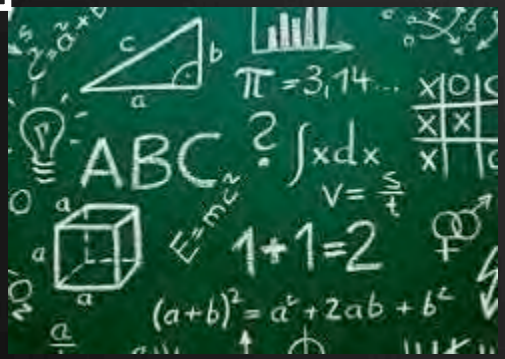
I am packer and so can you

Mike Sconzo



Agenda*

*Powered by business synergy

It's all about me!

- Threat Research at Bit9 + Carbon Black
- Enjoy static analysis, machine learning, network forensics, and eating BBQ and pie (yes, I live in TX)
- SecRepo.com
- @sooshie

What's the problem?

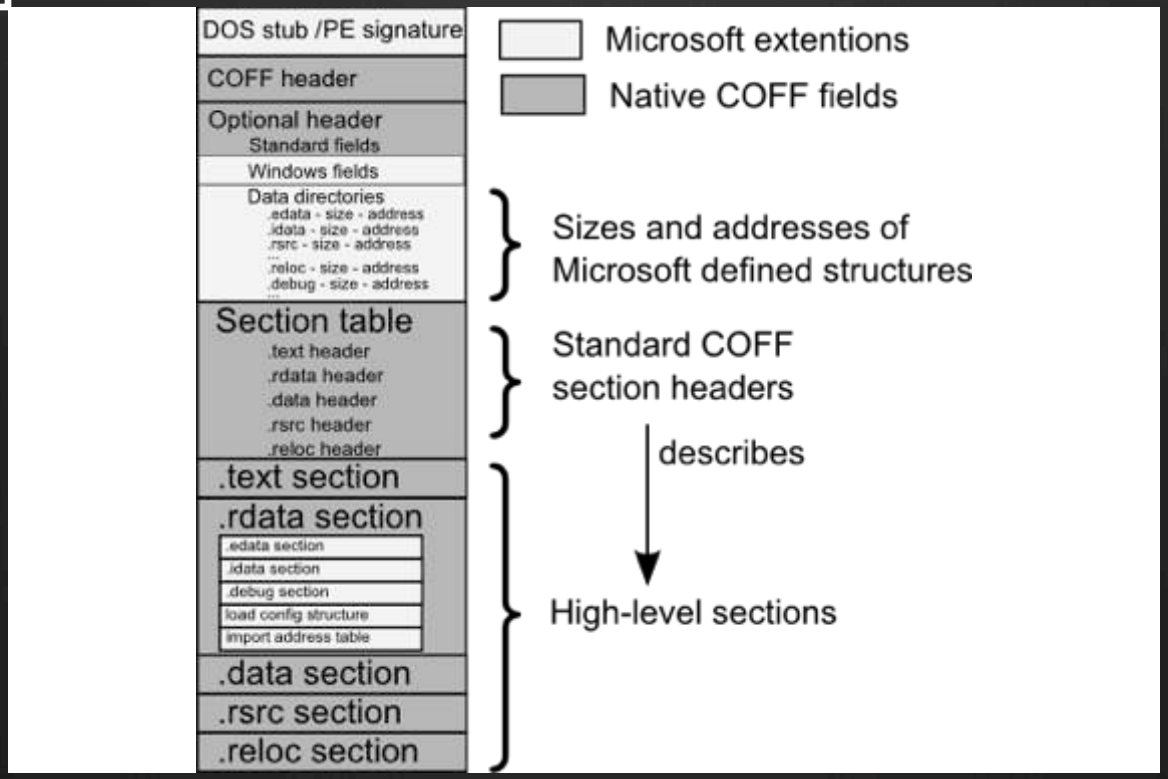
Detecting packers, compilers and various artifacts from the development cycle can be really useful when looking at malware or files in general. However, the de facto standard (PEiD) was created over 10 years ago, and there are very few recent signature updates. Is it time for something new and different?

Moar?

