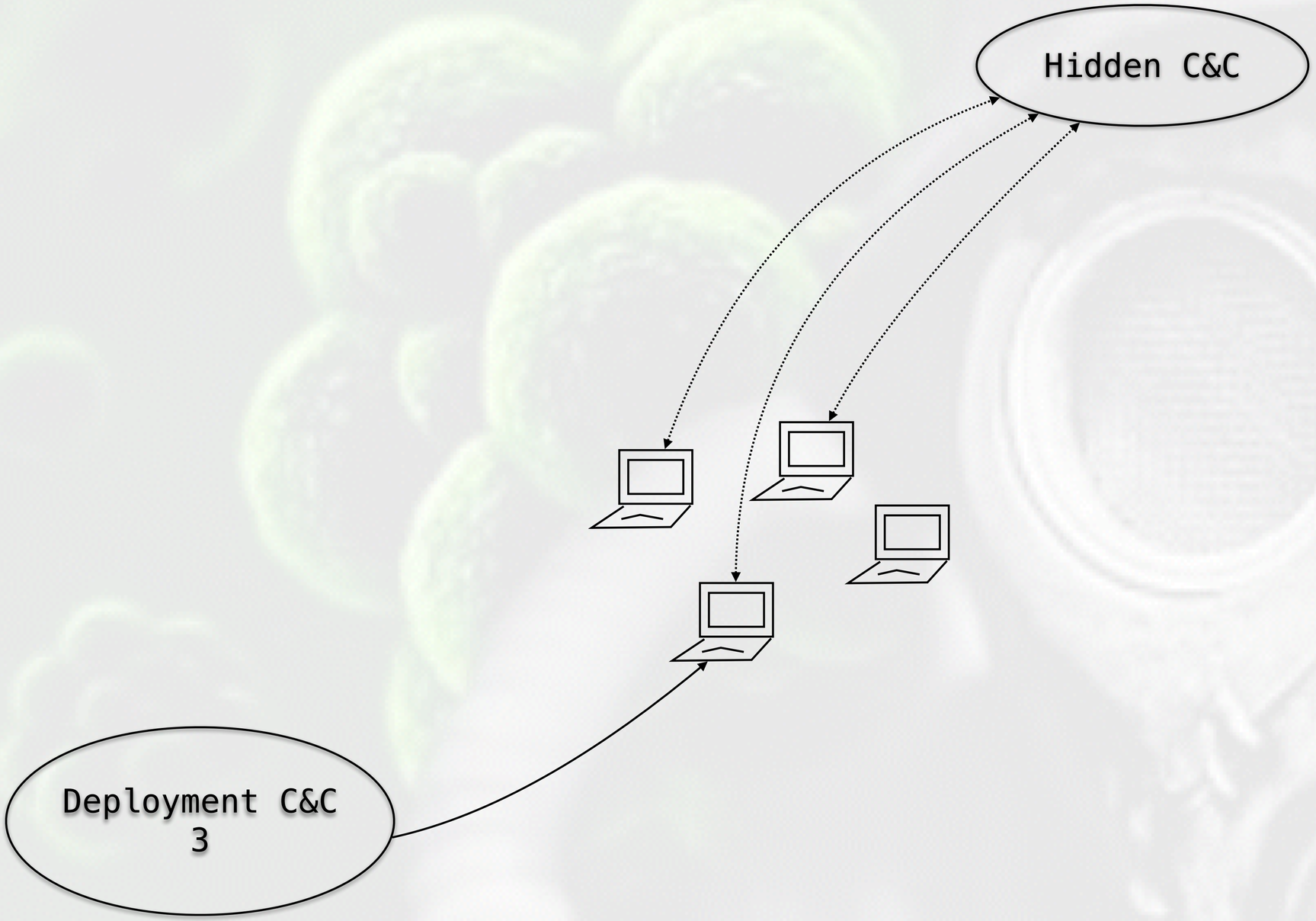

Hidden C&C
Deployment C&C
2

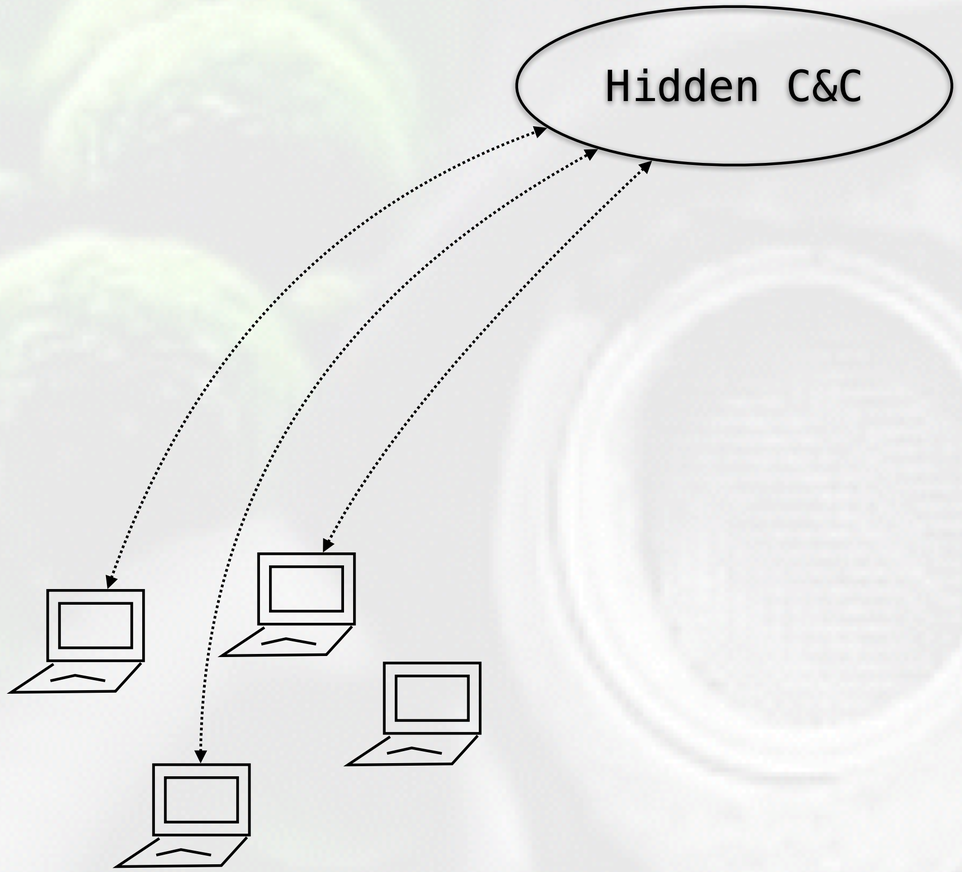

Hidden C&C
Deployment C&C
2


Hidden C&C
Deployment C&C
2

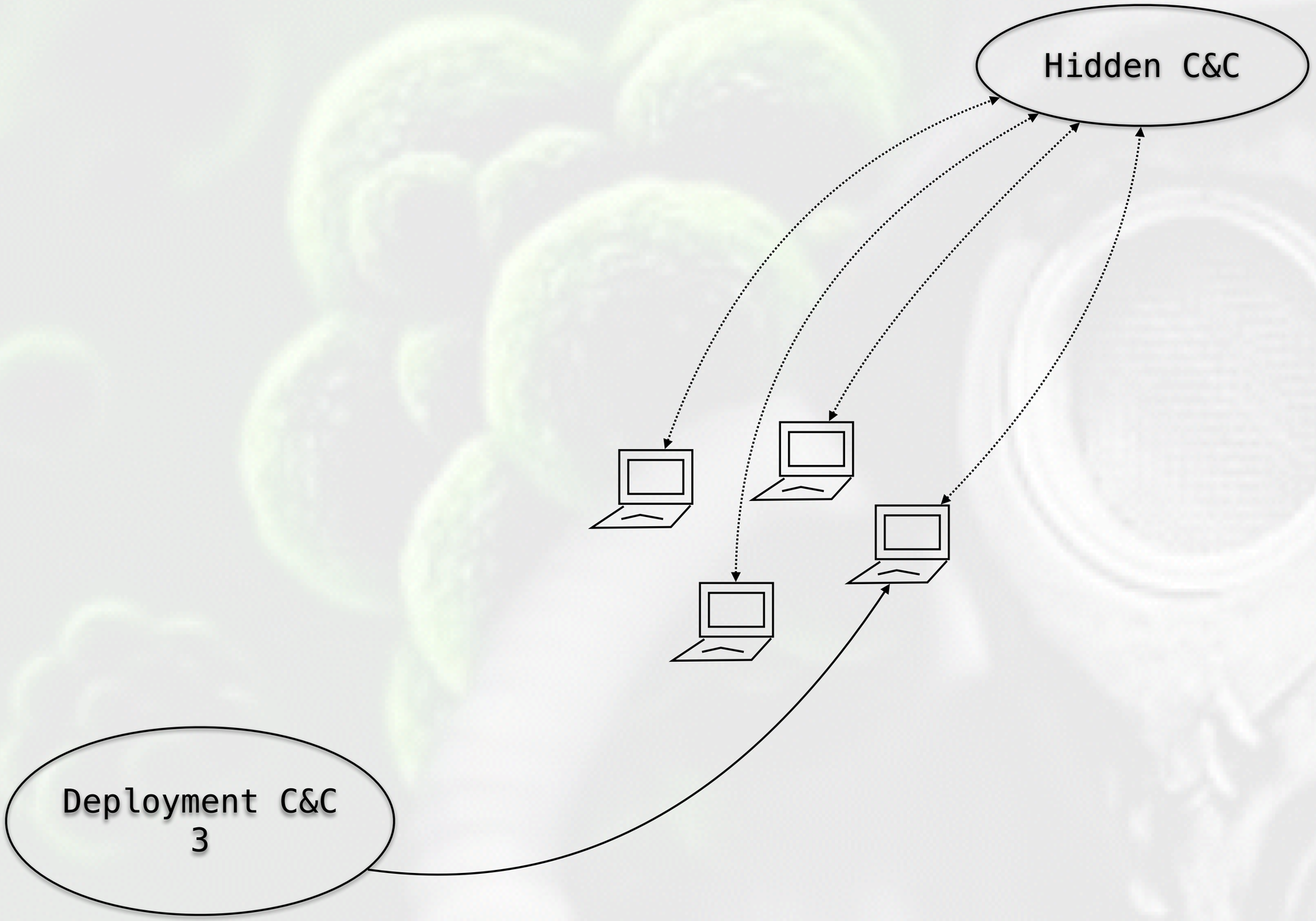

Hidden C&C

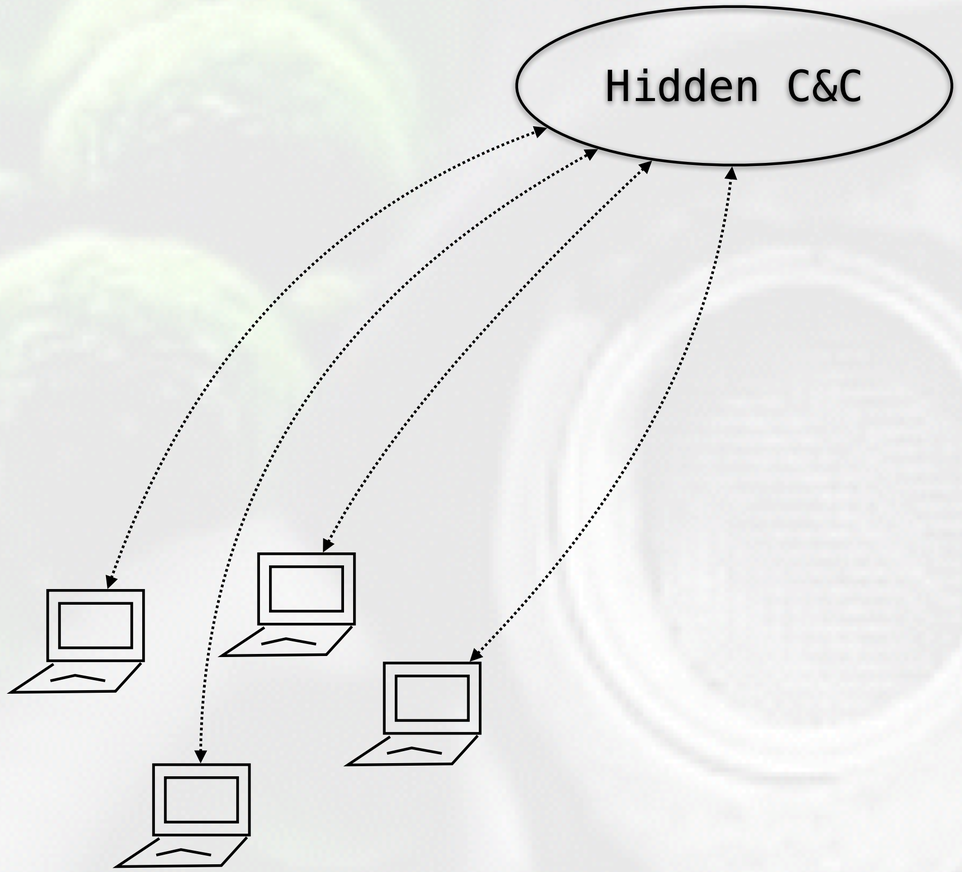

Hidden C&C
Deployment C&C
3


Hidden C&C
Deployment C&C
3



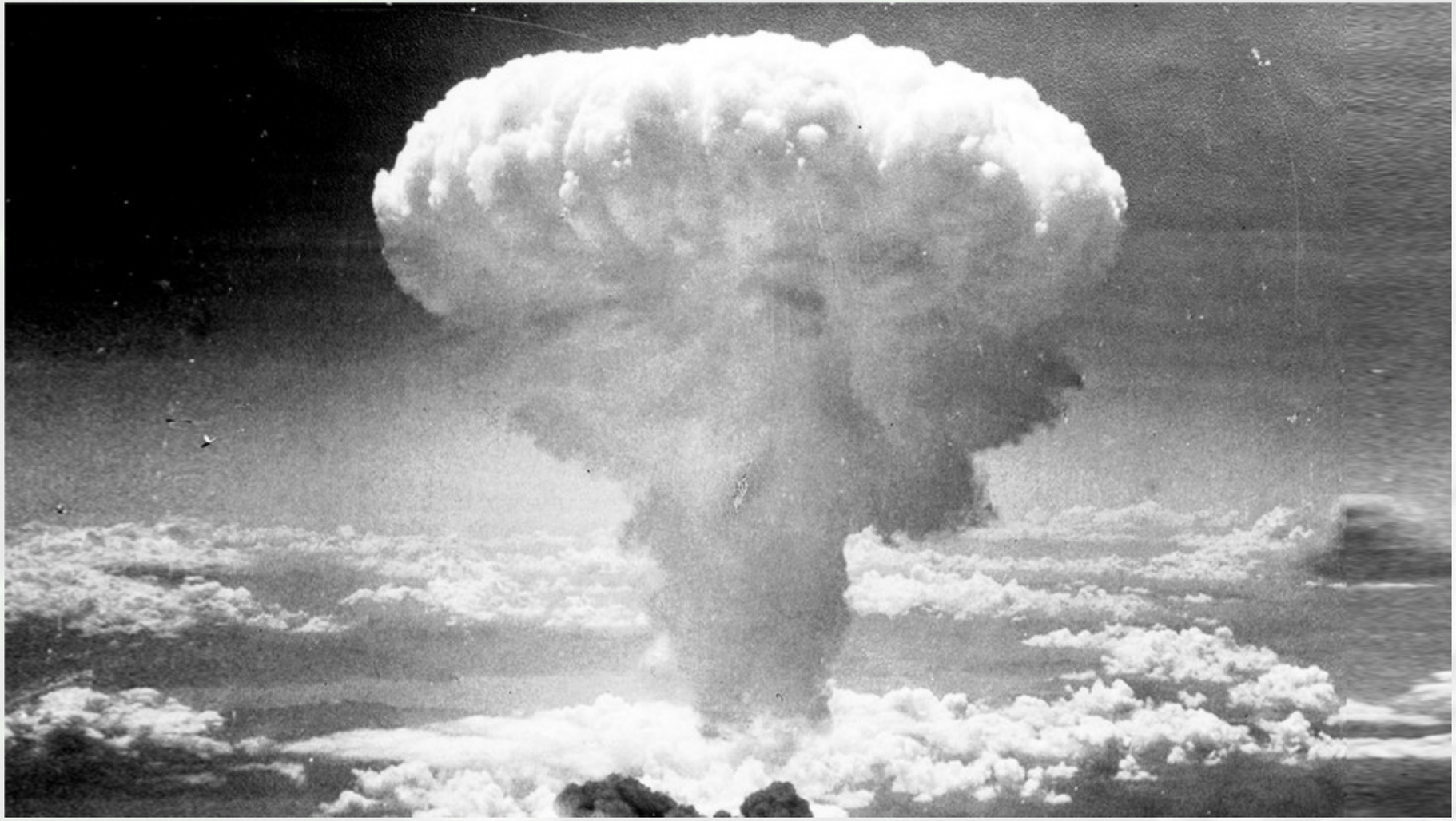
Deployment C&C
3


Hidden C&C
Deployment C&C
3



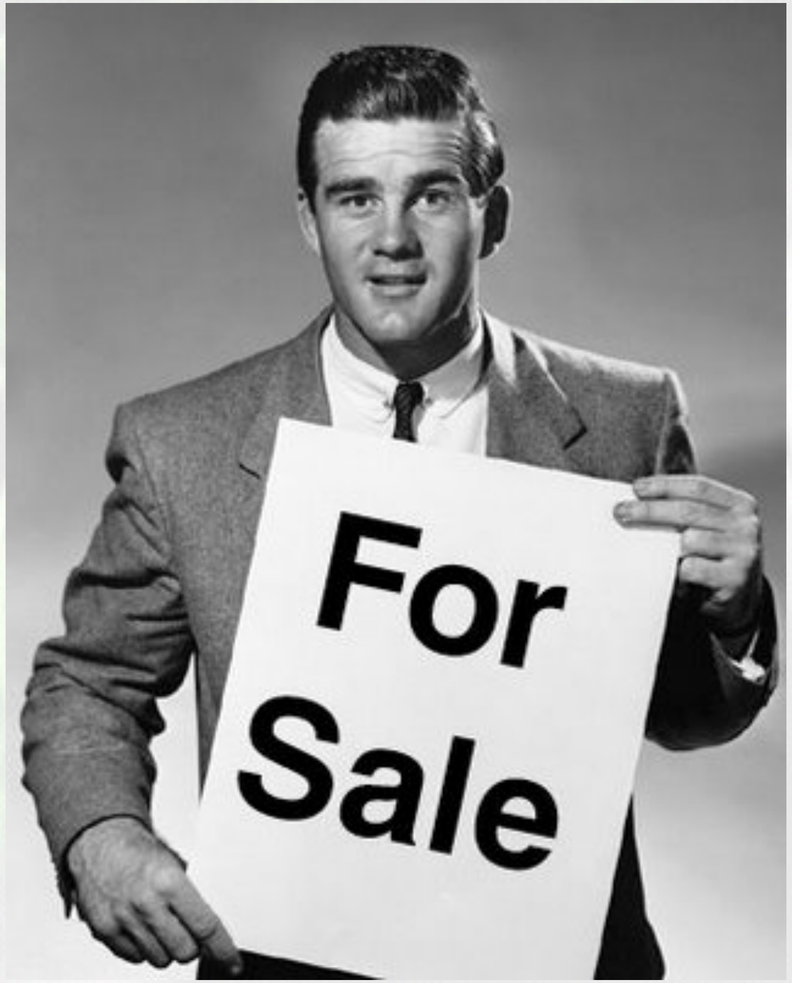
Deployment C&C
3

