Antivirus

Understanding & avoiding detection

DiabloHorn - https://diablohorn.com

Disclaimer

- This information may be outdated or incorrect :(
- This is not a full in depth tutorial into packers / crypters / etc.
- It is mostly focused towards meterpreter as an example
- Please correct me by writing your own blog post and keep spreading knowledge :)

Required knowledge

- Basic PE understanding
  - Knowing what sections are
- Basic scripting skills
  - Python / PowerShell
- Basic compiling skills
  - Able to compile C/C++ projects from github
- The will to research stuff
  - Lookup and research unknown terminology or concepts

Overview

- Common pitfalls
- Lab prerequisites
- AV detection methods
- Signature evasion
- Heuristics evasion
  - Packers / Crypters / etc
  - Payload transformations
- Building your own evasion
  - Meterpreter loaders
  - Shellcode executers

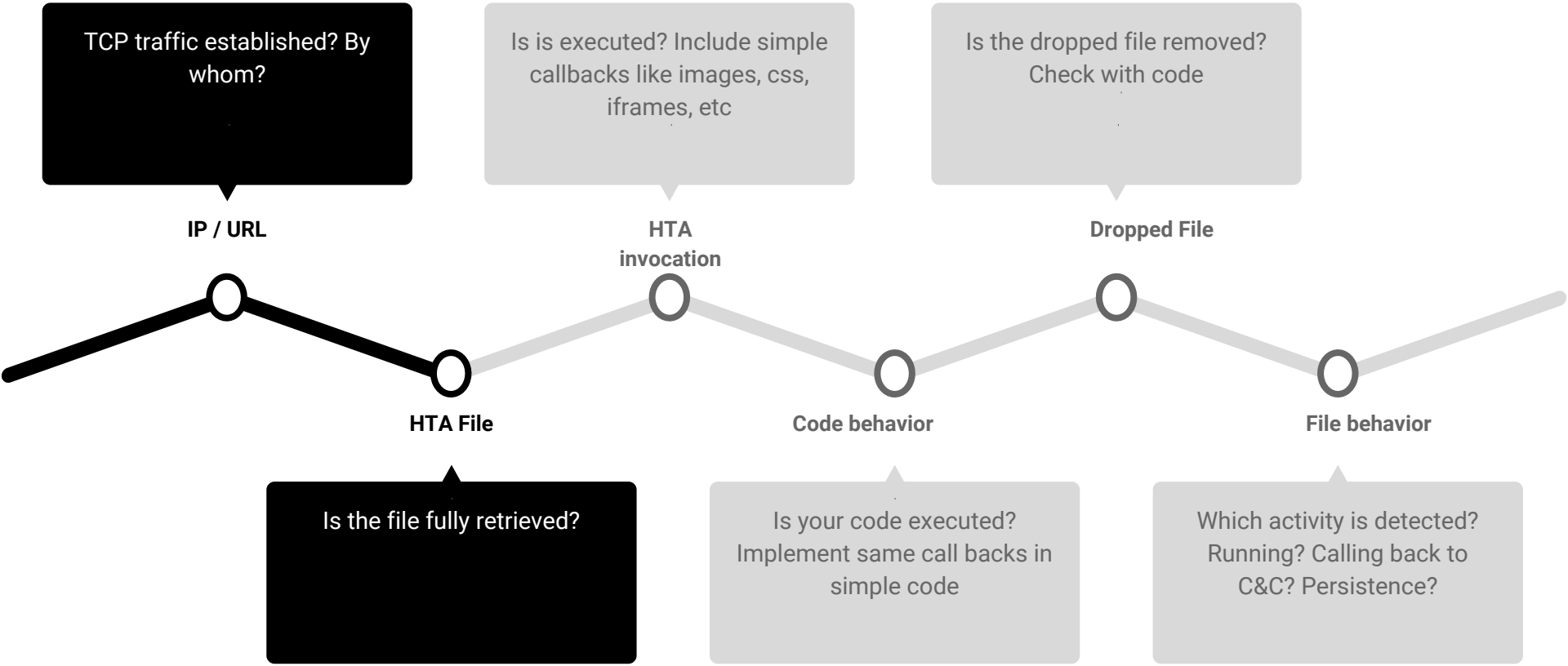
Lab prerequisites

- Linux VM
  - Metasploit
  - Hex editor
  - GIT client
- Windows 10 VM
  - Visual Studio
  - X64dbg
  - Windows Defender
    - or alternative AV engine

