

Picking the right needles from the malware haystack

**REVERSING
LABS**

Katja Pericin

BSidesLjubljana::ox7E2



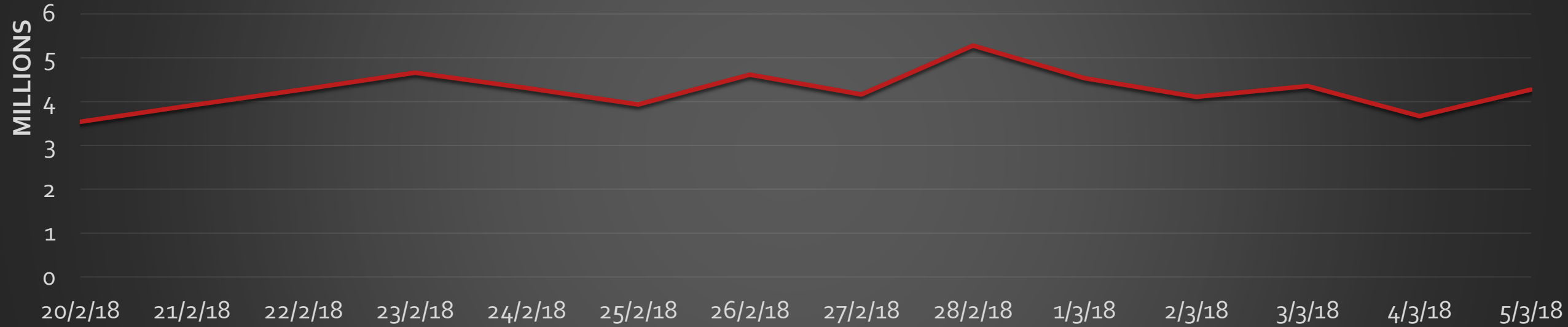
Outline

- Finding relevant data
- Methodology
- Custom packers
- Downloaders
- Results
- Try this at home

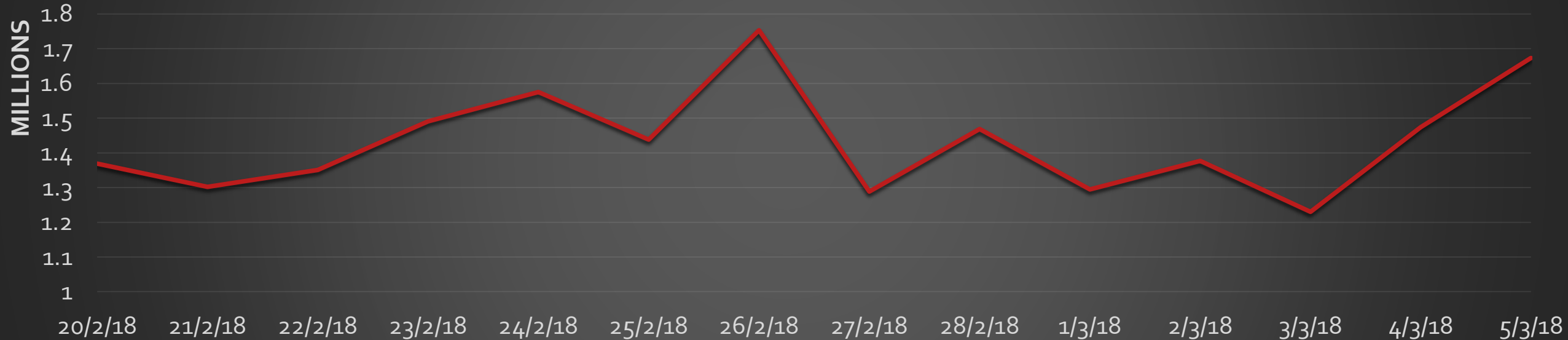


Finding relevant data

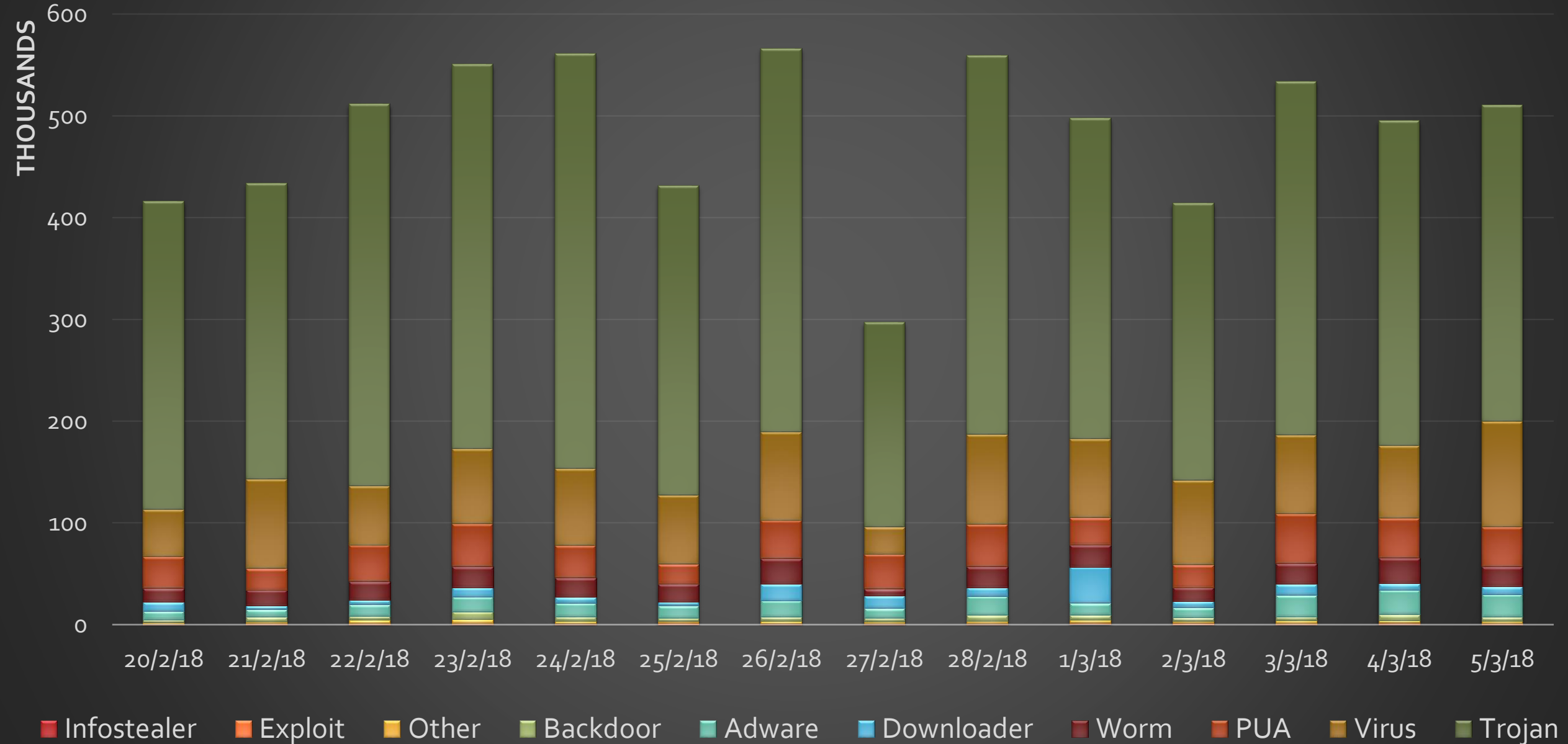
Daily sample count



Daily new sample count



Malware count by type





Methodology

Initial sample set

- 10-15 Upatre samples
- Both packed and not packed
- Written in assembly and in C++
- Ranging from few KB to hundreds of KB in size
- Found collection of Upatre hashes and used it as a starting point [1]

Code similarity

- Imphash – Import Table hash
- ReversingLabs Hashing Algorithm (“RHA”)
 - Correlation of files based on functional features
 - Multiple precision levels
 - Lowest level used to match as much files as possible
- Initial sample set yielded 342 RHA buckets
- Focused only on ones with 1000+ samples
 - 4 custom packers
 - 2 downloaders

Yara rules

- As loose as possible
- Focused on the most important parts
 - Custom decryption methods
 - Internal data parsing
 - Configuration parsing
- Found even more RHA buckets
- No new custom packers or downloaders
- Found samples are sometimes classified with the name of the family they download (zbot, dyre, ...)