Could you imagine a world where all malware was targeted?



http://www.livescience.com/45509-hiroshima-nagasaki-atomic-bomb.html

https://s-media-cache-ak0.pinimg.com/564x/61/8b/52/618b52fcfefecb3eada6f7bb74e8a5bc.jpg



http://mattruple.theworldrace.org/blogphotos/theworldrace/mattruple/salesman.jpg

E.B.O.W.L.A.

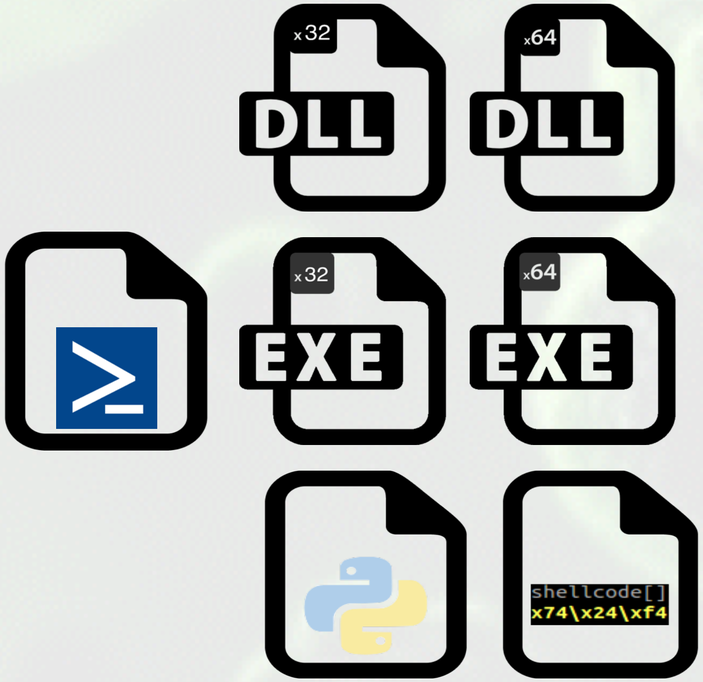
E.B.O.W.L.A.

Ethnic **B**i**O** **W**eapon **L**imited **A**ccess

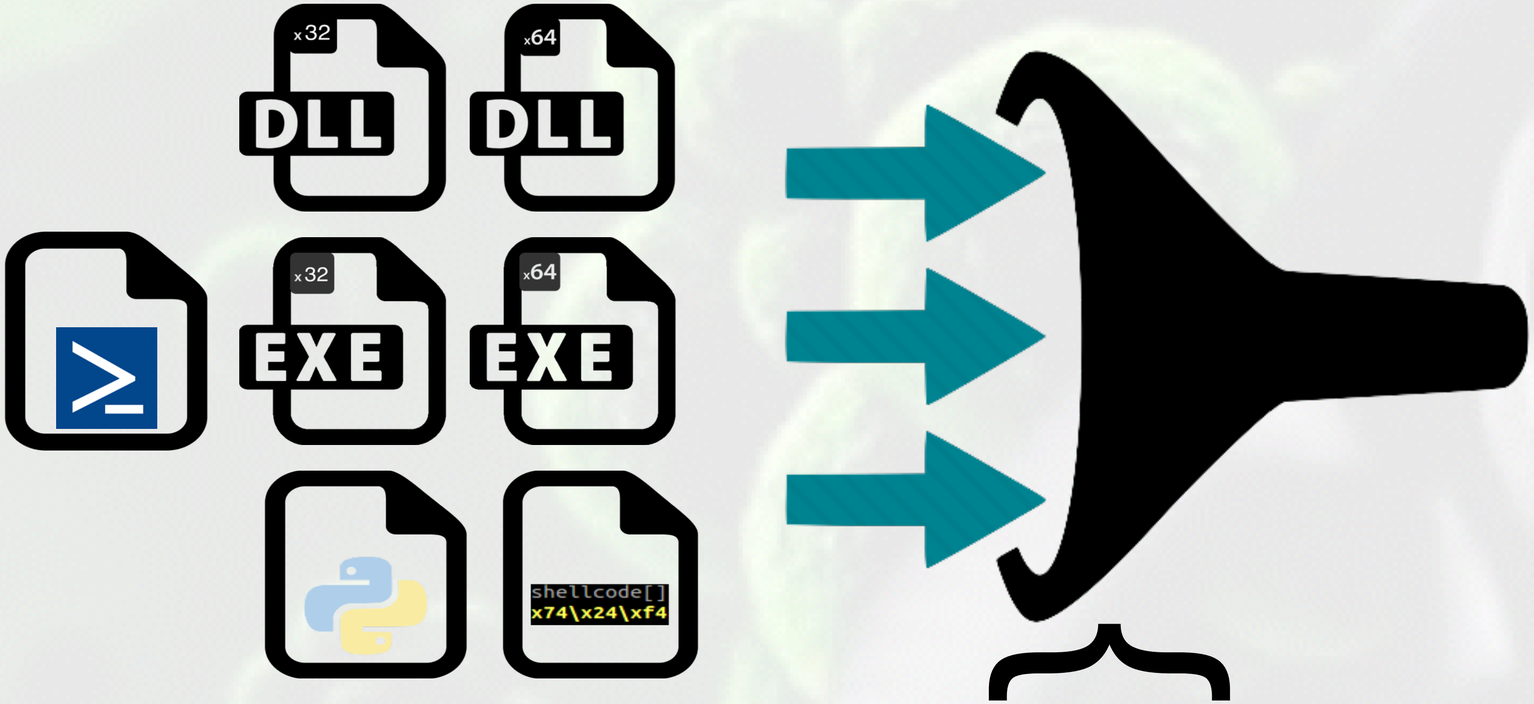
High Level Overview


NO
DISASSEMBLE!
quickmeme.com

E.B.O.W.L.A.