- Zero (or near zero) trust in prior solutions
  - Approach this from a clustering perspective
- Easy to generate signatures
  - Non-experts want to play too
- Cross platform
  - <Insert Mac Fanboi Statement>
- Simple to understand and extend
- Fuzzy Matching (similarity)
  - Understand signature overlap
  - Percent of each signature matched

Refresher

Terms, File Structure, 101



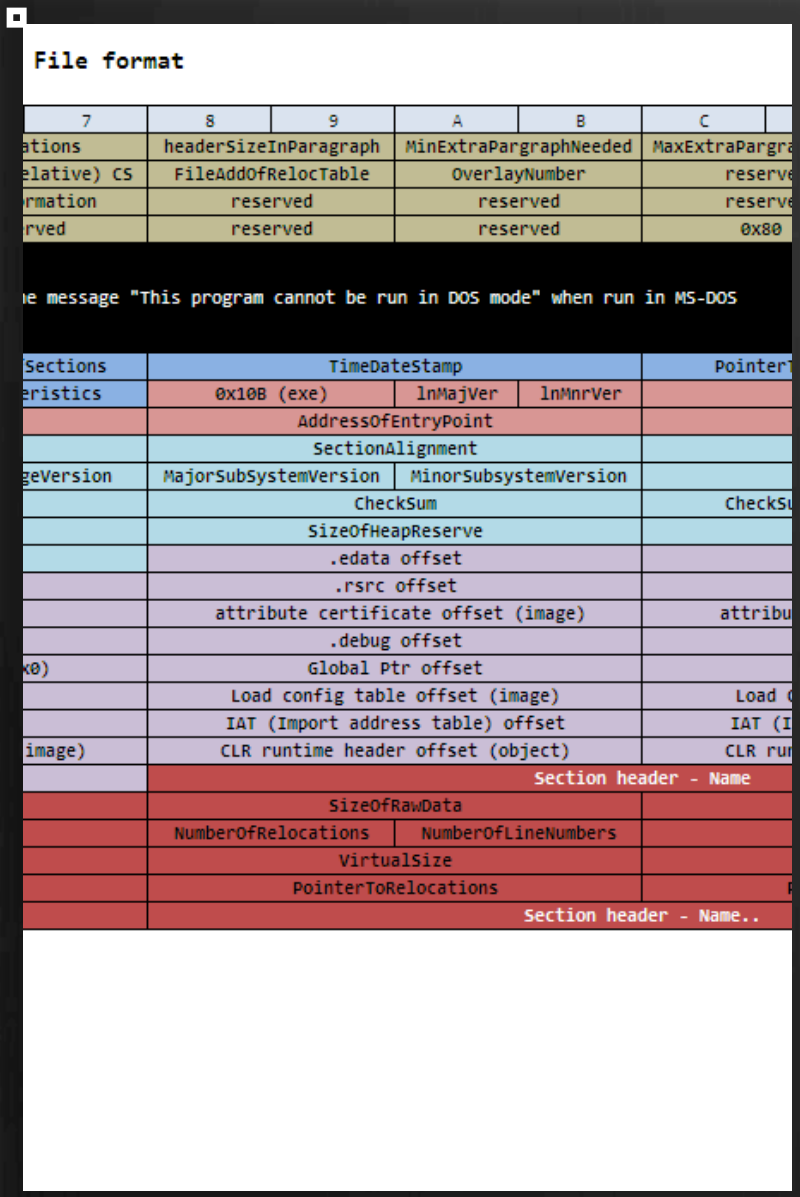
PE Simplified

PE format

PE File format

offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00000000	0x5A4D (MZ)		lastsize		PagesInFile		relocations		headerSizeInParagraph		MinExtraPargraphNeeded		MaxExtraPargraphNeeded		Initial (relative) SS	
0x00000010	Initial (relative) SP		checksum		Initial IP		Initial (relative) CS		FileAddOfRelocTable		OverlayNumber		reserved		reserved	
0x00000020	reserved		reserved		OEMIdentifier		OEMInformation		reserved		reserved		reserved		reserved	
0x00000030	reserved		reserved		reserved		reserved		reserved		reserved		0x80 (offset to PE signature)			
0x00000040	This block contains instructions to display the message "This program cannot be run in DOS mode" when run in MS-DOS															
0x00000050																
0x00000060																
0x00000070																
0x00000080	0x00004550 (PE\0\0 - PE Signature)				Target Machine		NumberOfSections		TimeDateStamp				PointerToSymbolTable (0 for image)			
0x00000090	NumberOfSymbols (0 for image)				SizeOfOptionalHeaders		Characteristics		0x10B (exe)		lnMajVer	lnMnrVer	SizeOfCode			
0x000000A0	SizeOfInitializedData				SizeOfUninitializedData				AddressOfEntryPoint				BaseOfCode			
0x000000B0	BaseOfData				ImageBase				SectionAlignment				FileAlignment			
0x000000C0	MajorOSVersion		MinorOSVersion		MajorImageVersion		MinorImageVersion		MajorSubSystemVersion		MinorSubsystemVersion		Win32VersionValue			
0x000000D0	SizeOfImage				SizeOfHeaders				CheckSum				CheckSum		DllCharacteristics	