Elastic Search/Kibana

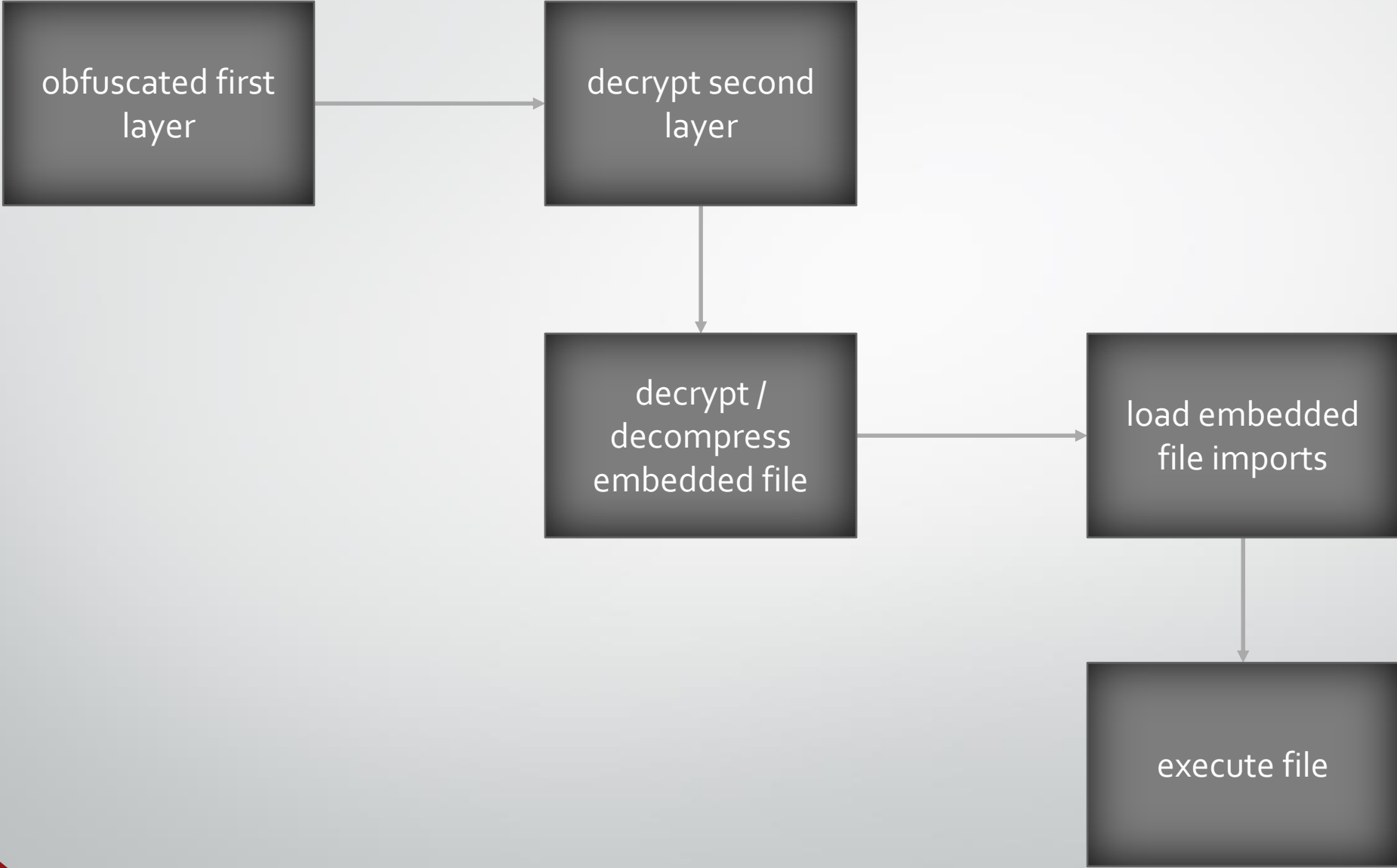
- For quick check of data
- Various searches and data correlation
- All results can be exported as .csv files



Custom Packers

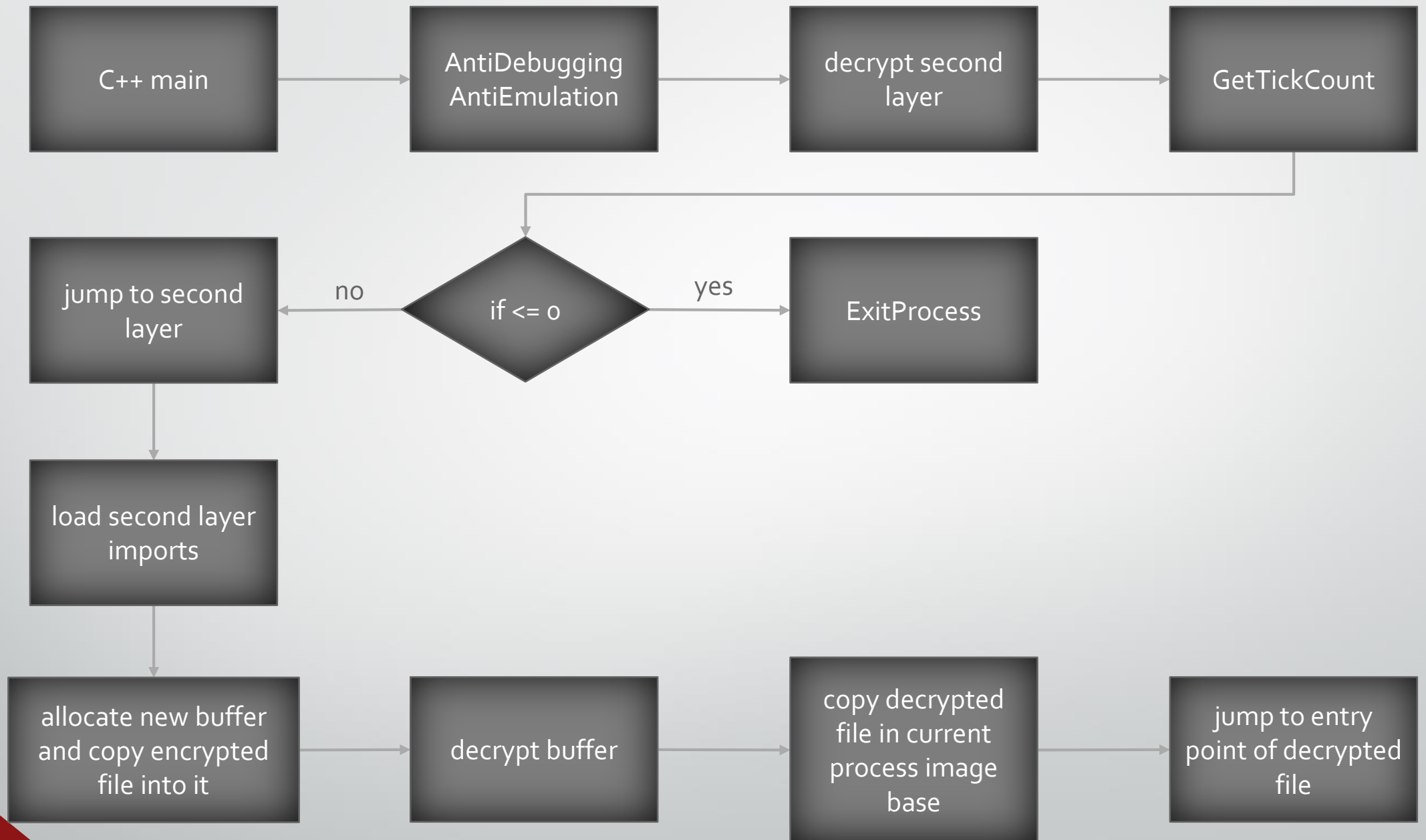
Custom packers

- Four different custom packers which pack Upatre downloader
- Run PE
- Anti-debugging and anti-emulation
- Their code differs significantly but all follow similar principles



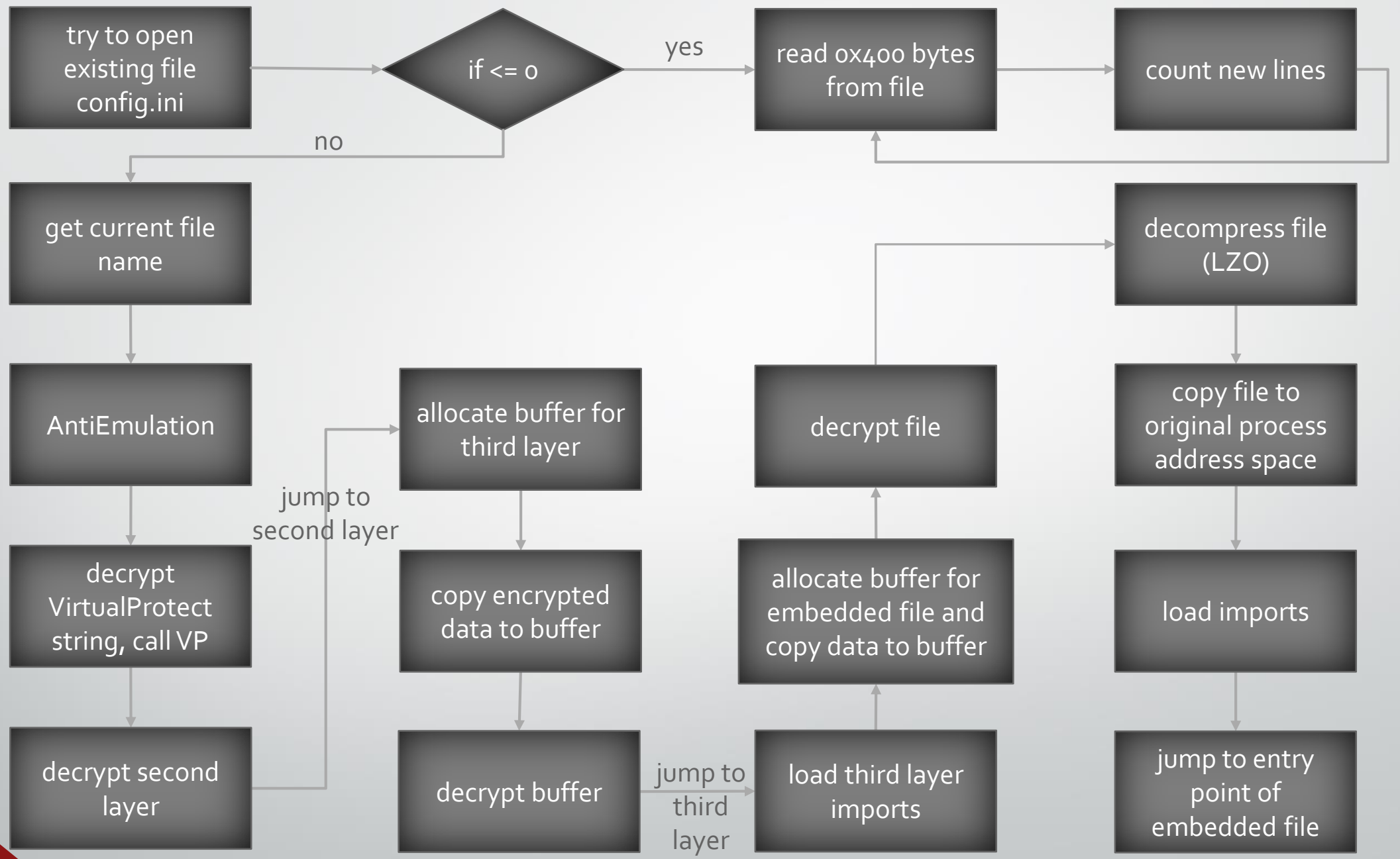
cpDalek

- Custom packer written in C++
- Most commonly used packer in samples we have observed
- 24 versions which unpack multiple versions of downloader
- Differences between versions in main function
- Embedded files are encrypted