Pitfalls

- Submitting to online virus scan websites
- Avoiding the effort of compiling when the source is available
- Testing against the wrong AV
- Sticking to same executable over and over again
- Testing continuously live on your target
- Using a trial and having it internet connected
- Not looking into currently supported AV evasion features
- Not using search engines
    - and actually reading and trying the results
- Not knowing what part of your attack is actually being detected

HTA dropper analysis 101



AV detection methods

AV detection

- Signature based
- Heuristics based
- Sandbox based
- Cloud based

AV detection - signatures

- Detection
  - Can be byte / yara / hash / string based
  - Usually matches one or multiple places within the file
  - Usually file based
- Evasion
  - Identify match point
  - Adjust match point
  - Retain functionality

AV detection - heuristics

- Detection
  - Combination of behavior rules
  - Could include runtime information
  - Might contain rules for combination of API calls
  - Can take into consideration if signed or not
- Evasion
  - Avoid ‘malware’ behavior
  - Delay call functionality
  - Perform benign functionality until triggered
  - Obfuscate code

AV detection - sandbox

- Detection
  - Runtime behavior
  - APIs called during execution
  - Observes behavior
- Evasion
  - Avoid running inside the sandbox
  - Outrun the sandbox
    - Sandboxes have limited time & resources

AV detection - cloud based

- Detection
  - Submission of different information
    - Source URL
    - Hashes
    - Actions performed
  - Benefits from multiple clients exhibiting same behavior
  - Can match and identify code snippets in larger code base
  - Can take file reputation into consideration
- Evasion
  - Mostly the same as previous methods
  - Avoid having your sample submitted to the cloud
  - Generate unique samples per infection
    - Strip initial payload from as much functionality as possible

Signature evasion

Identify offending bytes

- Best guess based on strings
  - Strings identifying the tool
  - Strings unique to this tool
- Split the files into small pieces
  - Parts containing offending bytes will be deleted by AV
- Change the hash
  - Append ‘\x00’ to the file
  - Modify bytes with no functionality impact
    - Remember: hashes can be PE section based
    - Remember: hashes can be ‘fuzzy’ hashes
- Reverse the AV database

Lab 01

Finding the offending bytes

- Generate a meterpreter bind payload
- Copy it to your Windows 10 VM
- Avoid the file being removed by the AV

Lab 01

Resources

- http://obscuresecurity.blogspot.nl/2012/12/finding-simple-av-signatures-with.html
- https://www.adampalmer.me/iodigitalsec/2013/04/18/anti-virus-evasion-part-1/

Meterpreter AV evasion features

- Template based
  - C source
  - Custom executables
- Source available
  - The sky is the limit

Lab 02

Changing the core

- Edit the template
- Edit source & compile
- Is it enough to avoid the file being deleted?
- Can you also run it?