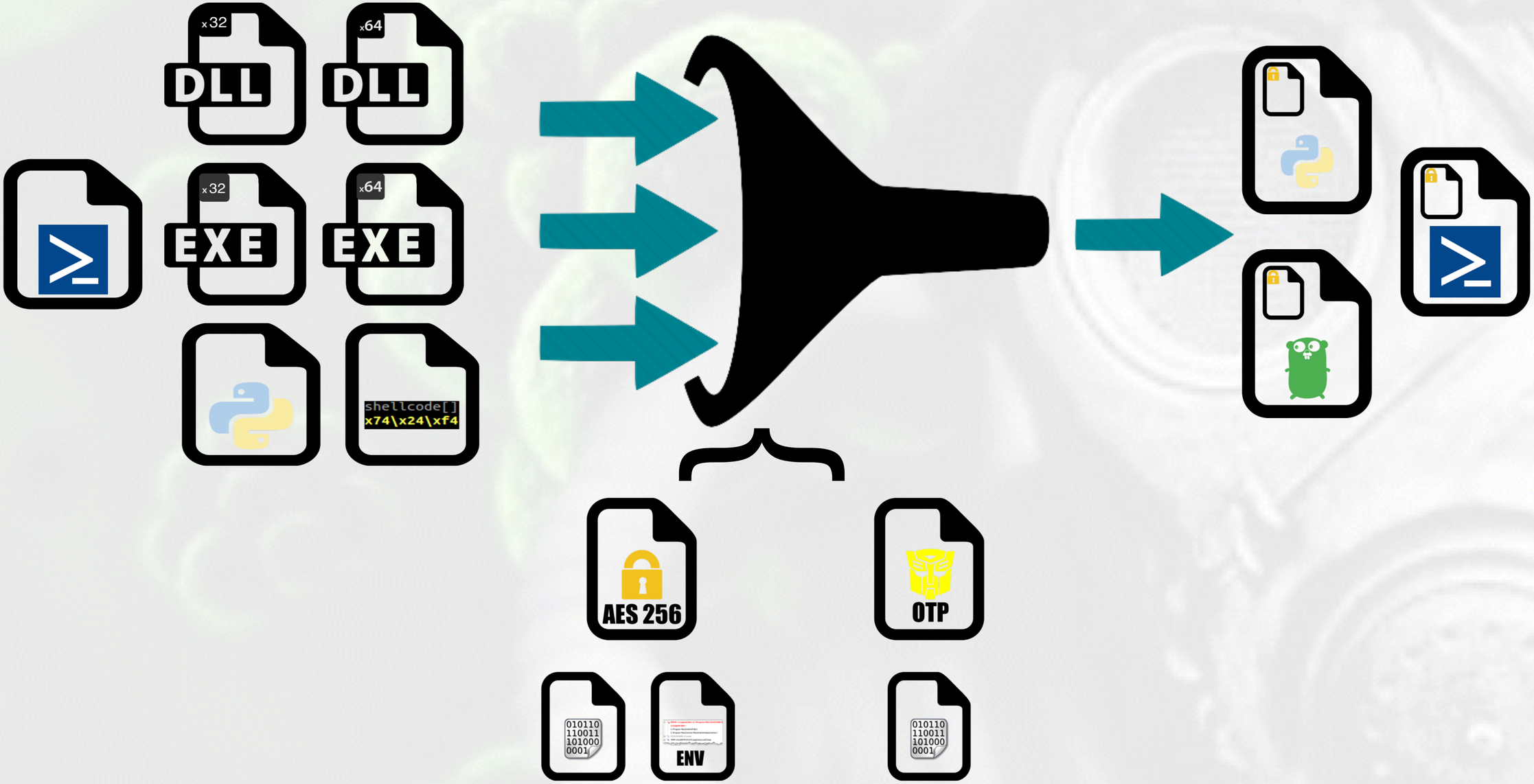


E.B.O.W.L.A.

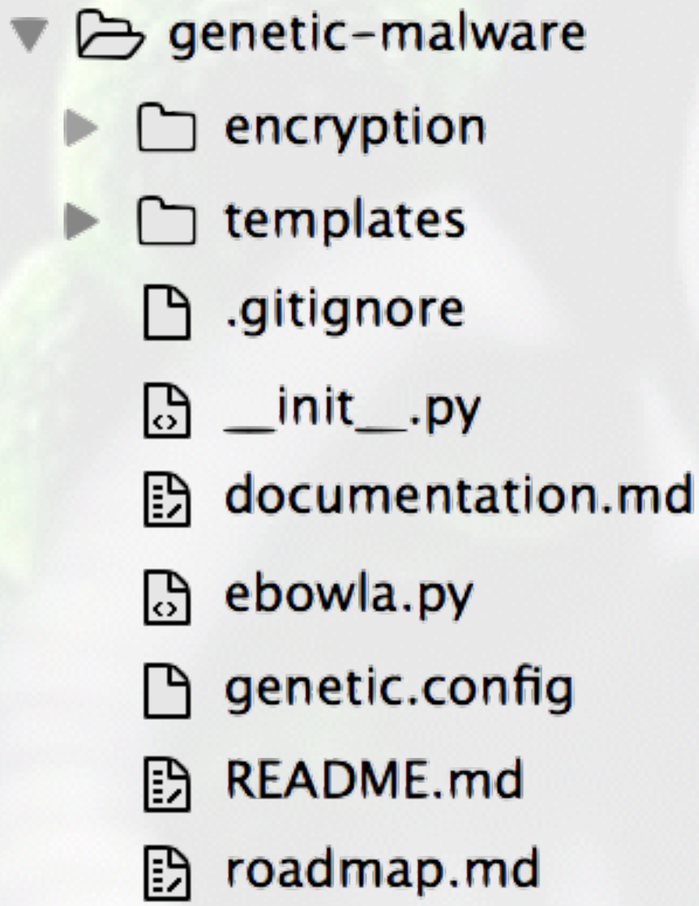

x32
DLL
x64
DLL
x32
EXE
x64
EXE
shellcode[]
x74\x24\xf4


AES 256
OTP
010110
110011
101000
0001
ENV
010110
110011
101000
0001

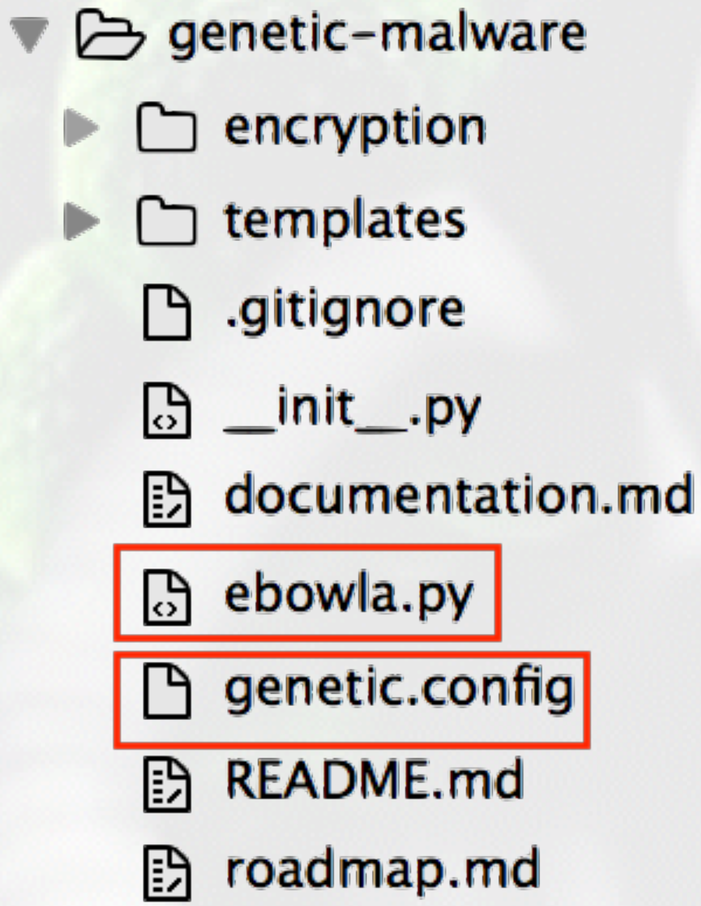
E.B.O.W.L.A.


x32
DLL
x64
DLL
x32
EXE
x64
EXE
shellcode[]
x74\x24\xf4
AES 256
OTP
010110
110011
101000
0001
ENV
010110
110011
101000
0001

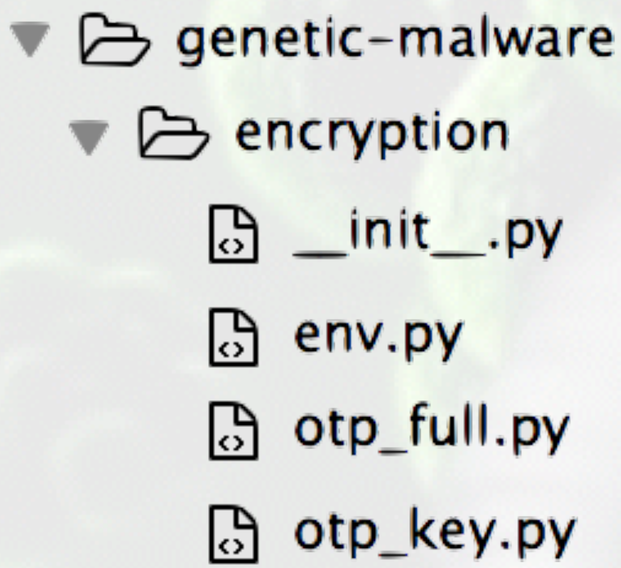
Framework


genetic-malware
encryption
templates
.gitignore
__init__.py
documentation.md
ebowla.py
genetic.config
README.md
roadmap.md

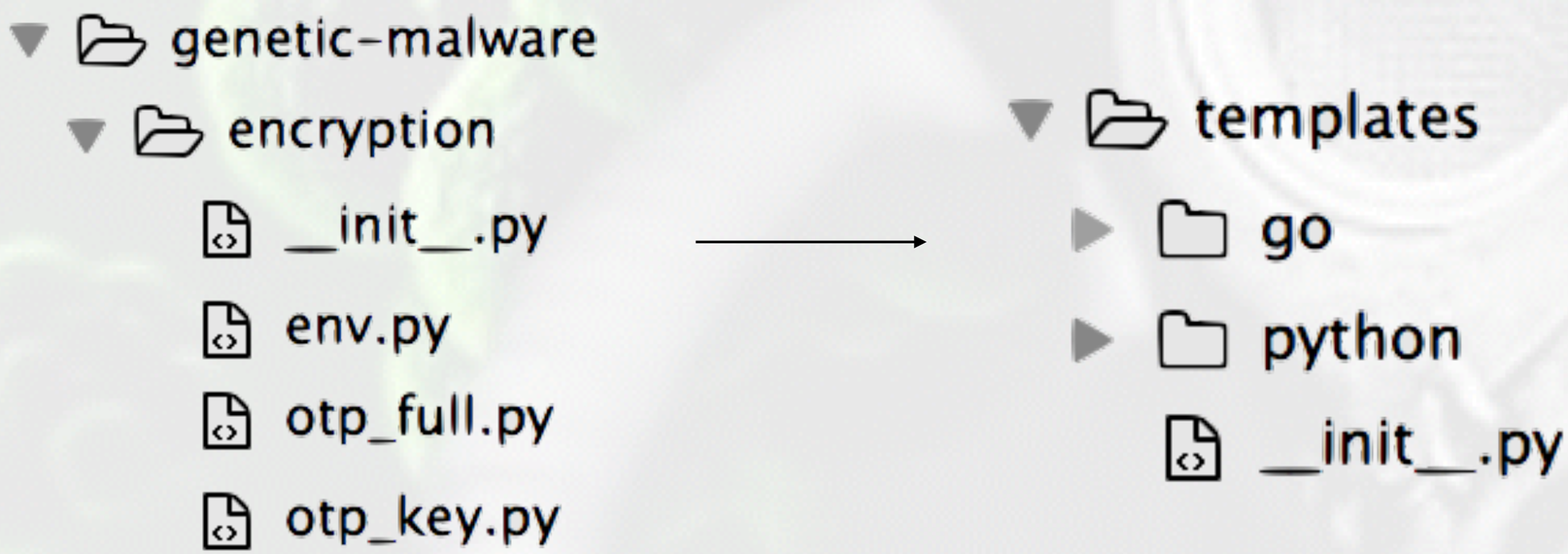
Framework


genetic-malware
encryption
templates
.gitignore
__init__.py
documentation.md
ebowla.py
genetic.config
README.md
roadmap.md