cpTooly

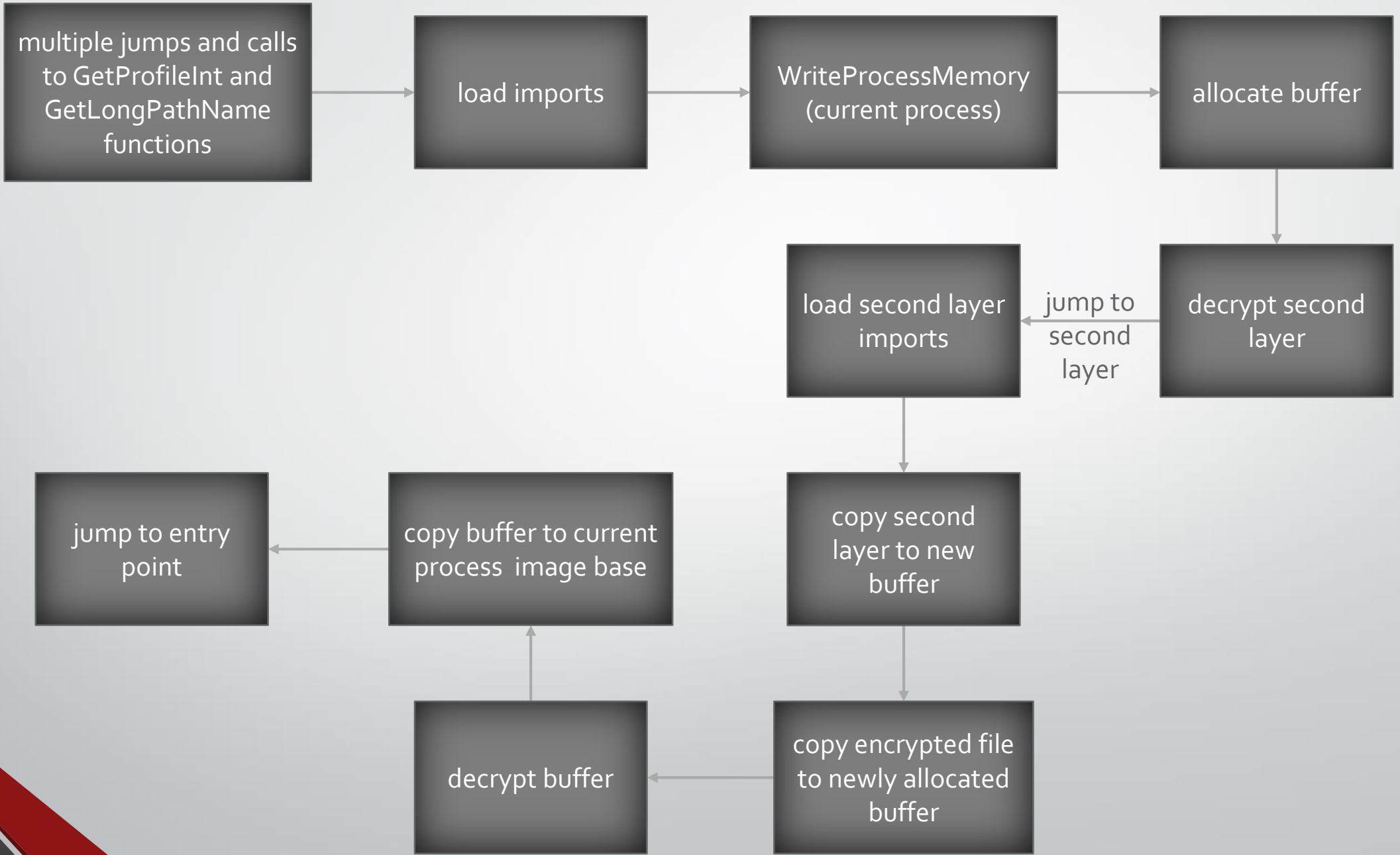
- Lots of anti-emulation code
- Embedded files are encrypted and compressed
- Written in assembly





cpEllie

- Lots of anti-emulation code
- Obfuscated
- Written in assembly



cpLupus

- Polymorphic custom packer for Upatre with many variants
- 20+ versions of polymorphic first layer
- Second layer always the same
- Embedded file is encrypted using modified RC₄ algorithm
- Detailed analysis of one of cpLupus variants can be found in [2]



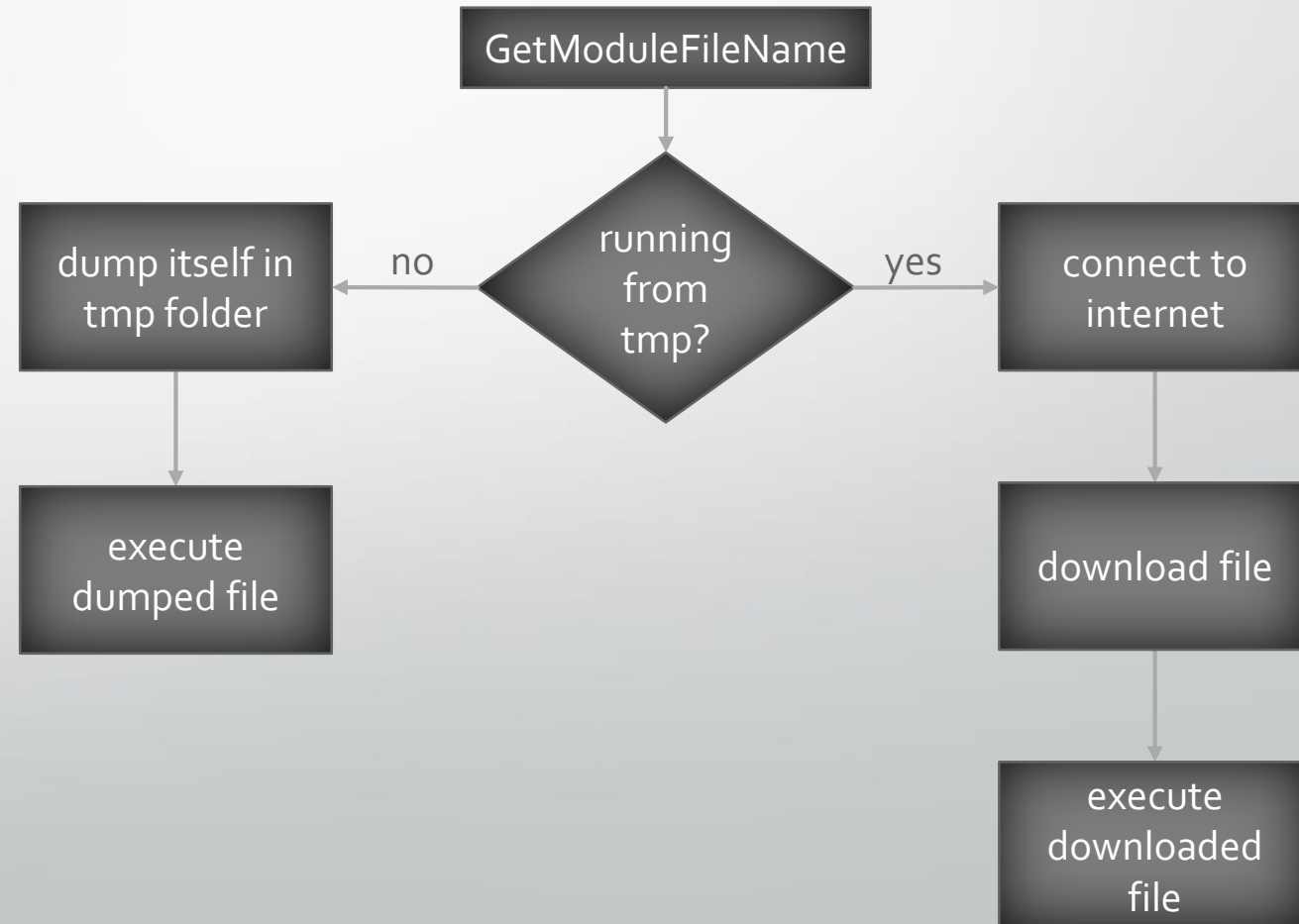
Downloaders

Downloaders

- Differences between downloaders are too significant to consider all of them a single malware family
- 91 versions of downloaders which spread across multiple families
- Families differ in
 - programming language
 - anti-reversing techniques
 - layout of configurations