Lab 02

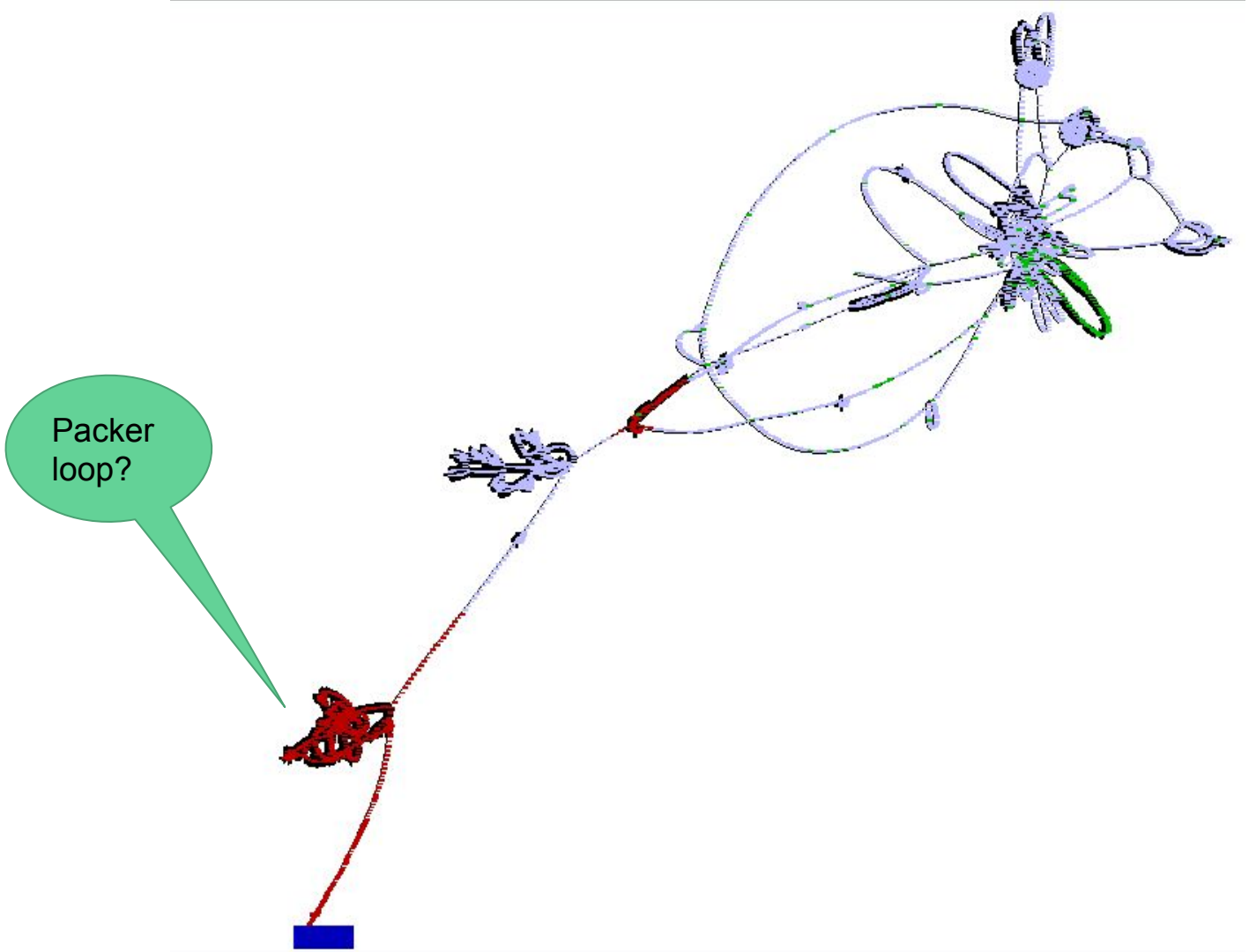
Resources

- https://www.blackhillsinfosec.com/modifying-metasploit-x64-template-for-av-evasion/
- https://diablohorn.com/2013/01/19/av-evasion-recompiling-optimizing-ftw/

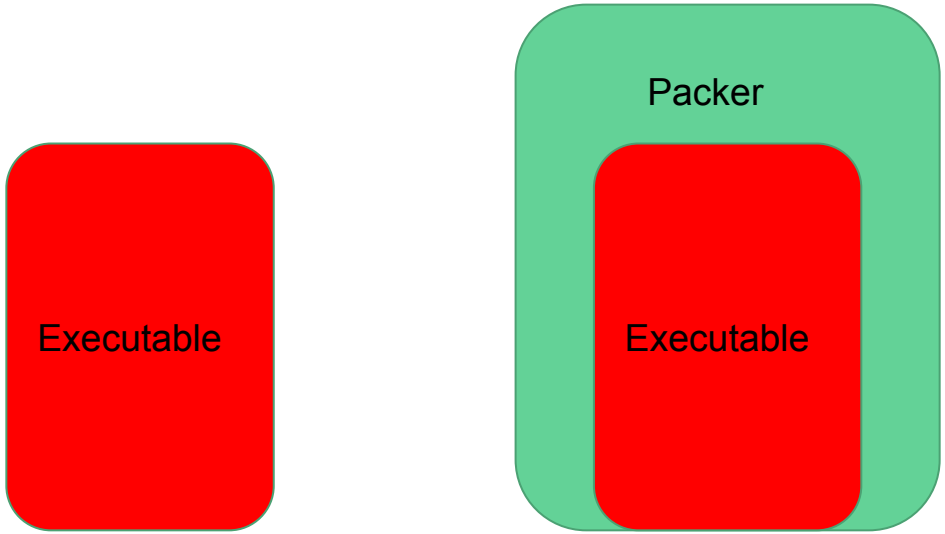
Heuristics evasion

Our arsenal

- Packers / Crypters / etc
- Transforming the payload
- (fake) Signing the executable