0x000000E0	SizeOfStackReserve				SizeOfStackCommit				SizeOfHeapReserve				SizeOfHeapCommit			
0x000000F0	LoaderFlags				NumberOfRVAandSizes				.edata offset				.edata size			
0x00000100	.idata offset				.idata size				.rsrc offset				.rsrc size			
0x00000110	.pdata offset				.pdata size				attribute certificate offset (image)				attribute certificate size (image)			
0x00000120	.reloc offset (image)				.reloc size (image)				.debug offset				.debug size			
0x00000130	Architecture (reserved - 0x0)				Architecture (reserved - 0x0)				Global Ptr offset				must be 0x0			
0x00000140	.tls offset				.tls size				Load config table offset (image)				Load Config table size (image)			
0x00000150	Bound import table offset				Bound import table size				IAT (Import address table) offset				IAT (Import address table) size			
0x00000160	Delay import descriptor offset (image)				Delay import descriptor size (image)				CLR runtime header offset (object)				CLR runtime header size (object)			
0x00000170	Reserved (must be 0x0)				Reserved (must be 0x0)				Section header - Name							
0x00000180	VirtualSize				VirtualAddress				SizeOfRawData				PointerToRawData			
0x00000190	PointerToRelocations				PointerToLineNumbers				NumberOfRelocations		NumberOfLineNumbers		Characteristics			
0x000001A0	Section header - Name								VirtualSize				VirtualAddress			
0x000001B0	SizeOfRawData				PointerToRawData				PointerToRelocations				PointerToLineNumbers			
0x000001C0	NumberOfRelocations		NumberOfLineNumbers		Characteristics				Section header - Name..							

			Size in bytes
	MS-DOS header	File header	64
	PE Signature		4
	COFF header		20
	Standard fields	Optional header	28
	Windows-Specific fields		68
	Data directories		variable
	Section table (each section header is 40 bytes)		variable



File format

7	8	9	A	B	C
ations	headerSizeInParagraph		MinExtraPargraphNeeded		MaxExtraPargra
elative) CS	FileAddOfRelocTable		OverlayNumber		reserve
rmation	reserved		reserved		reserve
rved	reserved		reserved		0x80