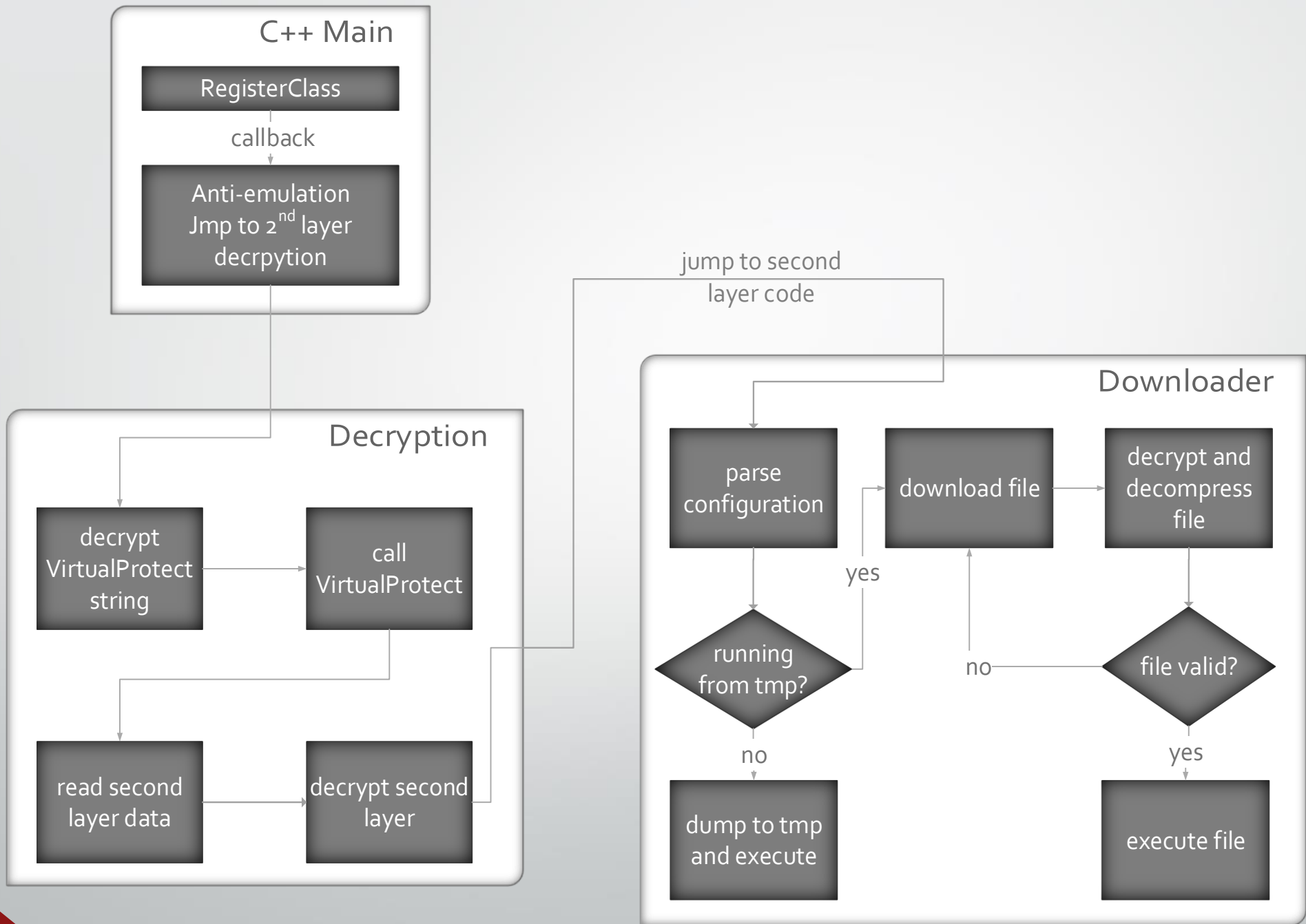
dIThunder

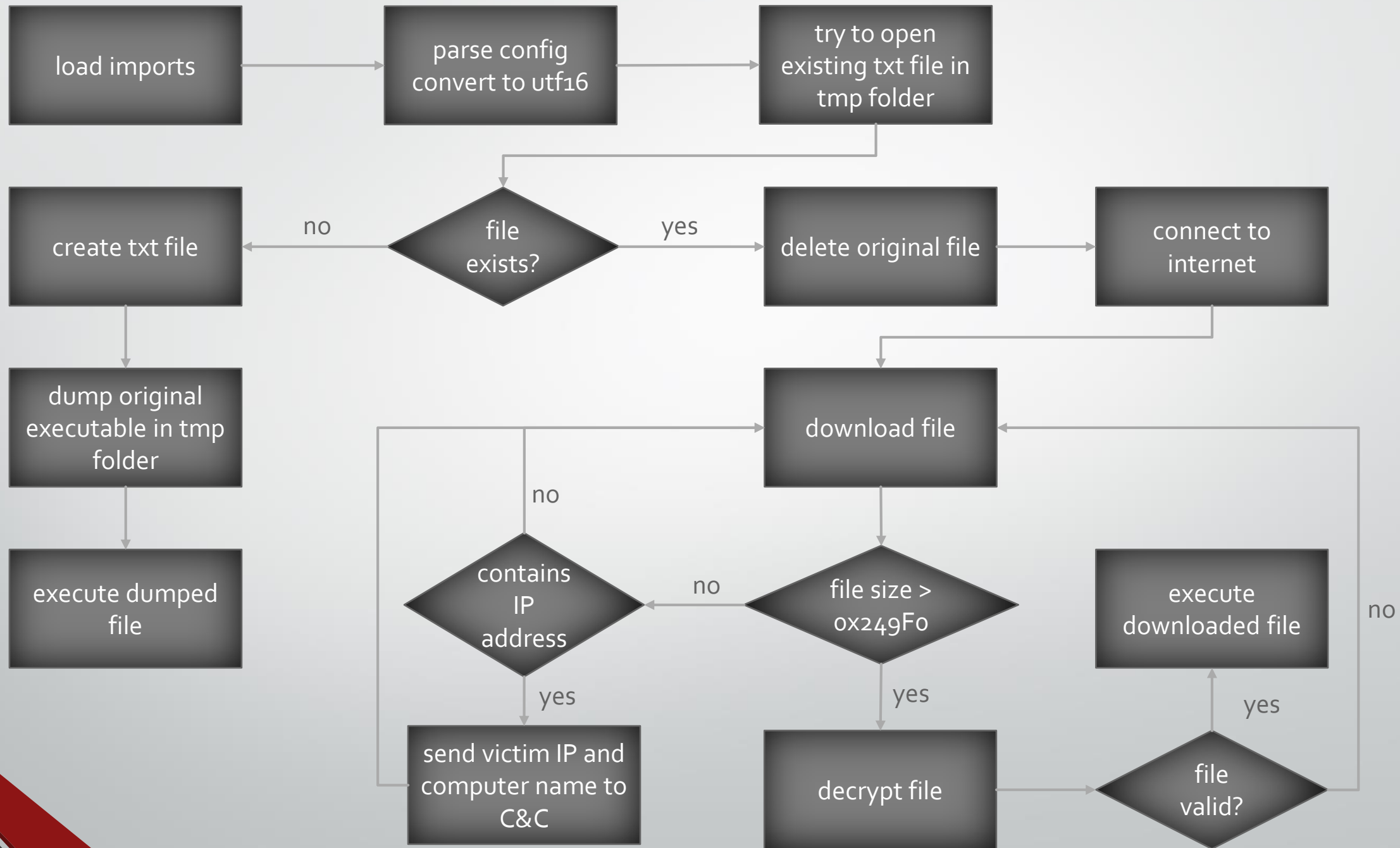
- The simplest downloader classified as Upatre
- Often seen unpacked and unprotected
- Can be found in overlays of other custom packers