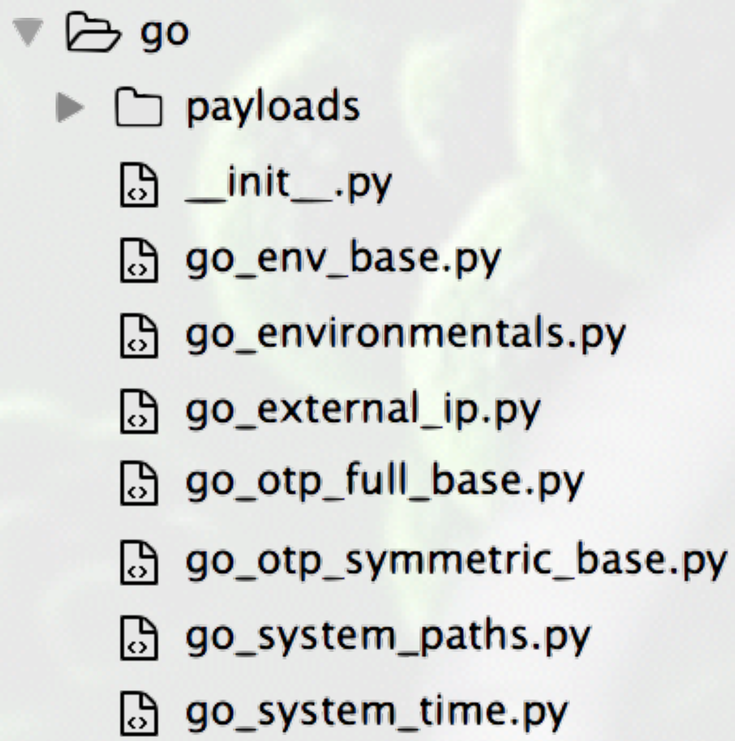
Framework


genetic-malware
encryption
__init__.py
env.py
otp_full.py
otp_key.py

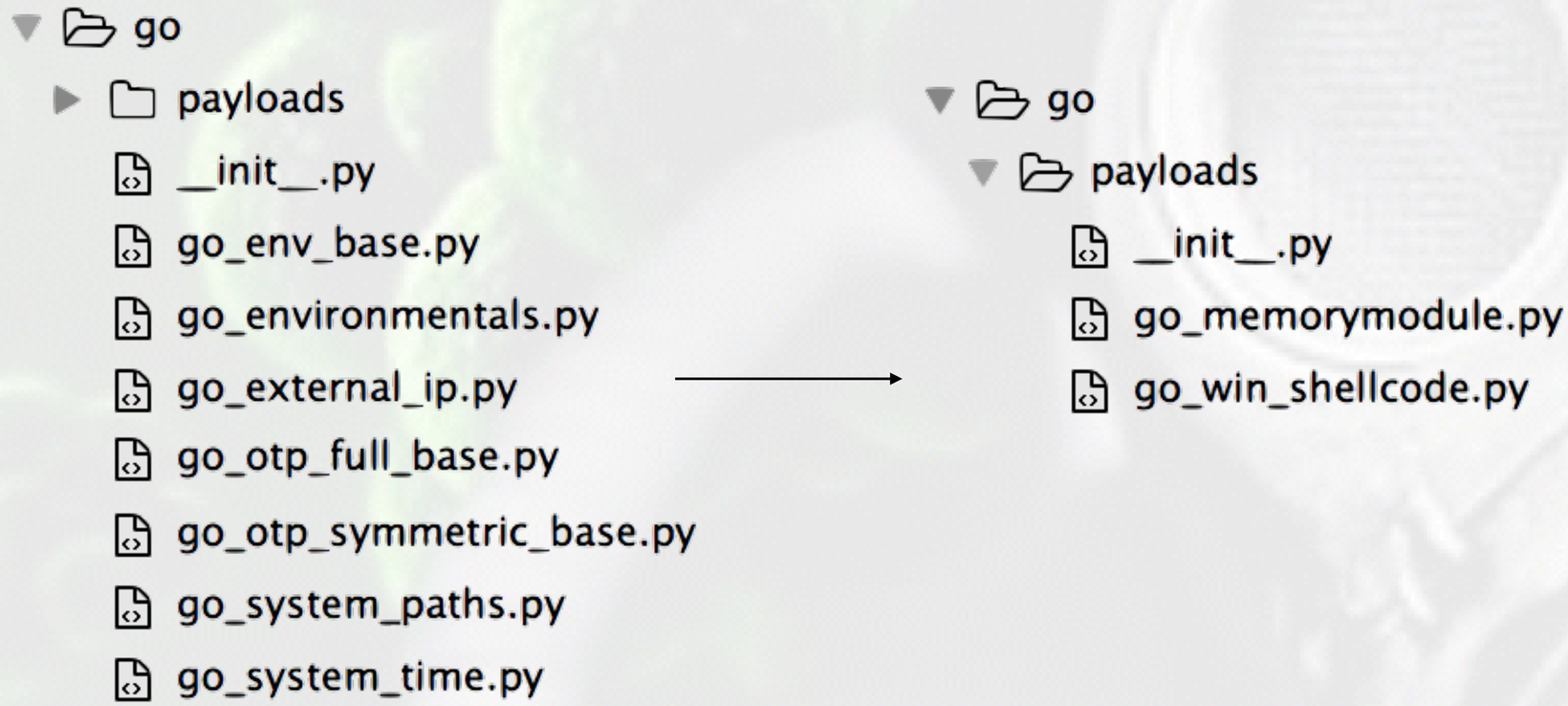
Framework


genetic-malware
encryption
__init__.py
env.py
otp_full.py
otp_key.py
templates
go
python
__init__.py

Framework


go
payloads
__init__.py
go_env_base.py
go_environmentals.py
go_external_ip.py
go_otp_full_base.py
go_otp_symmetric_base.py
go_system_paths.py
go_system_time.py

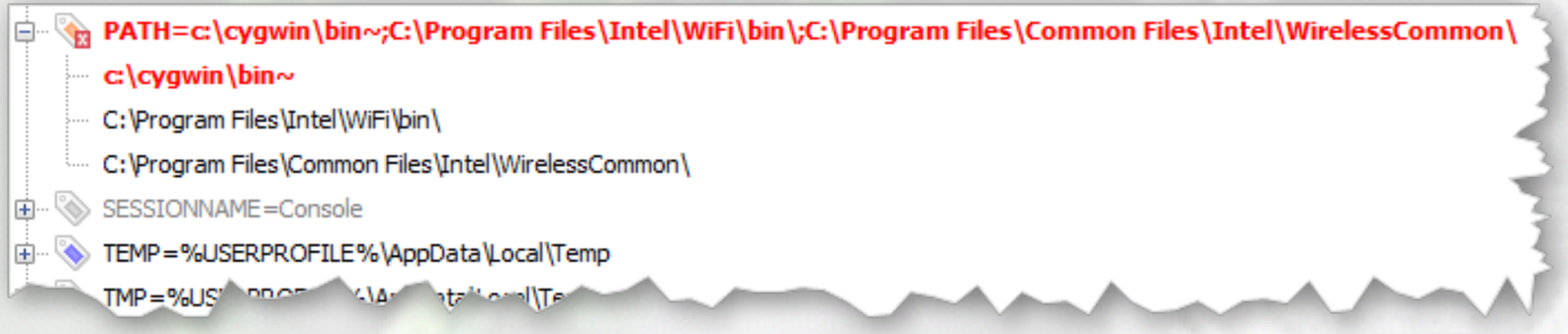
Framework


go
payloads
__init__.py
go_env_base.py
go_environmentals.py
go_external_ip.py
go_otp_full_base.py
go_otp_symmetric_base.py
go_system_paths.py
go_system_time.py
go
payloads
__init__.py
go_memorymodule.py
go_win_shellcode.py

Protection Mechanisms

Protection Mechanisms

Key Derivation:
Environmental Factors



Supported Environmentals

- Environment Variables (e.g. %TEMP%, %USERNAME%, %TEMP%, etc)
- File System Path (e.g. C:\windows\temp)
- External IP Range (e.g. 100.10.0.0, 100.0.0.0)
- Time Trigger (e.g. 20160401)

Key Derivation:
Environmental Factors

Encryption:

```
payload_hash = sha512(payload[:-offset_bytes])

key = ((sha512(token1+token2+…)) * Iterations)[:32]

enc_blob = base64(zlib*(iv+AES.CFB(key,iv,payload)))
```

Key Derivation: Environmental Factors

Encryption:

payload_hash = sha512(payload[:-offset_bytes])

key = ((sha512(token1+token2+…)) * Iterations)[:32]

enc_blob = base64(zlib*(iv+AES.CFB(key,iv,payload)))

Decryption:

1) Retrieve environment variables

2) Traverse File System from StartingPoint

3) Combine into all possible combinations and decrypt

** trial_key = sha512(token1 + token2 + …)* Iterations)[:32]

** if(sha512(decryptpayload(iv,enc_blob,trial_key[:-offset_bytes]) == payload_hash; continue

Key Derivation:
Unique File

Key Derivation:
Unique File

Encryption:

```
payload_hash = sha512(payload[:-offset_bytes])

location = rand_location(uniq_key_file)

key = ((sha512(read.location) * Iterations)[:32]

enc_blob = base64(zlib*(location + lc.length + iv +
AES.CFB(key,iv,payload)))
```

Key Derivation:
Unique File

Encryption:

```
payload_hash = sha512(payload[:-offset_bytes])

location = rand_location(uniq_key_file)

key = ((sha512(read.location) * Iterations)[:32]

enc_blob = base64(zlib*(location + lc.length + iv +
AES.CFB(key,iv,payload)))
```

Decryption:

1) Traverse File System from StartingPoint

2) Create a key from every file encountered & Attempt Decryption

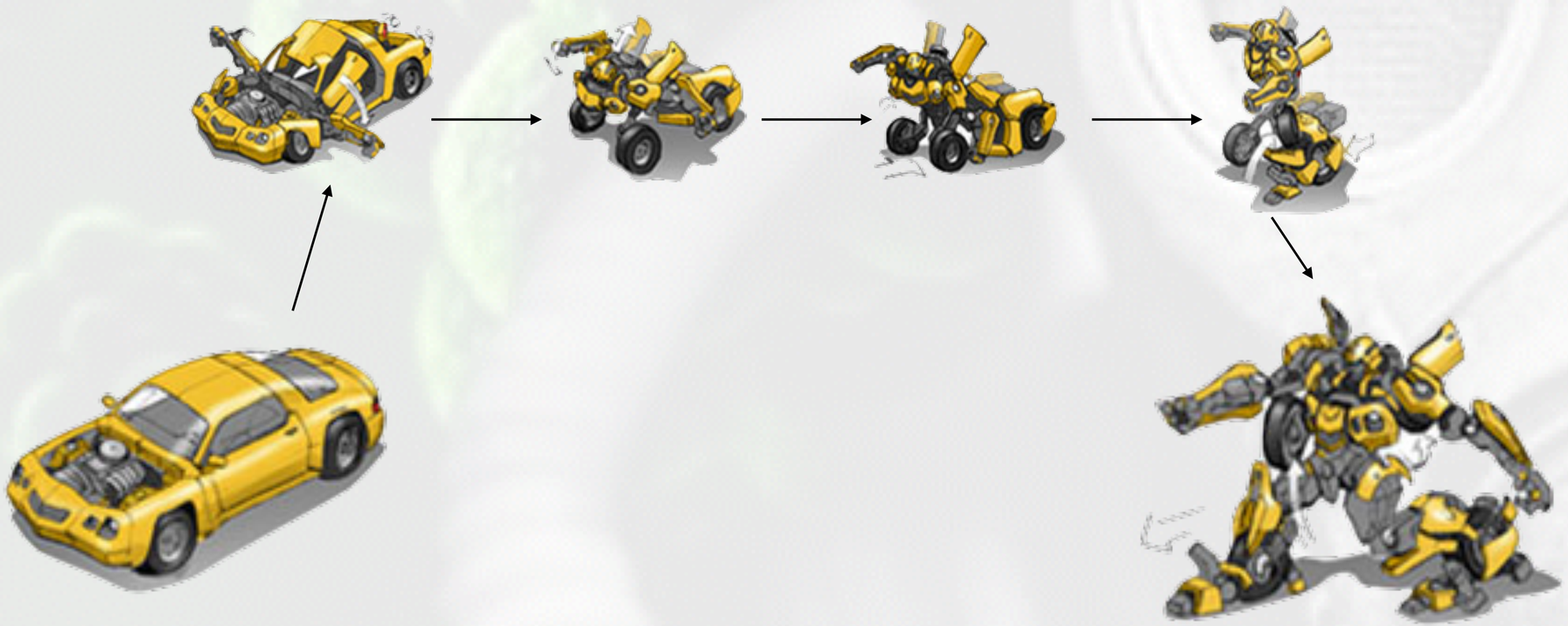
```
** trial_key = sha512(readFile.location)* Iterations)[:32]

** if(sha512(decryptpayload(iv,enc_blob[22:],trial_key)[:-
offset_bytes]) == payload_hash; continue
```

Protection Mechanisms

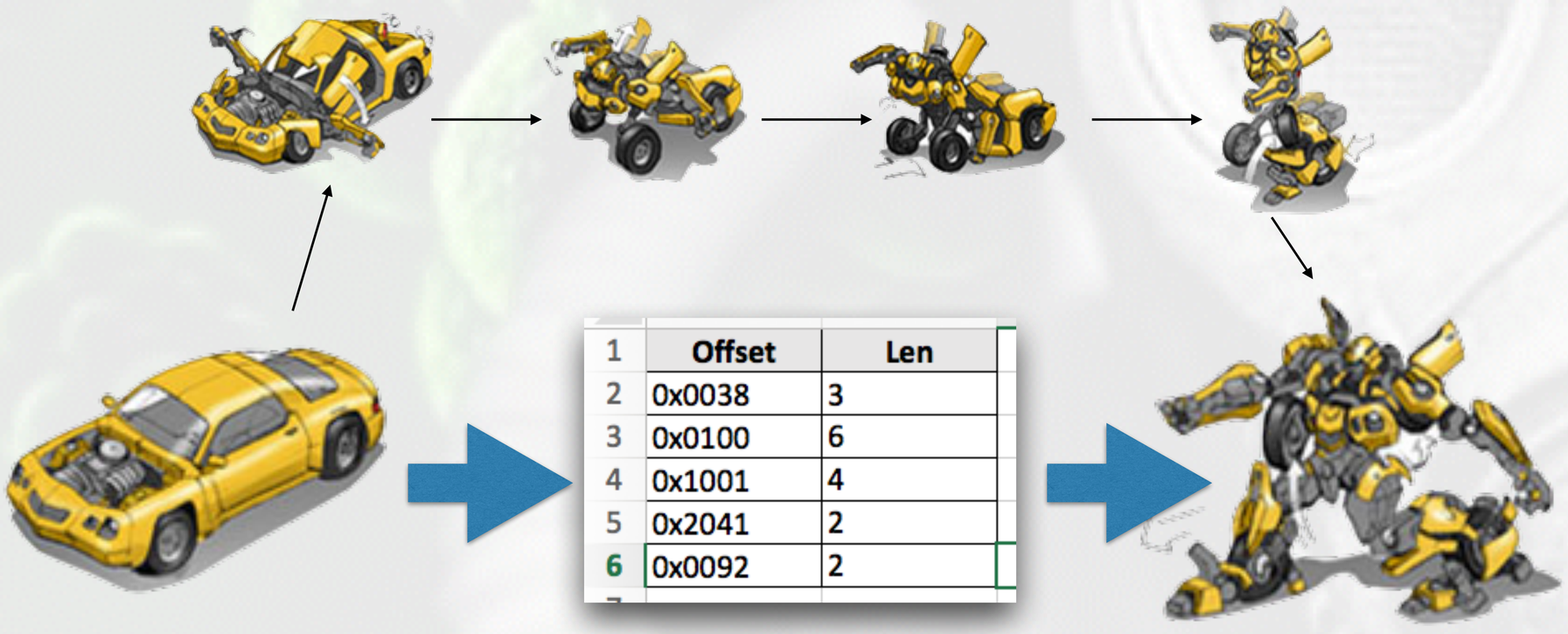
Protection Mechanisms

Protection Mechanisms

Key Derivation:
One Time Pad (OTP)



Key Derivation:
One Time Pad (OTP)