e message "This program cannot be run in DOS mode" when run in MS-DOS

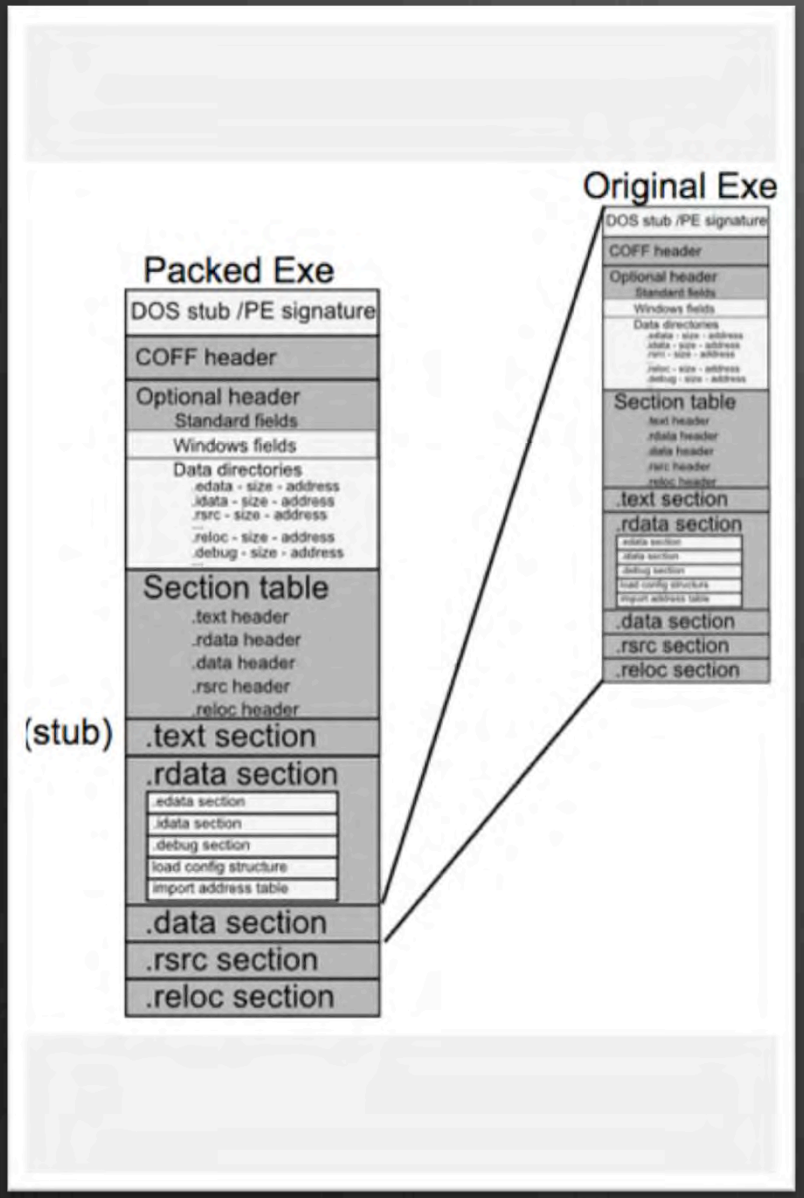
Sections	TimeDateStamp			Pointer
eristics	0x10B (exe)	lnMajVer	lnMnrVer	
	AddressOfEntryPoint			
	SectionAlignment			
geVersion	MajorSubSystemVersion	MinorSubsystemVersion		
	CheckSum			CheckSu
	SizeOfHeapReserve			
	.edata offset			
	.rsrc offset			
	attribute certificate offset (image)			attribu
	.debug offset			
x0)	Global Ptr offset			
	Load config table offset (image)			Load C
	IAT (Import address table) offset			IAT (I
image)	CLR runtime header offset (object)			CLR ru
	Section header - Name			
	SizeOfRawData			
	NumberOfRelocations	NumberOfLineNumbers		
	VirtualSize			
	PointerToRelocations			P
	Section header - Name..			

Things we care about (today)

- LinkerMajorVersion
- LinkerMinorVersion
- NumberOfSections

Tool chain

- Set of tools used to develop software
  - IDE
  - Compiler
  - Linker
  - Etc…
- This talk will touch on compiler/build environment detection
  - Information provided by the linker, etc… will also be used



Packer

- "Program within a program"
- Generally used to compress or obfuscate PE information
  - Evade AV
  - Make static analysis harder

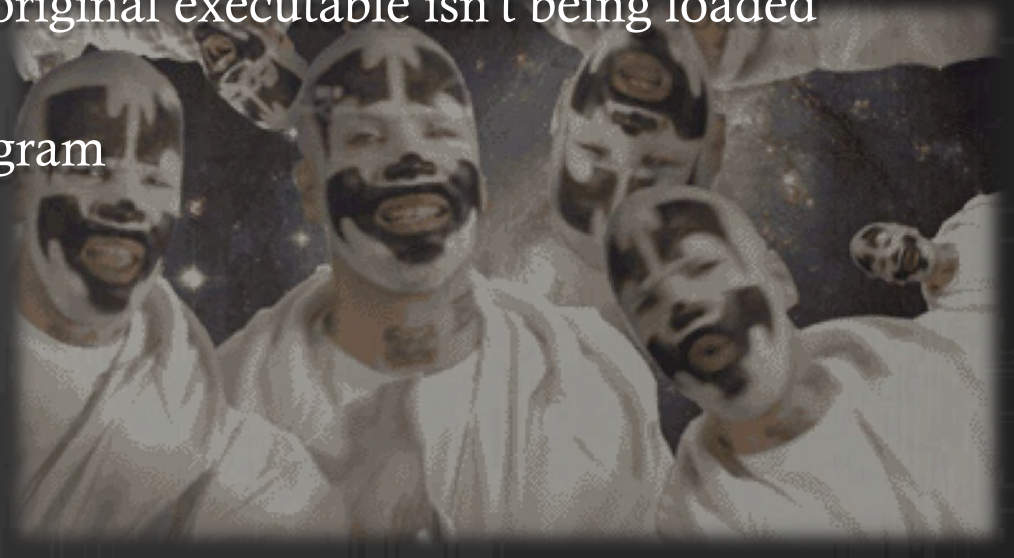
Packers, their parts.