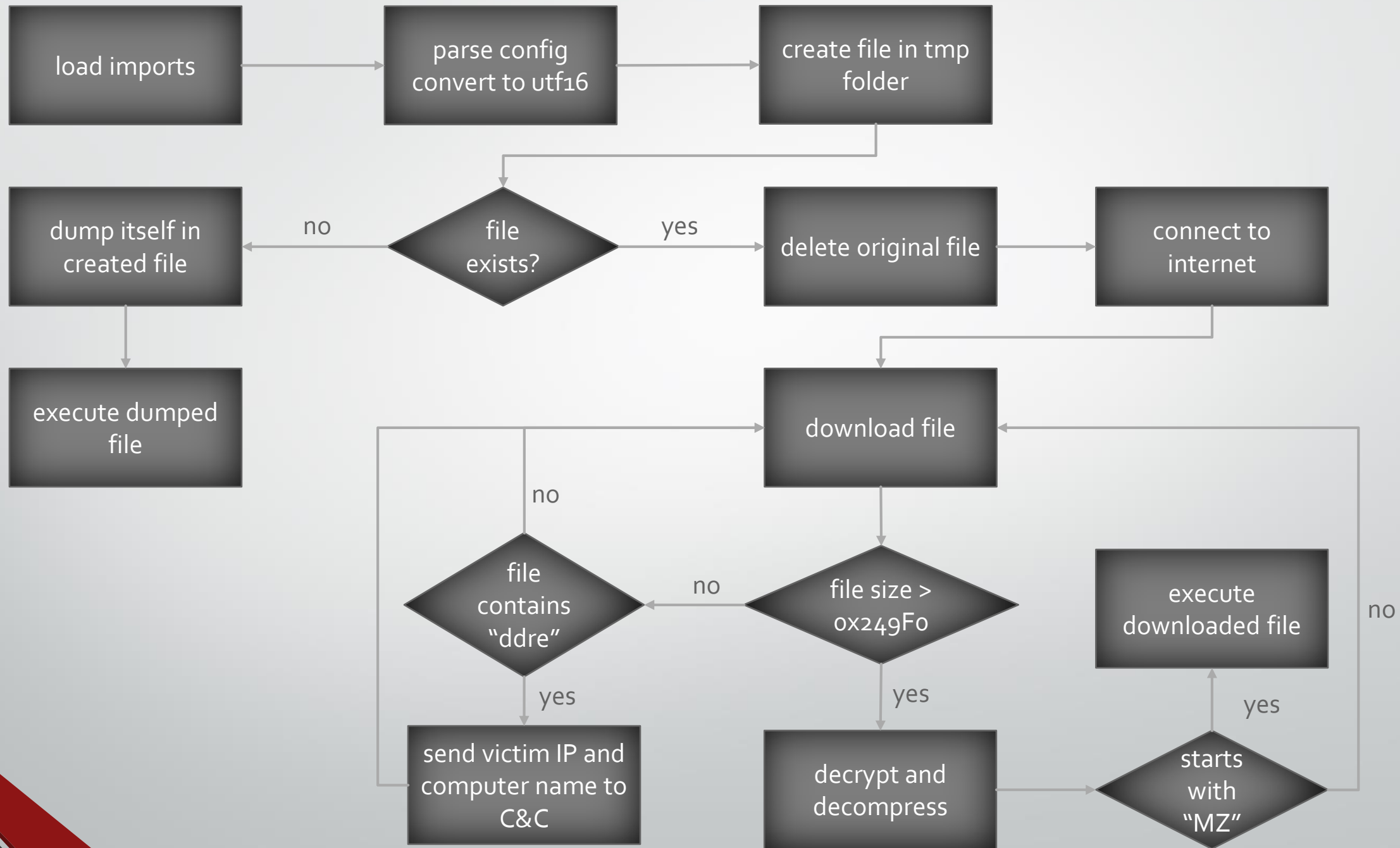


dIUpatreA and dIUpatreB

- Written in C++
- 81 versions
- Most of the differences between versions on the first layer
- Mimics ordinary application with UI
- Use *RegisterClass* callback to execute malicious code





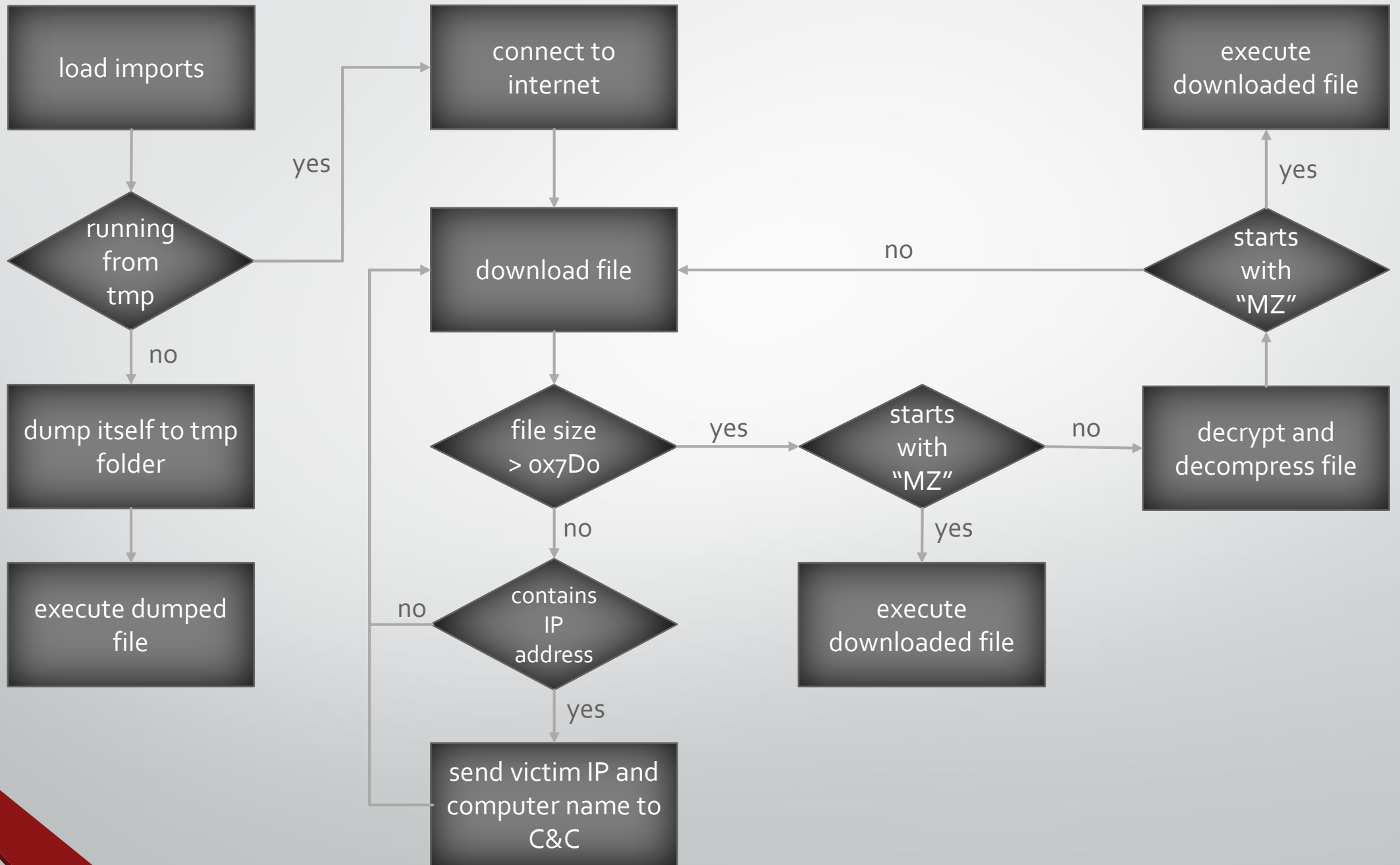


dIUpatreC

- The only Upatre version which is protected by a custom packer
- 6 versions of dIUpatreC with size of only 5 KB
- It can have encrypted configuration (single byte XOR which is usually 0x13)
- Reminds of second layer of dIUpatreA and dIUpatreB
- Detailed analysis of dIUpatreC can be found in [2]

dlUpatreD

- The oldest version of Upatre
- Written in assembly
- Uses anti-emulation techniques and RegisterClass callback
- Different configuration when compared to other versions



dIUpatre configurations

- dIUpatreA, dIUpatreB and dIUpatreC have the same configuration [2]
- Mostly variable sized and null terminated strings
- Structures at the end hold indexes into string array

Configuration V1

port
printf format string
printf format string
open
HTTP content type
HTTP content type
HTTP request type
user agent
dump file name
C&C server
string₁
string₂
...
string_N
terminator = 0x01
K = key count
K keys
M = struct count
M structs

Configuration V2

port
printf format string
printf format string
open
HTTP content type
HTTP content type
HTTP request type
user agent
dump file name
tmp file name
C&C server
string₁
string₂
...
string_N
terminator = 0x01
K = key count
K keys
M = struct count
M structs

Struct V2

path_index
server_index
binary_index
reserved0
reserved1
file_index
reserved2

Struct V1

path_index
server_index
binary_index
reserved0
reserved1
file_index

Configuration V1

port
dump file name
save file name
printf format string
printf format string
open
user agent
HTTP content type
HTTP content type
server1
server2