	Offset	Len
2	0x0038	3
3	0x0100	6
4	0x1001	4
5	0x2041	2
6	0x0092	2

Key Derivation:
One Time Pad (OTP)

Pad Creation:

1) payload_hash = sha512(payload[:-offset_bytes])

2) short_len = len(payload)*10%

3) payload_hash_short = sha512(payload)[:short_len]

4) lookup_table(uniqueBinary) = base64(zlib*([[offset_loc][len],[offset_loc][len], …]))

Key Derivation:
One Time Pad (OTP)

Offset	0	1	2	3	4	5	6	7	8
00000...	80	C0	00	20	DD	BE	00	00	CD
00000...	F5	AD	00	00	F9	AD	00	00	FD
00000...	05	AE	00	00	09	AE	00	00	0D
00000...	11	AE	00	00	15	AE	00	00	E5
00000...	19	AE	00	00	1D	AE	00	00	21
00000...	29	AE	00	00	2D	AE	00	00	31
00000...	39	AE	00	00	3D	AE	00	00	41

Attacker Payload

DD	BE	00	00	CD	AD	00	00	F1
F9	AD	00	00	FD	AD	00	00	01
09	AE	00	00	0D	AE	00	00	D9
15	AE	00	00	E5	AD	00	00	55
1D	AE	00	00	21	AE	00	00	25
2D	AE	00	00	31	AE	00	00	35
3D	AE	00	00	41	AE	00	00	45
3D	AE	00	00	41	AE	00	00	45

Target UniqueBinary

Key Derivation:
One Time Pad (OTP)

Offset	0	1	2	3	4	5	6	7	8
00000...	80	C0	00	20	DD	BE	00	00	CD
00000...	F5	AD	00	00	F9	AD	00	00	FD
00000...	05	AE	00	00	09	AE	00	00	0D
00000...	11	AE	00	00	15	AE	00	00	E5
00000...	19	AE	00	00	1D	AE	00	00	21
00000...	29	AE	00	00	2D	AE	00	00	31
00000...	39	AE	00	00	3D	AE	00	00	41

Attacker Payload

DD	BE	00	00	CD	AD	00	00	F1
F9	AD	00	00	FD	AD	00	00	01
09	AE	00	00	0D	AE	00	00	D9
15	AE	00	00	E5	AD	00	00	55
1D	AE	00	00	21	AE	00	00	25
2D	AE	00	00	31	AE	00	00	35
3D	AE	00	00	41	AE	00	00	45
3D	AE	00	00	41	AE	00	00	45

Target UniqueBinary

Lookup Table

1	Offset	Len
2	0x0038	3
3		
4		

Key Derivation:
One Time Pad (OTP)

Decryption:

1) Traverse File System from StartingPoint

2) Open Each file and build 10%

3) Validate 10% hash Matches then build entire payload

** if(sha512(rebuild_payload(lookup_table,current_file)[:-offset_bytes] == payload_hash; exec()

Execution Loader Protection

Issue

Execution mechanism easily discovered in scripting languages

Fix

Protect the loader in Powershell and Python

Outputs
(aka Cyber Pathogens)

Warning: Dormant
Cyber Pathogen

Outputs

GO

PS

Python

Outputs

GO PS Python

Input/Out Compatibility

Payload	Python		GO		PowerShell	
	x64	x32	x64	x32	x64	x32
Reflective DLL			In Memory	In Memory	In Memory	In Memory
DLL			In Memory	In Memory	In Memory	In Memory
EXE	On Disk	On Disk	In Memory	In Memory	In Memory	In Memory
ShellCode	In Memory	In Memory	In Memory	In Memory	In Memory	In Memory
Python Code	In Memory	In Memory				
PowerShell Code					In Memory	
FileDrop	Supported		In Progress		Supported	

Input/Out Compatibility

Payload	Python		GO		PowerShell	
	x64	x32	x64	x32	x64	x32
Reflective DLL			In Memory	In Memory	In Memory	In Memory
DLL			In Memory	In Memory	In Memory	In Memory
EXE	On Disk	On Disk	In Memory	In Memory	In Memory	In Memory
ShellCode	In Memory	In Memory	In Memory	In Memory	In Memory	In Memory
Python Code	In Memory	In Memory				
PowerShell Code					In Memory	
FileDrop	Supported		In Progress		Supported	

New PowerShell

- Uses Invoke-ReflectivePEInjection for PE/DLL injection by Joe Bialek: @JosephBialek
- Uses Invoke-Shellcode by Matt Graeber (@mattifestation)

Usage

```
$ ./ebowla.py payload config

$ #Then compile output
```

The config file

Three Sections

- Overall
- OTP Settings
- Symmetric Settings

Overall Section

```
Encryption_Type
 OPTIONS: OTP ENV

output_type
 OPTIONS: Python, GO, PowerShell

payload_type
 OPTIONS for GO: EXE, DLL_x86, DLL_x64, SHELLCODE
 OPTIONS for PYTHON: EXE, SHELLCODE, CODE, FILE_DROP
 OPTIONS for PS: CODE, FILE_DROP, DLL_x86, DLL_x64, EXE, SHELLCODE

key_iterations
 OPTIONS: Any number? Be reasonable.
```

Symmetric Key Settings

This has four sections:

- ENV_VARS
- PATH
- IP_RANGES
- SYSTEM_TIME

Symmetric Key Settings

```
ENV_VARS
 Can be anything, can add whatever you want
 if value is ‘’, it is not used. The value is used as a key.

 examples:
 username = ‘Administrator’ #  Used
 homepath = ‘’  # Not used

PATH

 path
  This is used as a key.
  OPTIONS: A full static path.

 start_loc
  Location to start looking for path match
  OPTIONS: Static location or Env variable (%PROGRAMFILES%)
```

Symmetric Key Settings

IP_RANGES

external_ip_mask
Simple IP MASK, limited to /24 /16 /8
Example: 11.12.13.14, 11.12.13.0, 11.12.0.0, 11.0.0.0

SYSTEM_TIME

Time_Range
Limited to Year, Month, or DAY
Format: YYYYMMDD
Example: 20160401, 20160400, or 20160000

DEMO TIME

DEMO TIME

The Scenario

- An American in Paris is low on Rubles
- Wants Starcraft really bad
- Answer: BitTorrent a cracked game!
- Unfortunately the cracked starcraft games are patched with a backdoor targeting the most current version of BitTorrent

DEMO 1: OTP

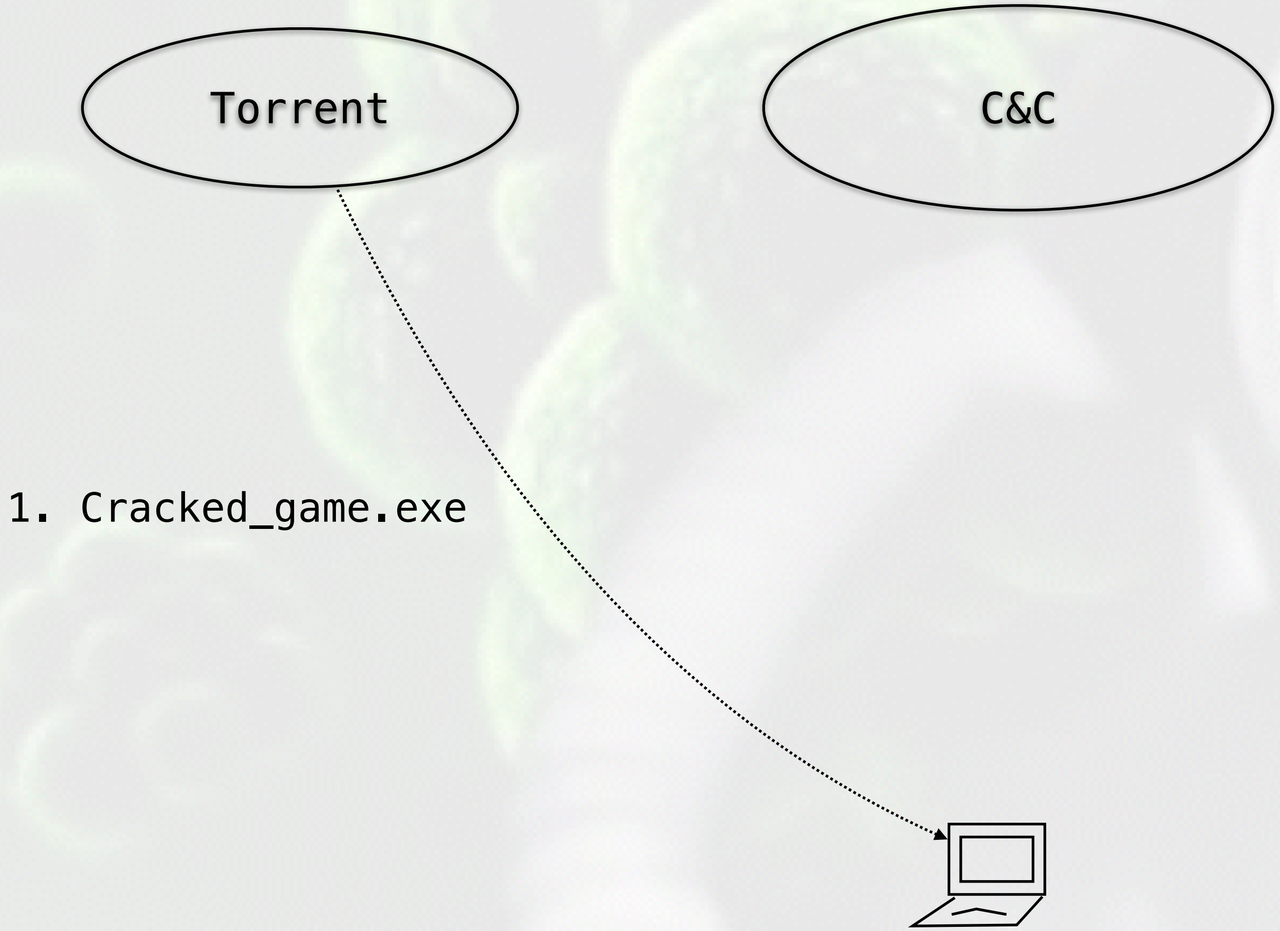
- Using BitTorrent.exe as the PAD
  - Version 7.9.5, Build 41866, 32bit
  - Meterpreter reverse https is the payload via a first stage DLL
  - Searching for the PAD starts in %APPDATA%
- Code delivered through a backdoored/cracked game
  - Download and Execute payload