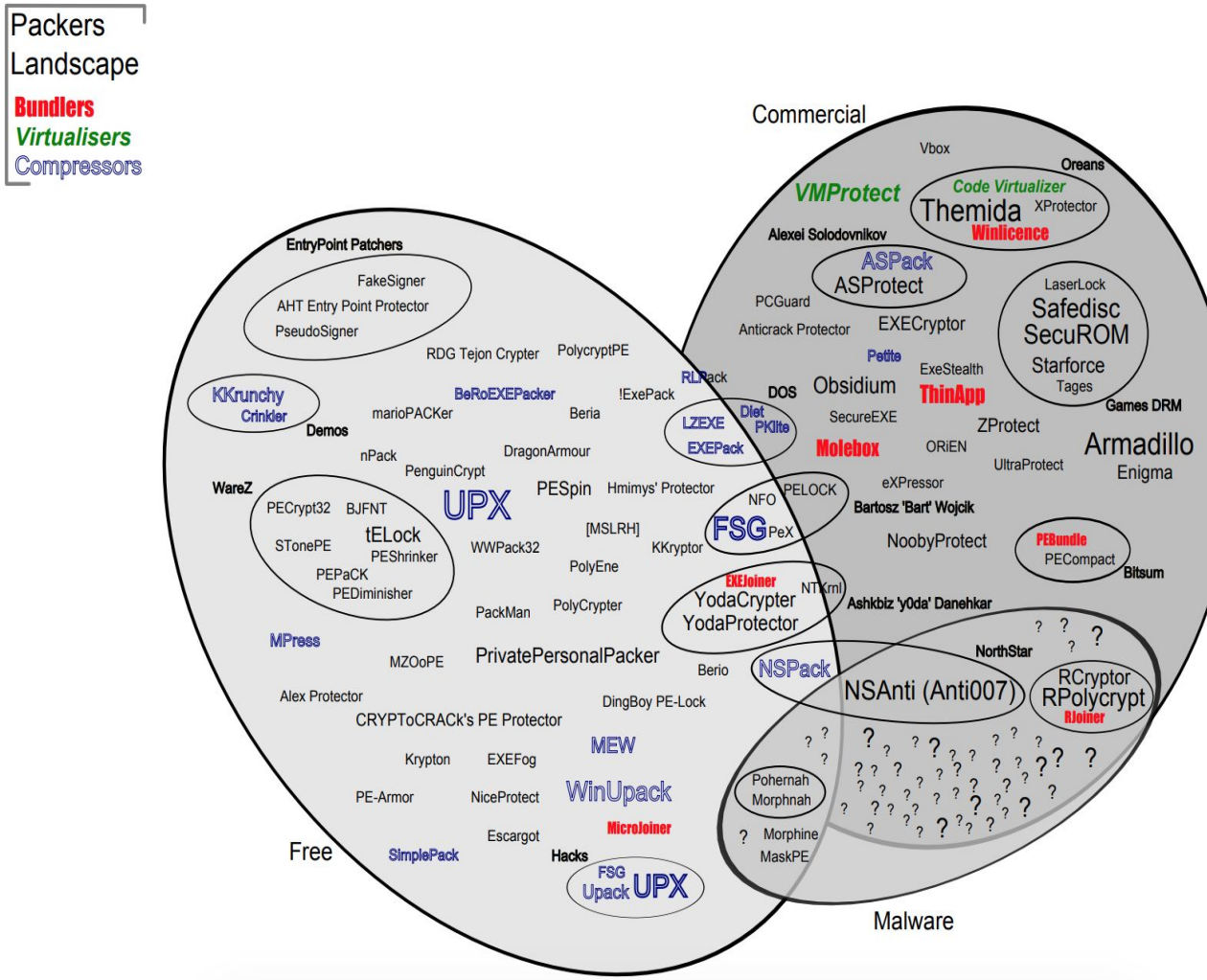

Packer
loop?