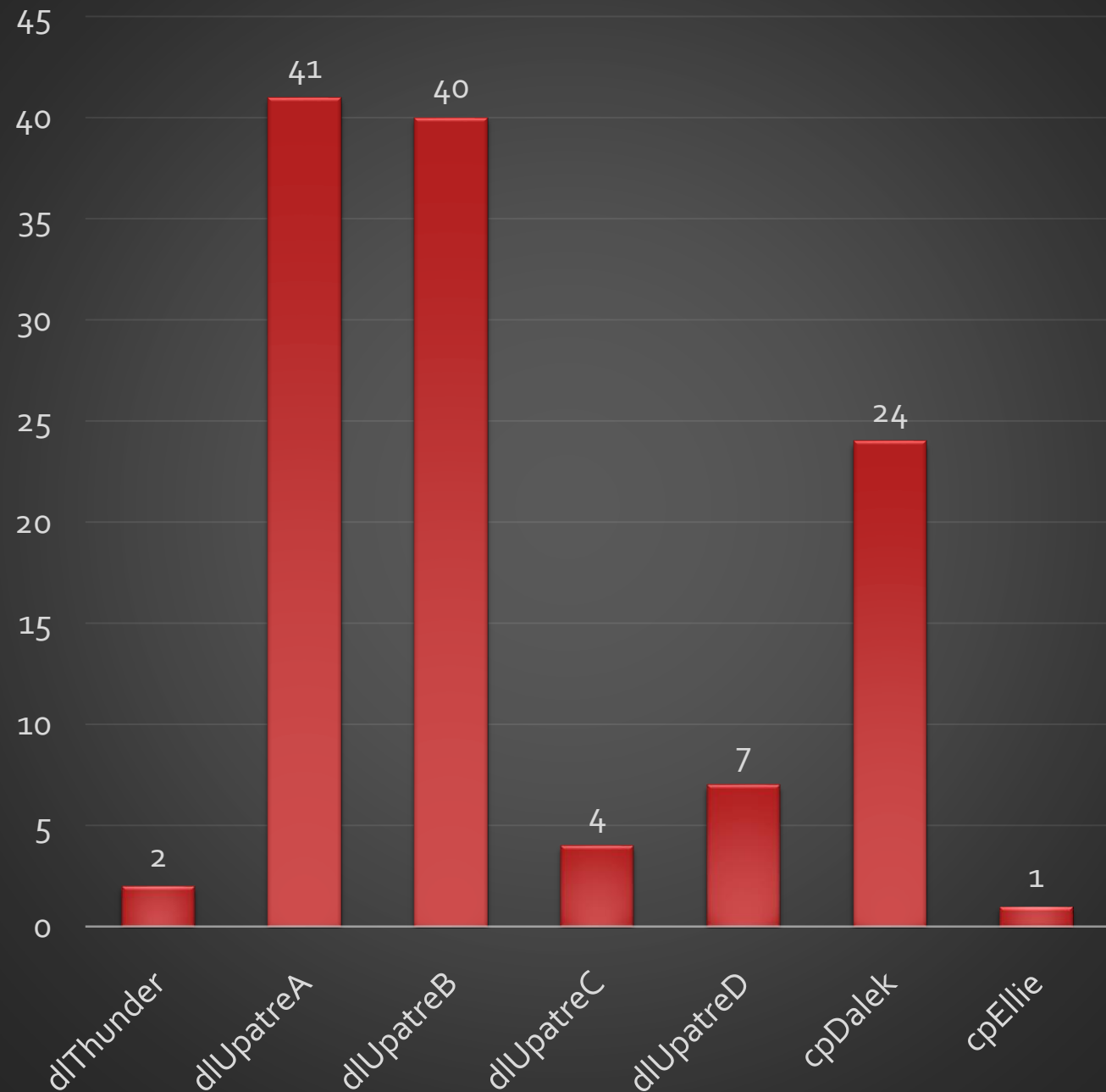
Configuration V2

port
dump file name
save file name
printf format string
printf format string
open
user agent
HTTP content type
HTTP content type
server1
server2
server3
server4