DEMO 1: OTP

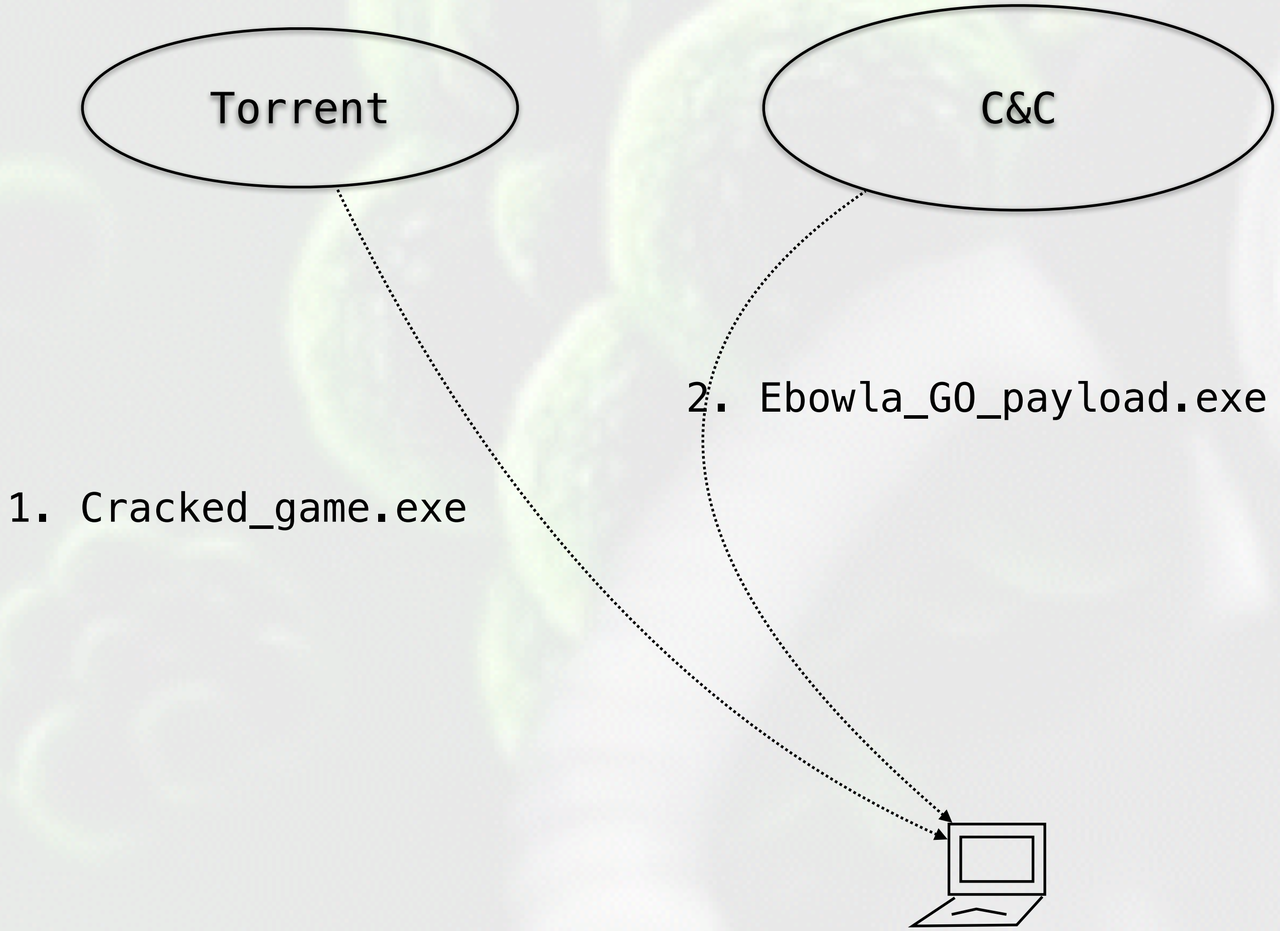
Torrent

C&C

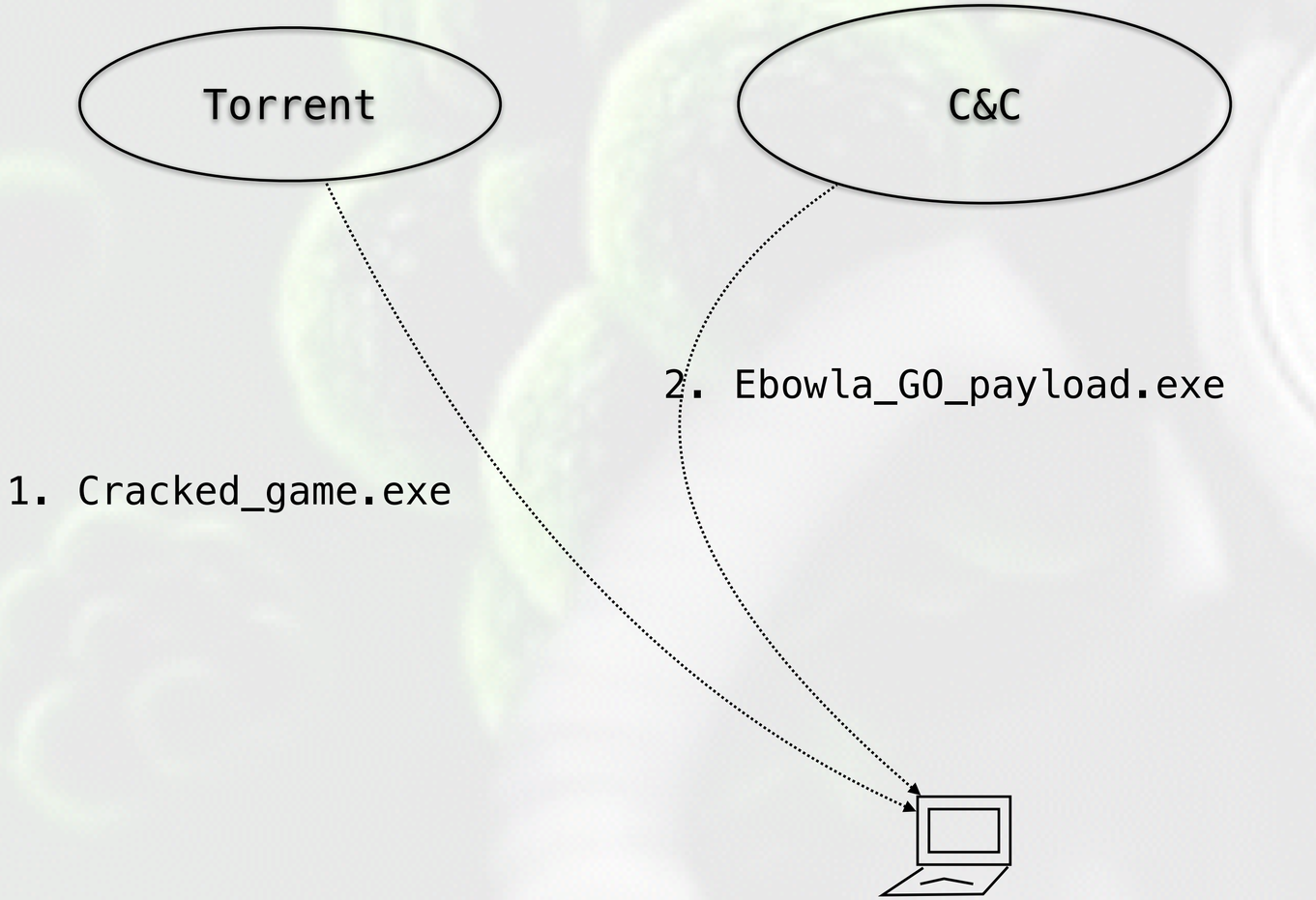
DEMO 1: OTP



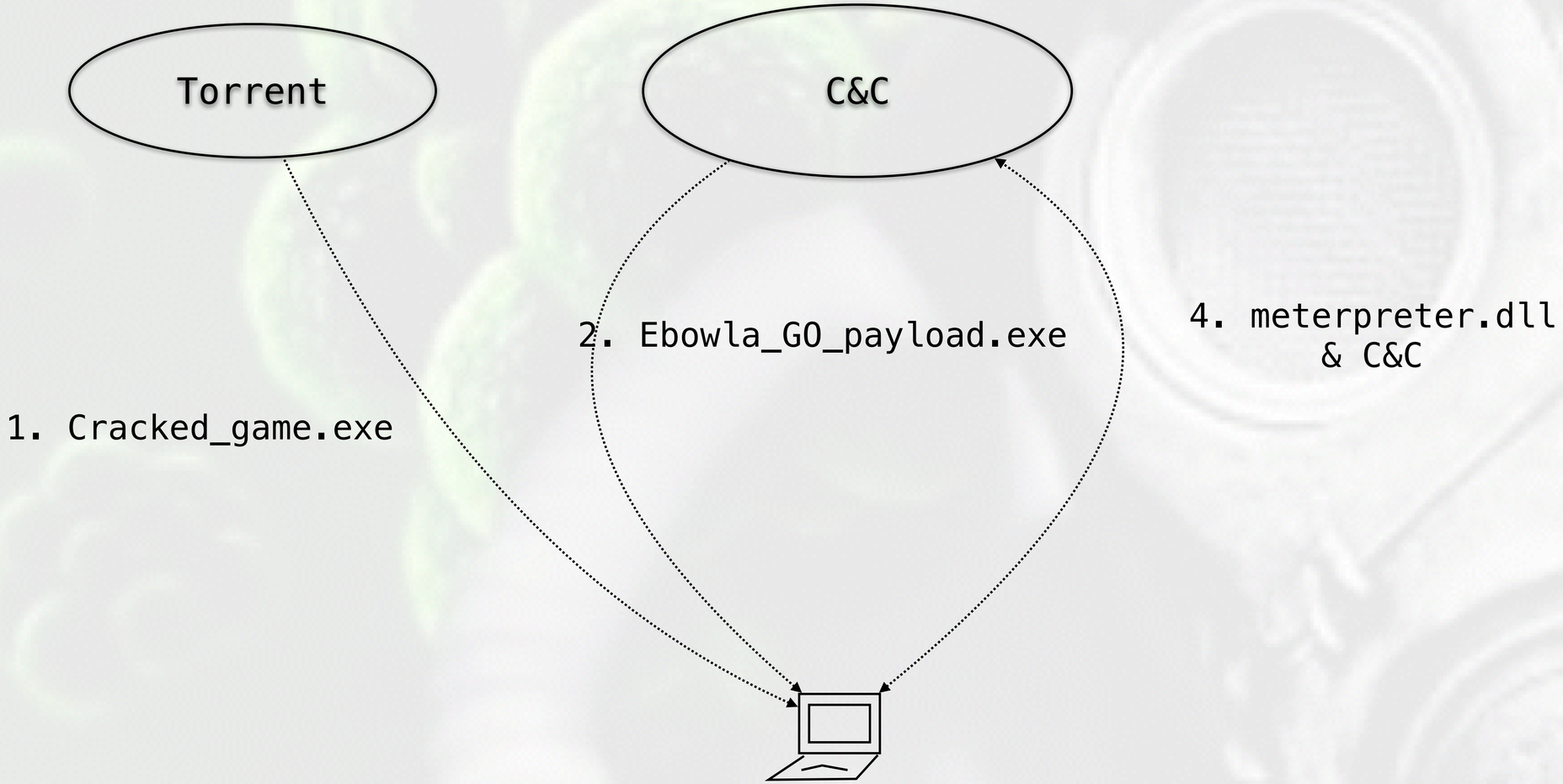
DEMO 1: OTP



DEMO 1: OTP


Torrent
C&C
2. Ebowla_GO_payload.exe
1. Cracked_game.exe
3. In memory Execution of a
reverse https stage one payload
as a DLL

DEMO 1: OTP


Torrent
C&C
1. Cracked_game.exe
2. Ebowla_GO_payload.exe
3. In memory Execution of a reverse https stage one payload as a DLL
4. meterpreter.dll & C&C

DEMO 2: Key from File

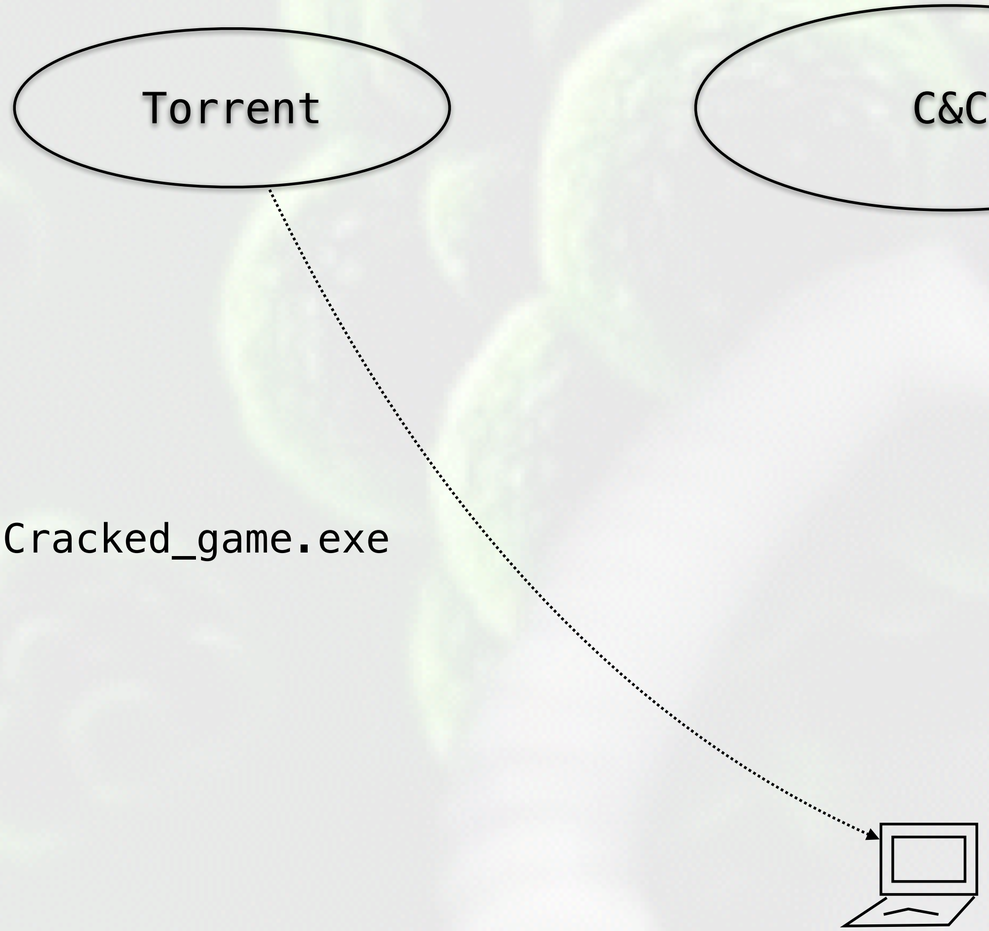
- Using a location in BitTorrent.exe as the AES key source
  - Version 7.9.5, Build 41866, 32bit
  - Pupy EXE reverse https
  - Searching starts in %APPDATA%
- Code delivered through a backdoored/cracked game
  - Download and Execute payload