Packer theory *very* high over



Mirriad of options

https://github.com/corkami/docs/blob/master/packers.pdf



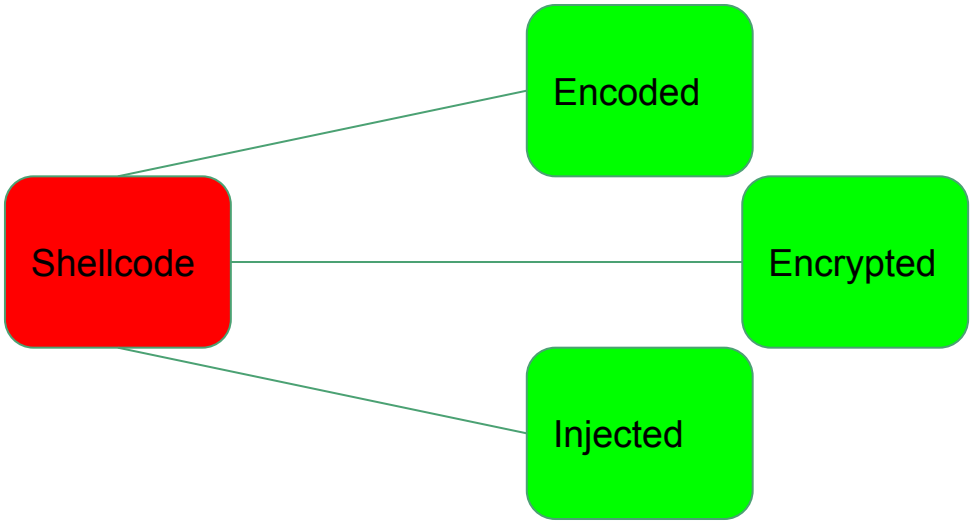
Lab 03

Using packers

- UPX
- Rcrypt
- Core-packer
- ASProtect

Payload transformation

- All code can essentially be reduced to a collection of assembly instructions
  - Also known as ‘shellcode’
- Everything (almost) an OS does you can emulate yourself
- Nothing stops you from emulating it using different languages



Lab 04

Playing with shellcode

- Shellcodeexec
- Shellter
- Veil
- Unicorn
- Syringe
- <scriptinglanguage>2exe

PE signatures

- Can be signed in multiple ways
- Prevents most of the tampering with an executable
- Indicates some kind of 'trust'
- Needs to be parsed, verified
- Somehow it sometimes also means 'it is ok, I'm not malicious' even when the signature is invalid

https://pentestlab.blog/2017/11/06/hijacking-digital-signatures/

https://blog.didierstevens.com/2008/12/31/howto-add-a-digital-signature-to-executables/

Lab 05

Sign all the things

- Copy a signature
- Create a normal signature
- What are the effects?

Building your own

Do you need it?

- Can you achieve your goal using a different tool?
- Did you try different bypass combinations?
- Are you really sure your payload is being detected?
  - What about your dropper?
- What is the bare minimum that you need?
  - Hint: Download, Execute & Persist
- Try to improve existing tools instead of ‘yet another bypass evasion tool / poc’
  - Yes I’m guilty of not doing this myself :(
- Did you analyse WHY your payload is being detected?
  - Don’t just try random stuff hoping it works
  - You need to KNOW what needs to be bypassed :)
- See references for an overview of options to build your own

References

References - I

- http://unprotect.tdgt.org/index.php/Antivirus_Evasion
- http://hooked-on-mnemonics.blogspot.nl/2011/01/intro-to-creating-anti-virus-signatures.html
- http://www.thegreycorner.com/2010/04/bypassing-av-detection-netcat.html
- https://www.gracefulsecurity.com/anti-virus-evasion/
- https://www.cyberark.com/threat-research-blog/illusion-gap-antivirus-bypass-part-1/
- https://pentest.blog/art-of-anti-detection-1-introduction-to-av-detection-techniques/
- https://www.blackhillsinfosec.com/modifying-metasploit-x64-template-for-av-evasion/

References - II

- https://averagesecurityguy.github.io/learn/research/2011/04/20/using-metasploit-templates-to-bypass-av/
- https://dl.packetstormsecurity.net/papers/bypass/bypassing-av.pdf
- https://www.securitysift.com/pecloak-py-an-experiment-in-av-evasion/
- http://securityxploded.com/bypassing-antivirus-using-code-injection.php
- https://marcoramilli.blogspot.nl/2012/02/new-way-to-detect-packers.html
- https://github.com/hackedteam/core-packer
- https://github.com/trustedsec/unicorn
- https://github.com/securestate/syringe
- http://vxer.org/

References - III

- http://www.0xrage.com/?p=210
- https://diablohorn.com/2011/12/10/remote-av-detection-with-eicar/
- https://diablohorn.com/2013/01/20/hash-encapsulation-to-bypass-av/
- https://diablohorn.com/2013/01/19/av-evasion-recompiling-optimizing-ftw/
- https://diablohorn.com/2013/02/21/we-bypassed-antivirus-how-about-idsips/
- https://diablohorn.com/2013/02/04/evade-antivirus-convert-shellcode-to-c/