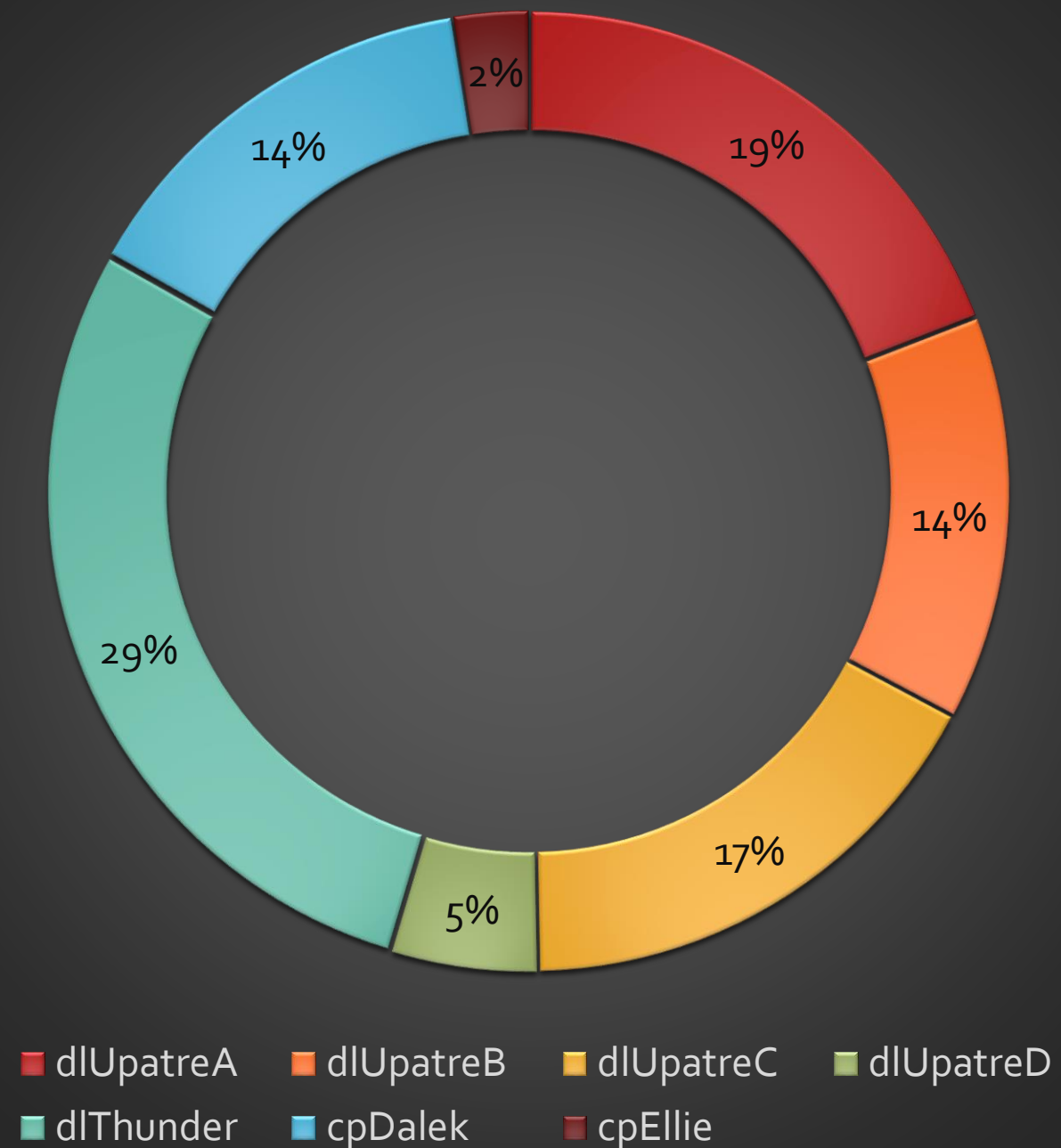


Results

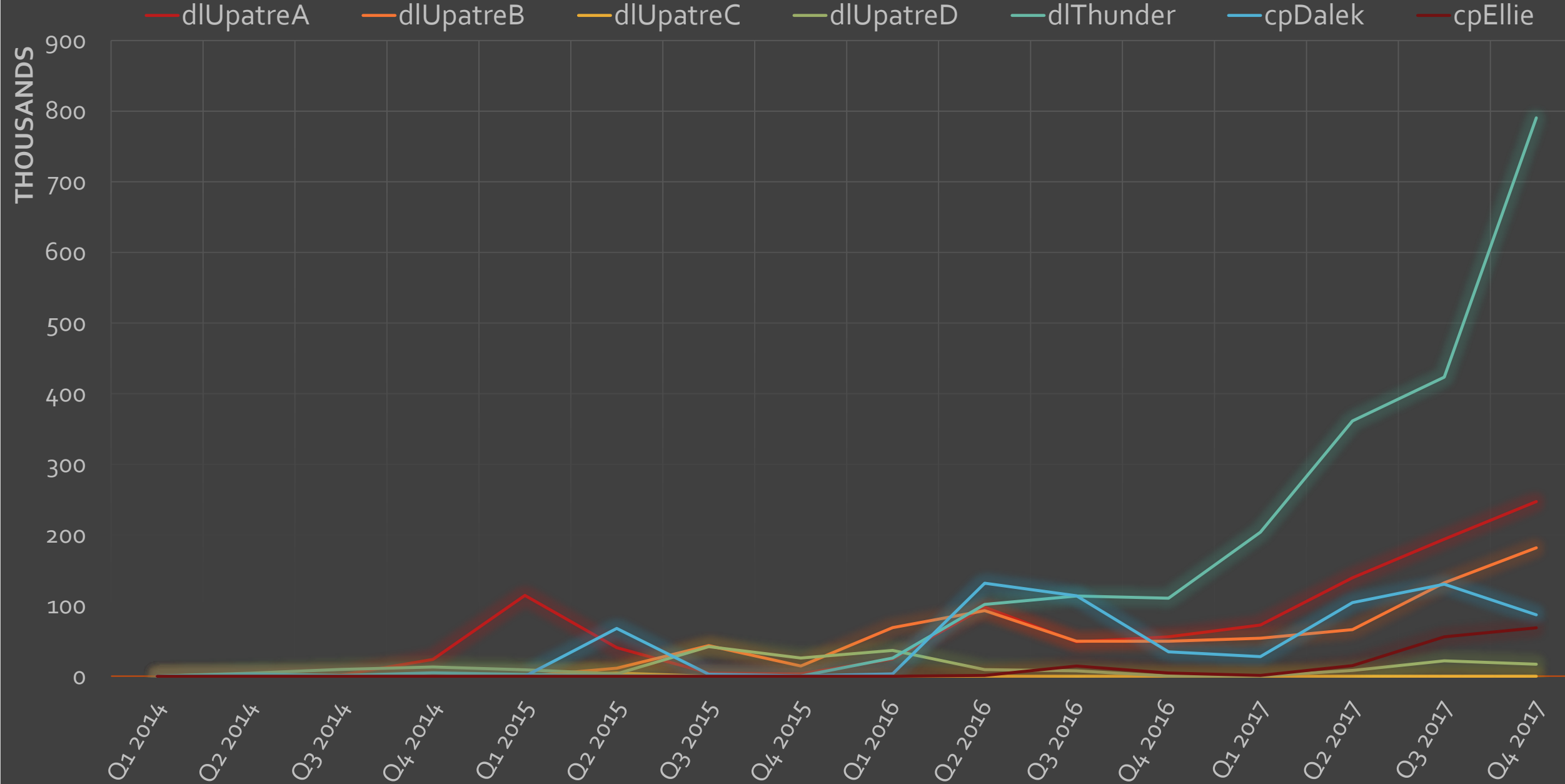
RHA Buckets



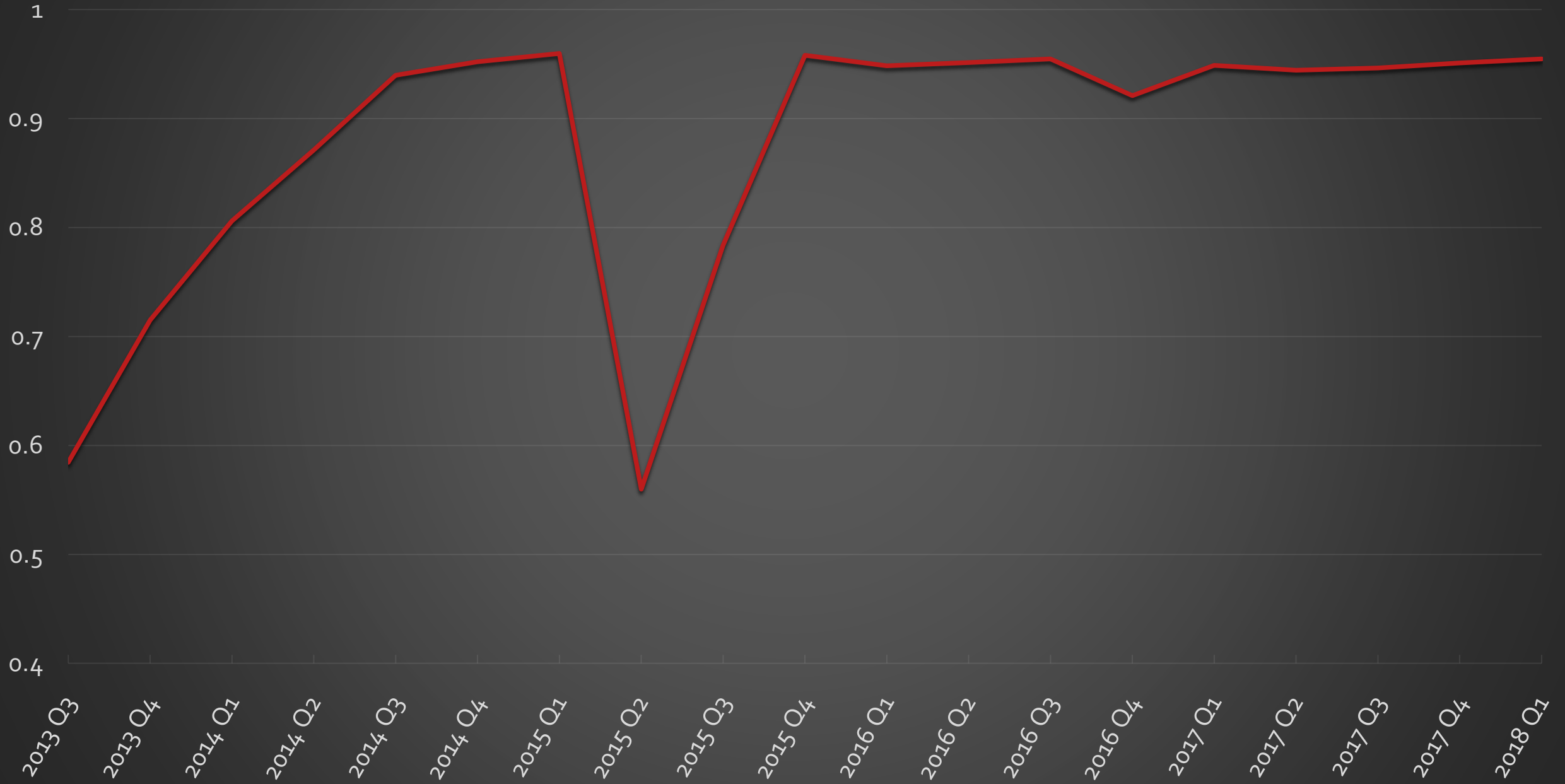
Upatre samples by family



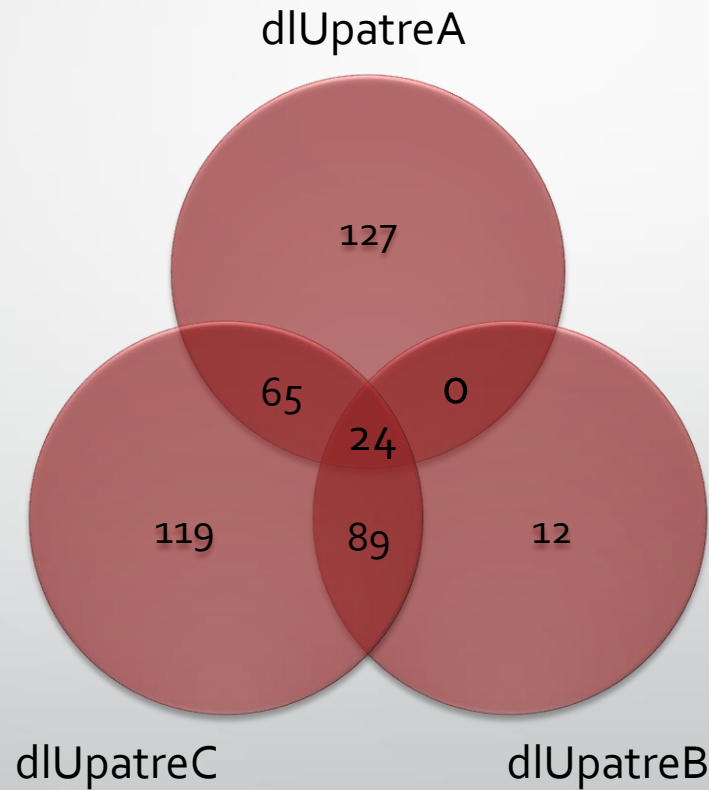
Sample counts



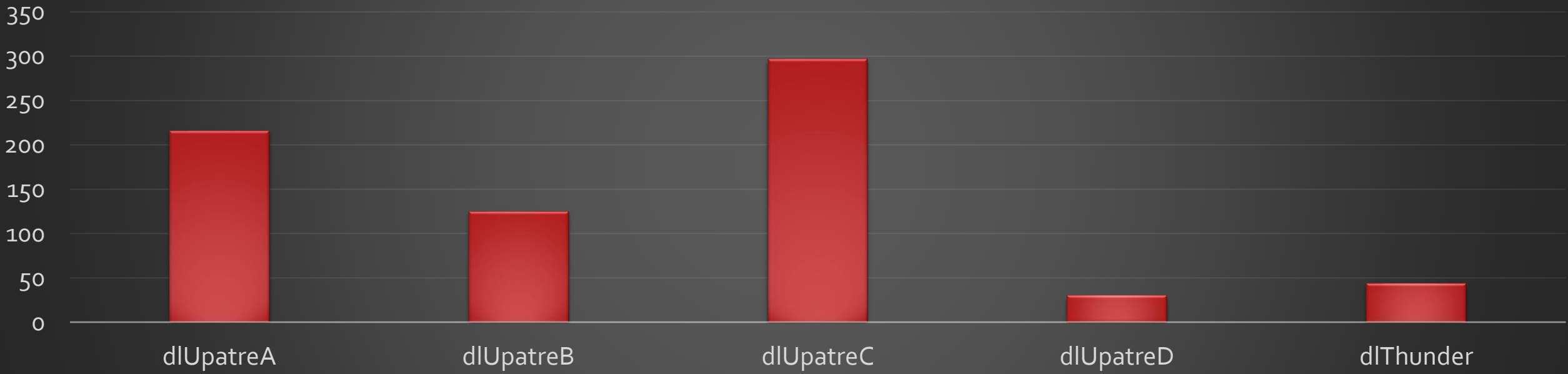
AV detection rates



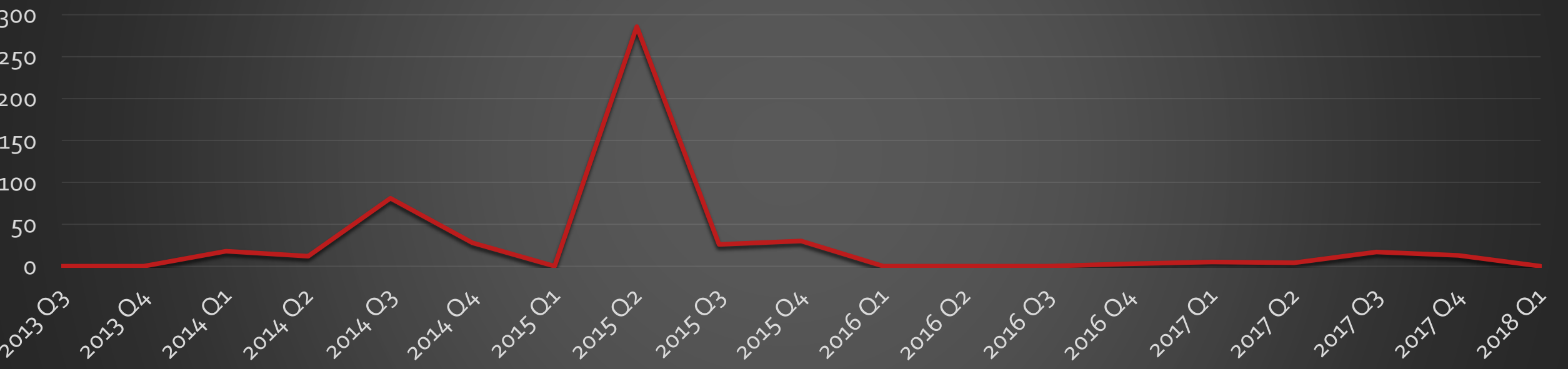
Upatre infected websites

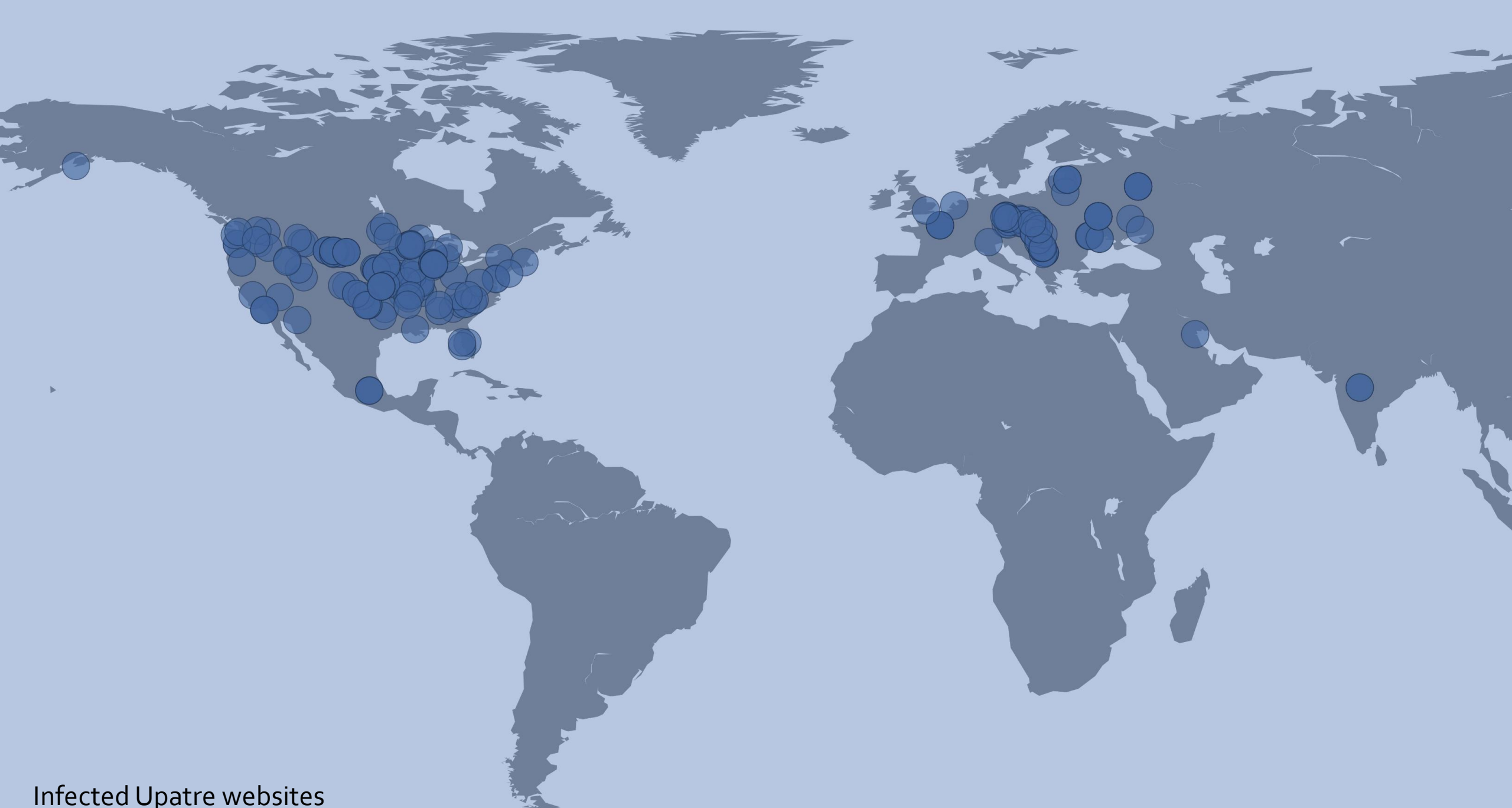


Unique server count by family



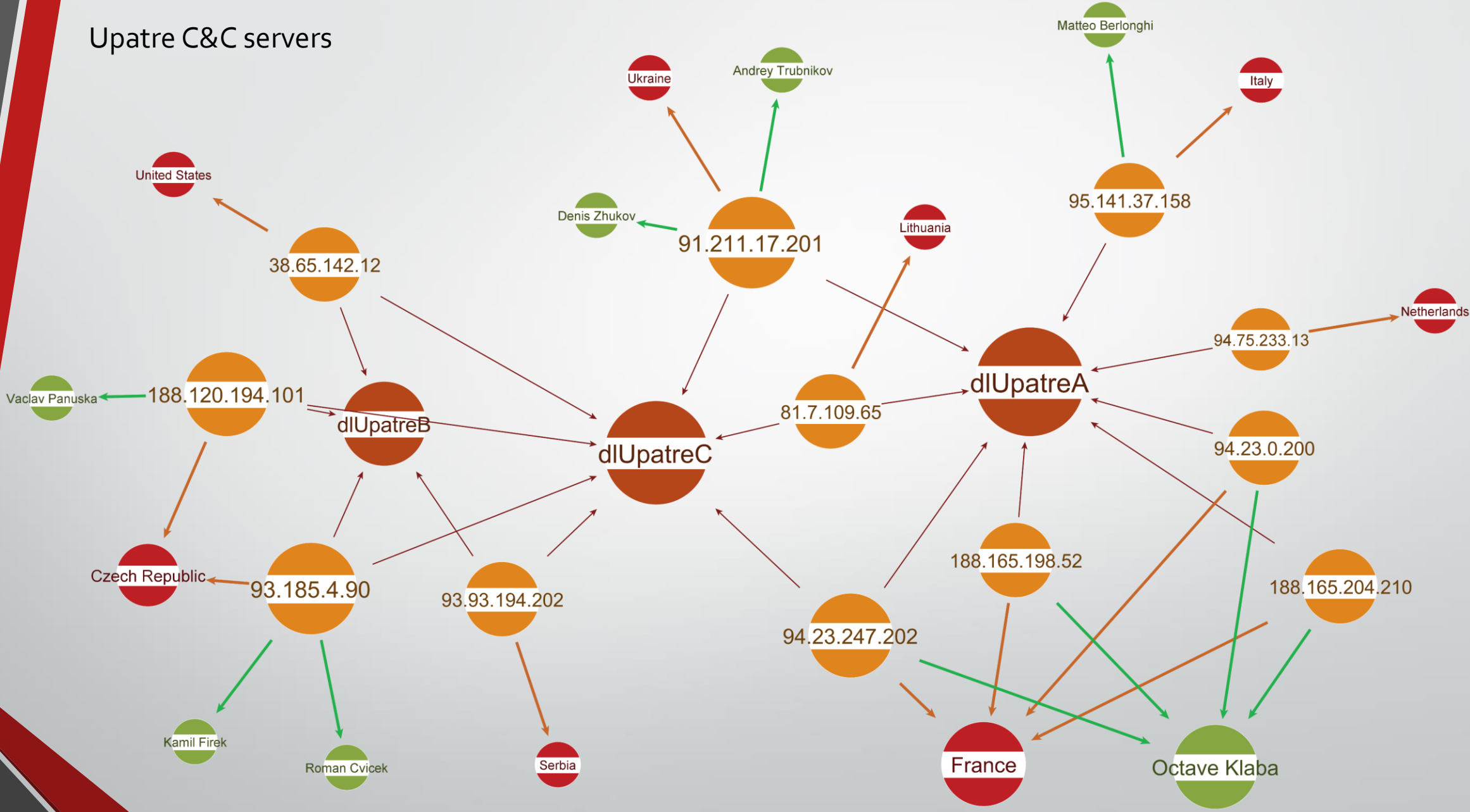
Unique server count per quarter






Infected Upatre websites

Upatre C&C servers





Try this at home

Try this at home

- Choose family of interest
- Find few samples which differ as much as possible
- Use code similarity (imphash, or create your own) to find even more samples
- Write small and loose Yara rules to capture even more diversity
- Analyse samples to extract interesting info
- Load results in Elastic Search and visualise with Kibana
- Use Maltego transforms for additional info (Whois, DNS lookup, VirusTotal API, ...)

Bibliography

[1] J. Bader, "Collection of Upatre Samples," [Online]. Available: <https://www.johannesbader.ch/projects/upcol.php>

[2] J. Bader, "Win32/Upatre.BI - Part One," [Online]. Available: <https://johannesbader.ch/2015/06/Win32-Upatre-BI-Part-1-Unpacking/>