DEMO 2: Key from File

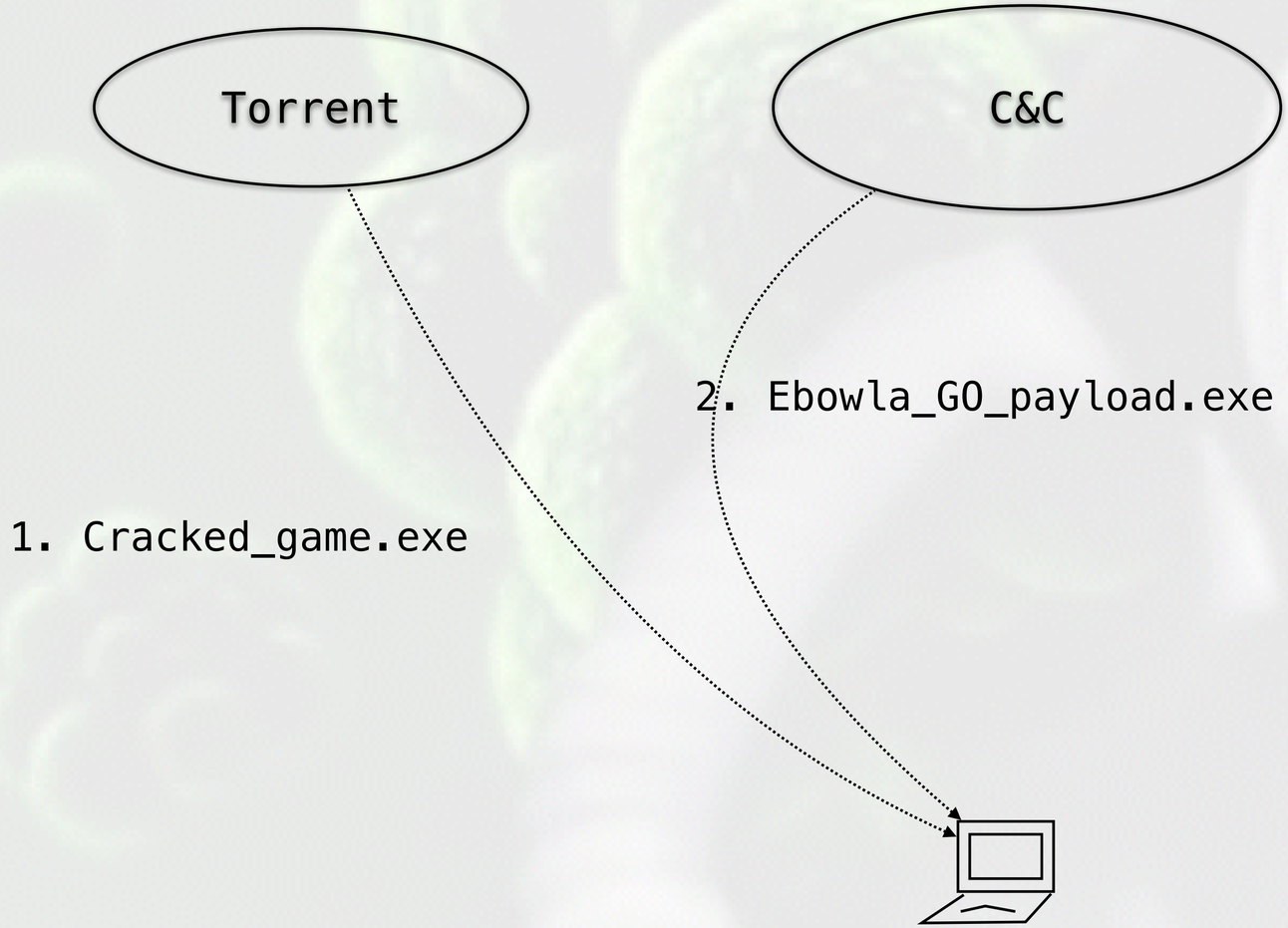
Torrent

C&C

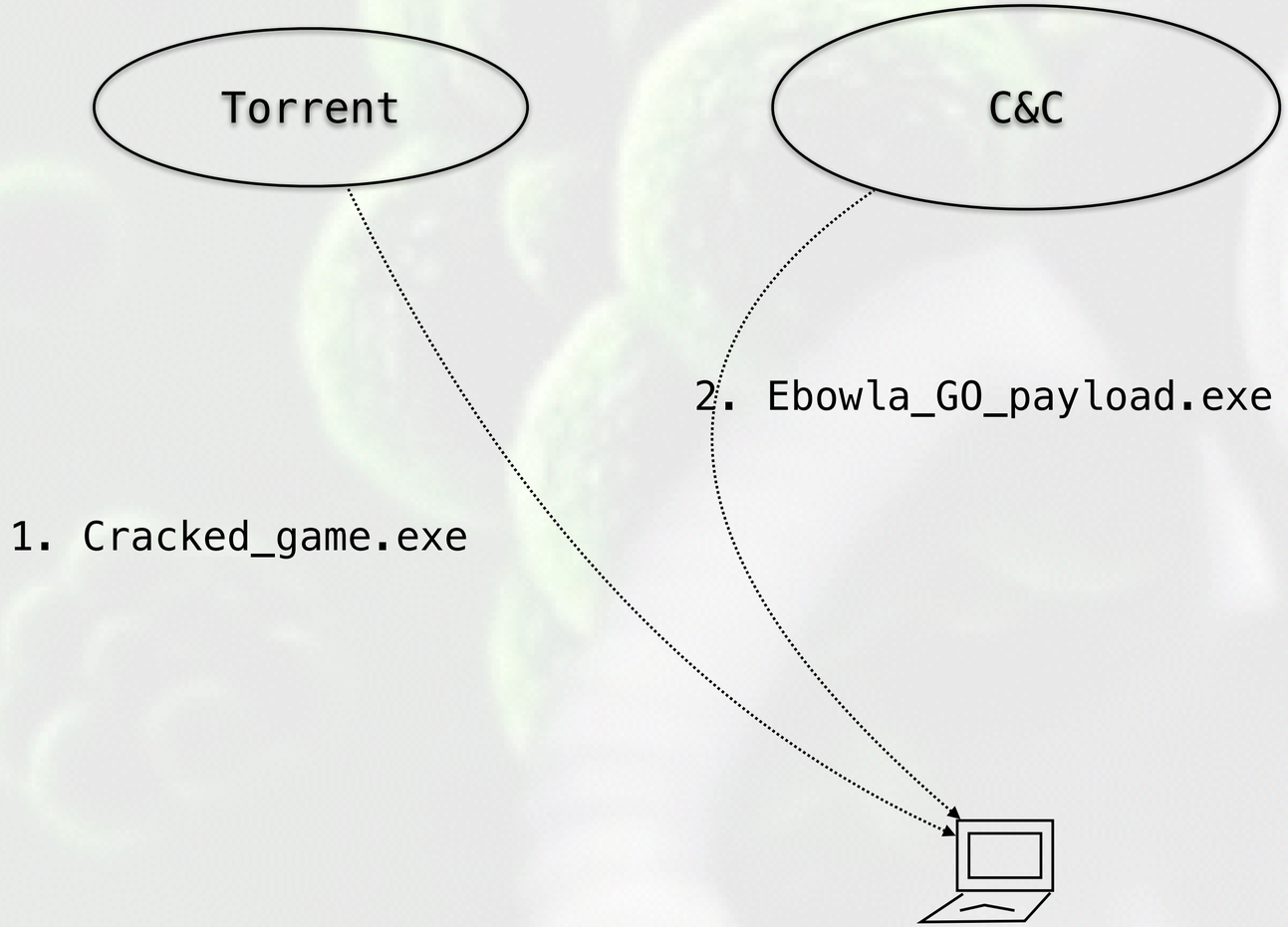
DEMO 2: Key from File



DEMO 2: Key from File

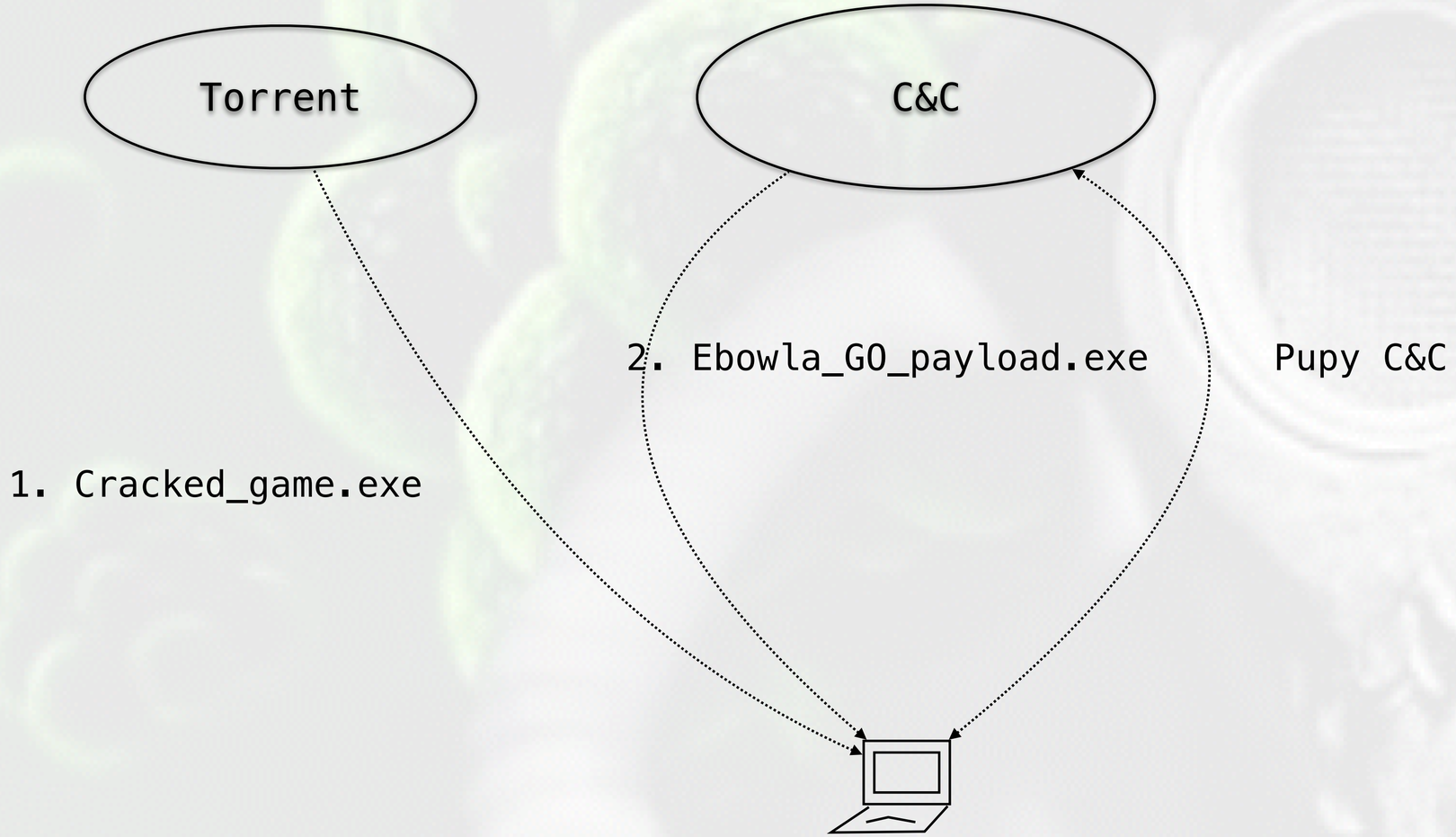


DEMO 2: Key from File



3. In memory execution the Pupy EXE

DEMO 2: Key from File



DEMO 3:Layered Payload

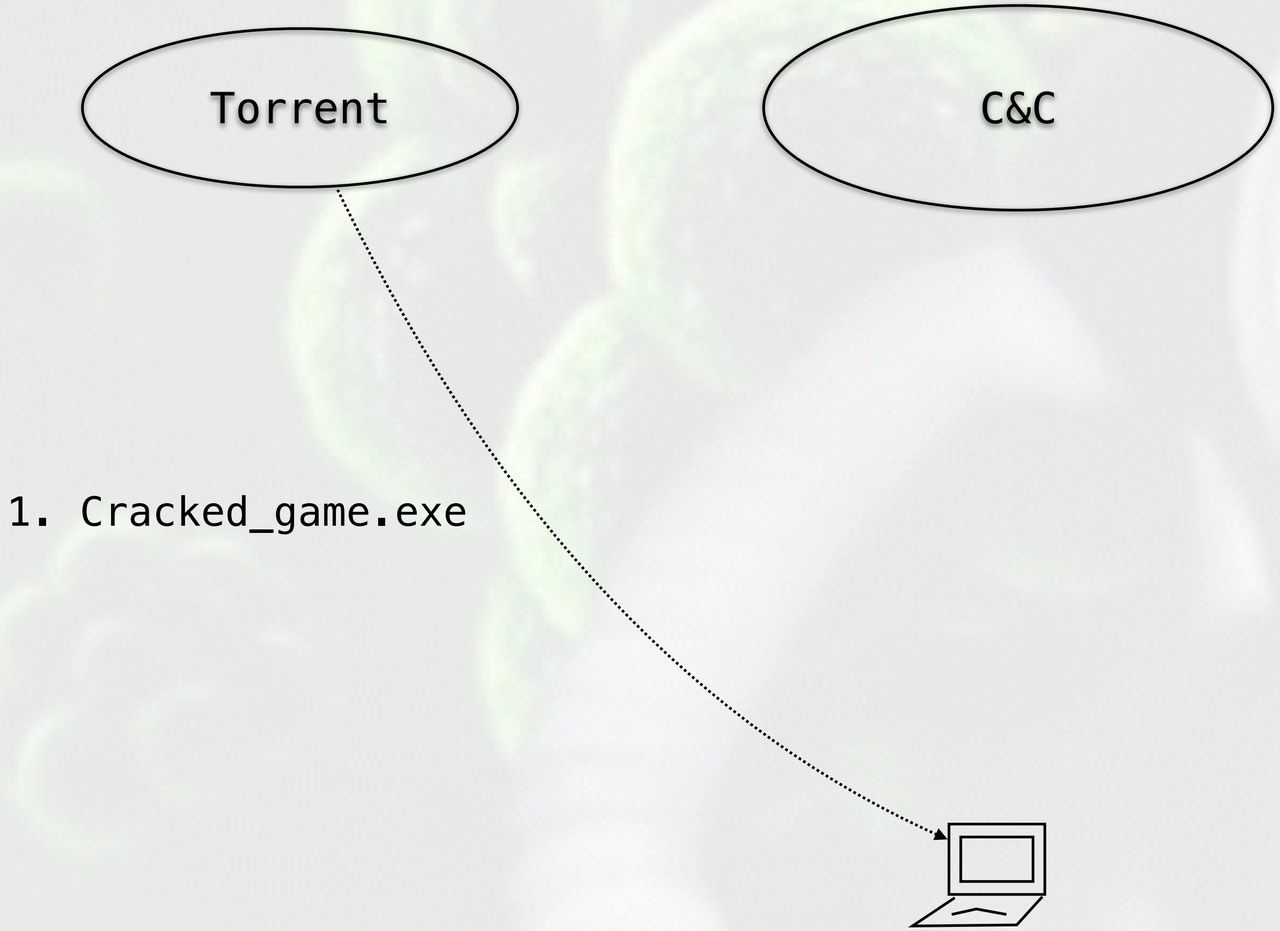
- Using Environmental Factors
- Stage 2:
  - Env Vars: Computer Name, number of processors as keys
  - GO EXE launching Pupy x64 DLL
- Stage 1:
  - Using Date Range and IP Mask as keys
  - Python EXE, writes stage 1 to disk and Executes
- Code delivered through a backdoored/cracked game
  - Download and Execute payload