[3] KoreLogic, "Callback Functions in Malware," 27 05 2014. [Online]. Available: https://blog.korelogic.com/blog/2014/05/27/malware_callback

[4] T. H. a. J. C. Brandon Levene, "Upatre: Old Dog, New [Anti-Analysis] Tricks," 20 11 2015. [Online]. Available: <http://researchcenter.paloaltonetworks.com/2015/11/upatre-old-dog-new-anti-analysis-tricks/>

[5] Symantec, "Dyre: Emerging threat on financial fraud landscape," 13 June 2015. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/dyre-emerging-threat.pdf

[5] B. Griffin, "The Many Faces of Gameover Zeus," 02 05 2014. [Online]. Available: http://cdn2.hubspot.net/hub/241665/file-951587296-pdf/T3_SpecialTopicReport_05_02_2014.pdf

[7] ReversingLabs, "ReversingLabs Hashing Algorithm," ReversingLabs, [Online]. Available: <https://www.reversinglabs.com/technology/reversinglabs-hash-algorithm.html>

[8] ReversingLabs, "Malware Analysis Solution - TitaniumCore," [Online]. Available: <https://www.reversinglabs.com/products/malware-analysis-solution.html>

[9] Elasticsearch, "Kibana," [Online]. Available: <https://www.elastic.co/products/kibana>

[10] Paterva, "Maltego CE," [Online]. Available: <https://www.paterva.com/web7/buy/maltego-clients/maltego-ce.php>