DEMO 3:Layered Payload

Torrent

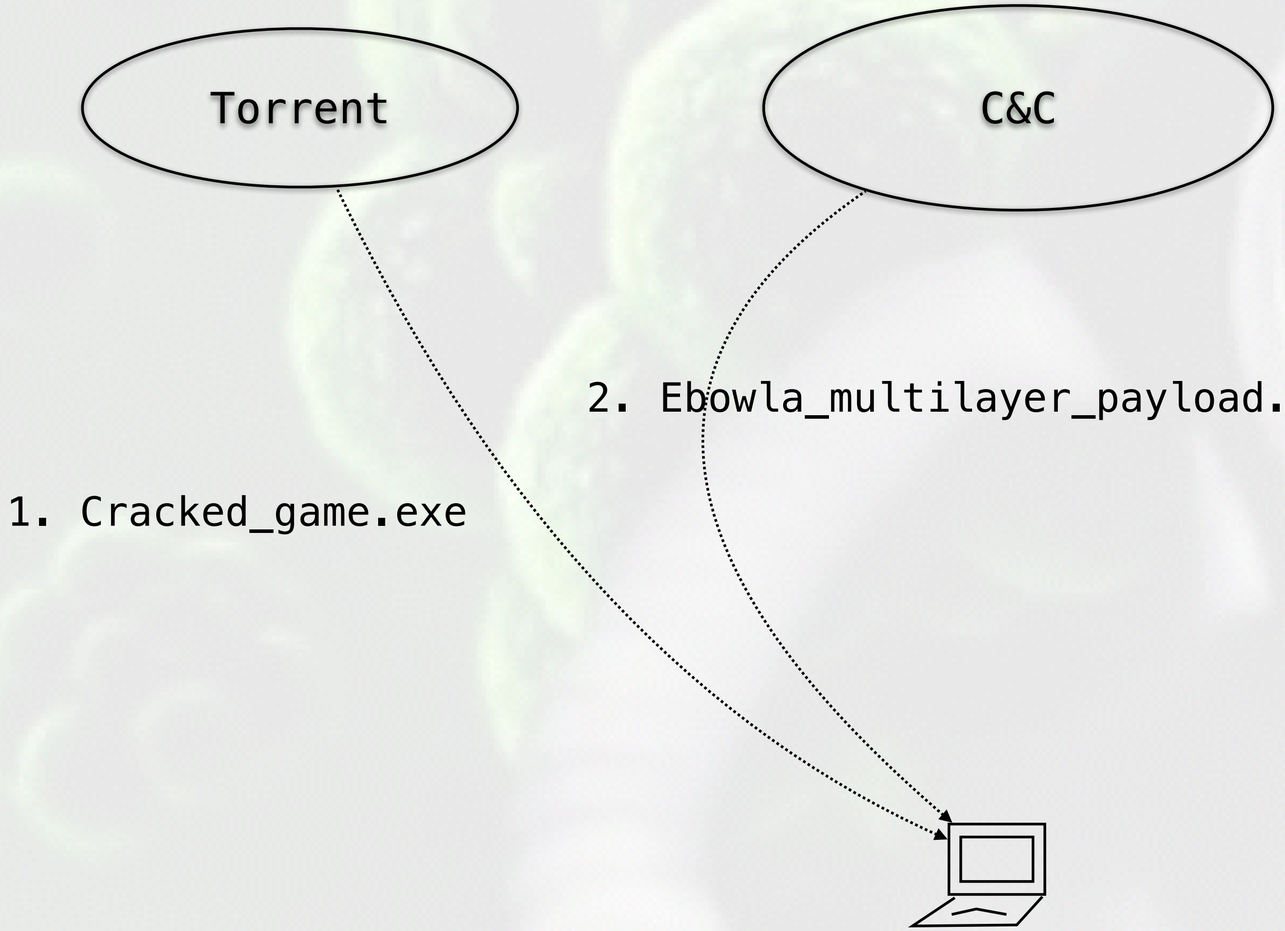
C&C

DEMO 3:Layered Payload

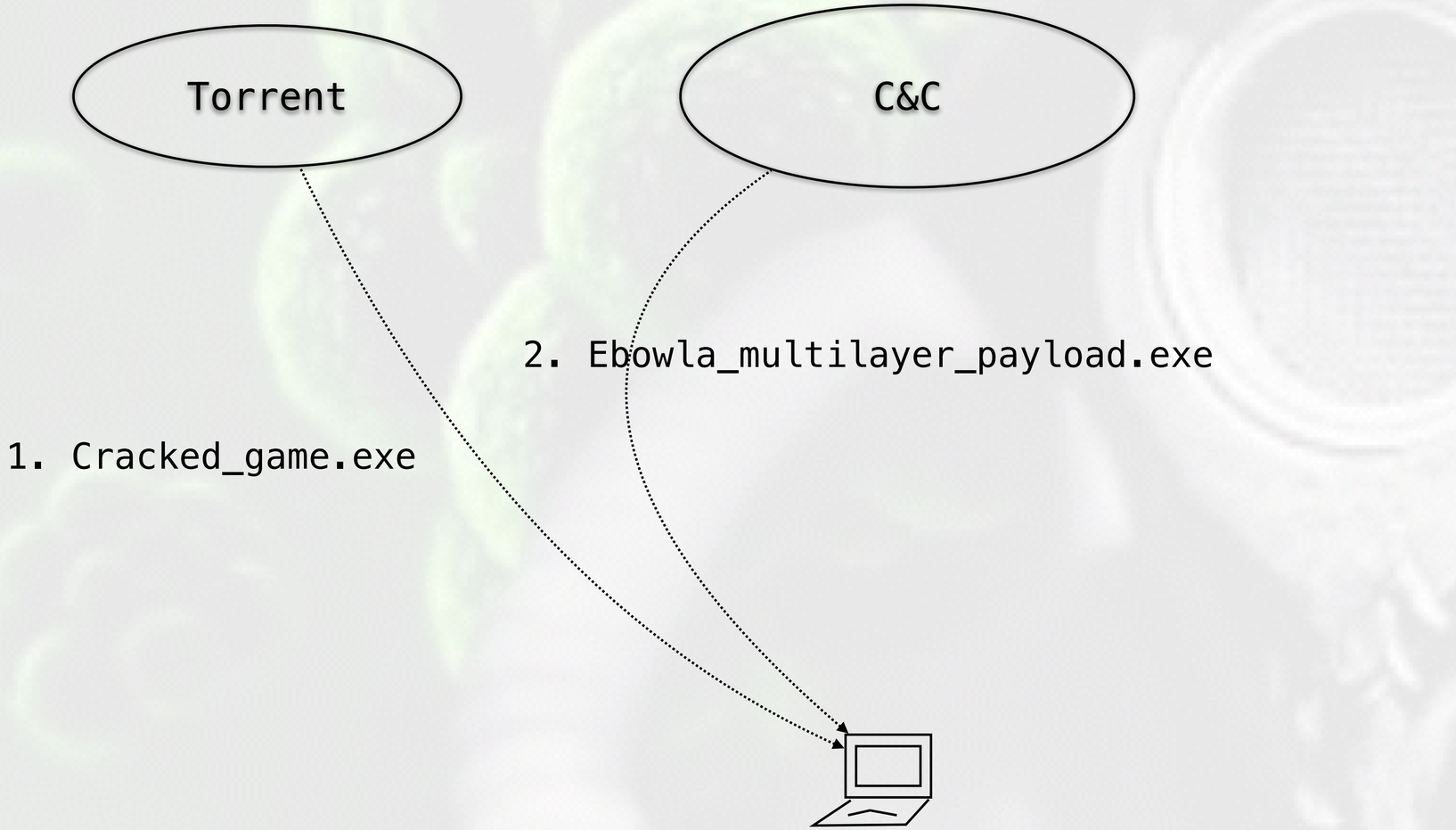


1. Cracked_game.exe

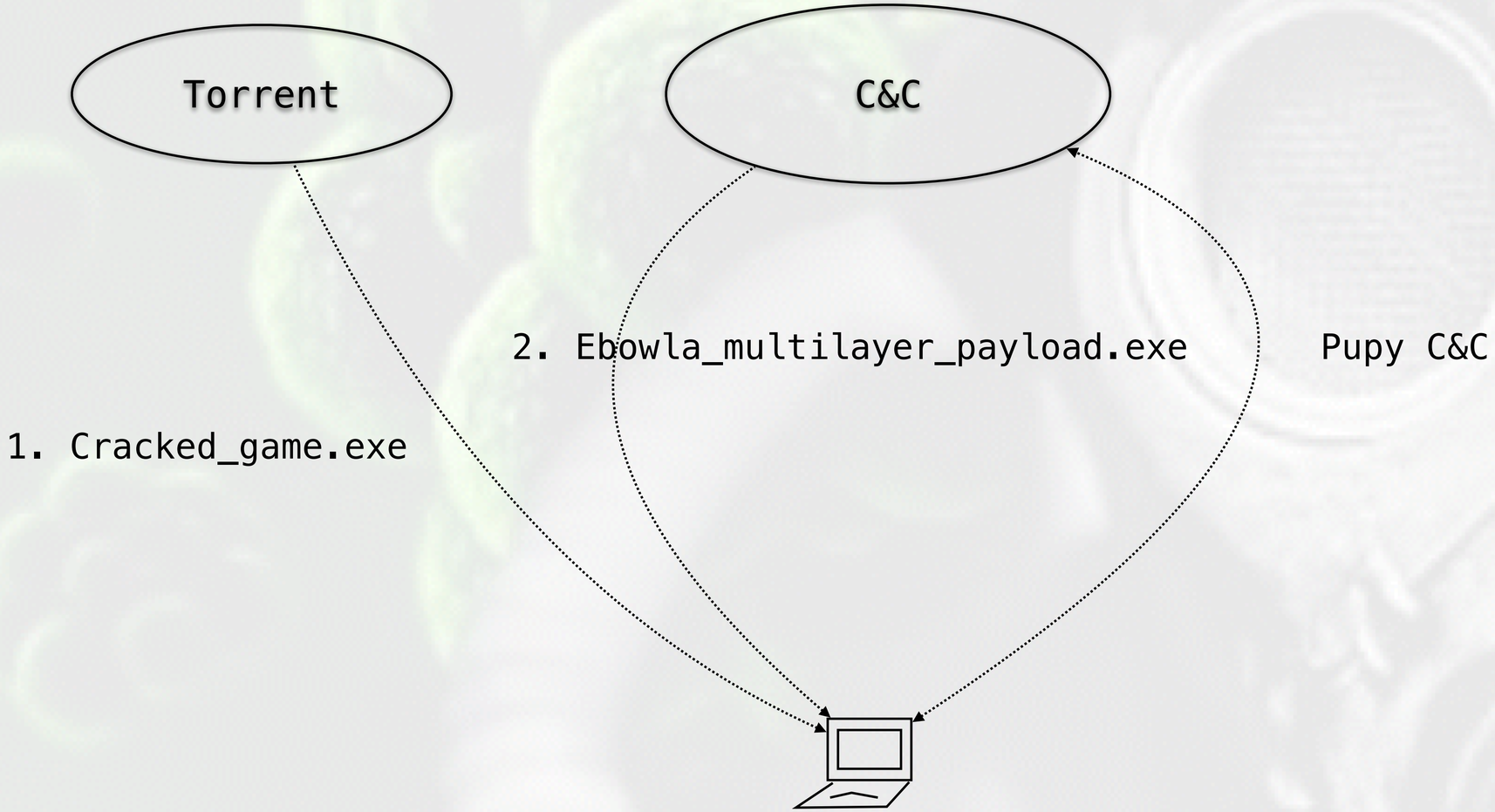
DEMO 3:Layered Payload



DEMO 3:Layered Payload



DEMO 3:Layered Payload



3. PyInstaller EXE => (disk)GO EXE => (memory)Pupy DLL

DEMO 4: Powershell

Deploy Empire via powershell

Known Issues/Bugs

- Chaining payloads:
  - GO EXE launching GO via Memory Module – DIE IN A FIRE
  - Pyinstaller EXE launching Pyinstaller EXE FROM DISK – Loses namespace
  - GO (memory module) -> Pyinstaller – Just no…
  - Metasploit x86 PE EXE template does not work with MemoryModule
- OTP:
  - MZ/DOS Header Leak

This is OK

- Go EXE
- PyInstaller EXE
- Chaining PyInstaller EXE -> GO EXE

Roadmap

- C++ loaders – In Progress
- Additional execution modules
- Better chaining of payloads
- OSX/Linux Support

Questions?

Download: https://www.github.com/genetic-malware/Ebowla

@wired33

@midnite_runr

Credits

https://github.com/vyrus001/go-mimikatz

https://github.com/clymb3r

https://github.com/mattifestation

https://archive.org/details/P-G_Ohst_Exploitation

https://matrixbob.files.wordpress.com/2015/03/bio-weapons.gif

http://blogs-images.forbes.com/benkerschberg/files/2015/02/crowdsourcing-spigot.jpg

http://static5.businessinsider.com/image/51e418a66bb3f7230a00000e-1200-900/guys-drinking-coffee-in-tel-aviv.jpg