- Packer
  - Compresses/obfuscates the original executable and creates a new executable complete with decompression/deobfuscation code
- Unpacker
  - A.k.a Stub
  - Run when the new executable is executed and is responsible for producing the original executable

Unpackers, how do they work?

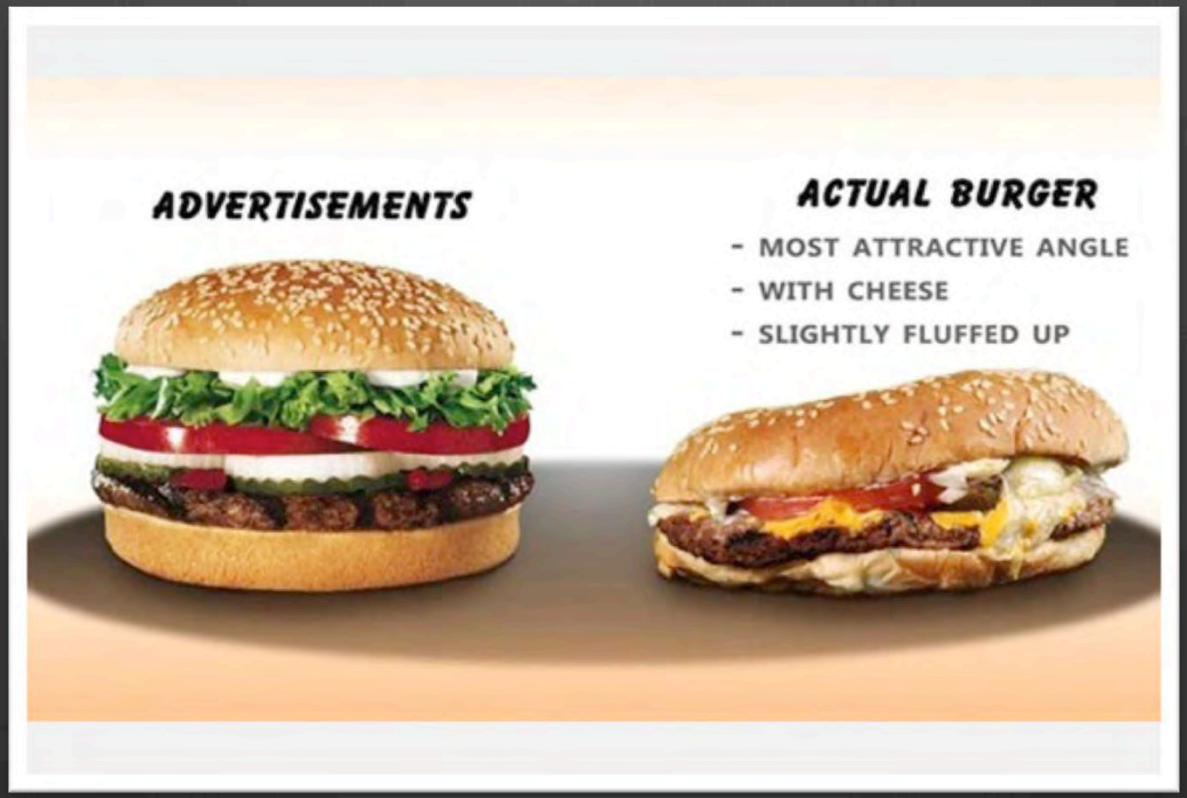
- Take control of AddressOfEntryPoint
- Run the unpacking routine
  - Find the packed data
  - Restore the data contents
  - Perform relocation fixes
  - Resolve imports since the original executable isn't being loaded by the Windows loader
  - Jump into the original program

http://www.stonedcoder.org/~kd/lib/61-267-1-PB.pdf



The popular kids

- PEiD
  - “PEiD detects most common packers, cryptors, and compilers for PE files.”
- YARA
  - “YARA is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples”
- RDG Packer Detector
  - “RDG Packer Detector is a detector of packers, cryptors, Compilers, Packers Scrambler, Joiners, Installers”



Back to Reality (Data)

Now on to the more exciting stuff, even though it's ugly at times.

Data

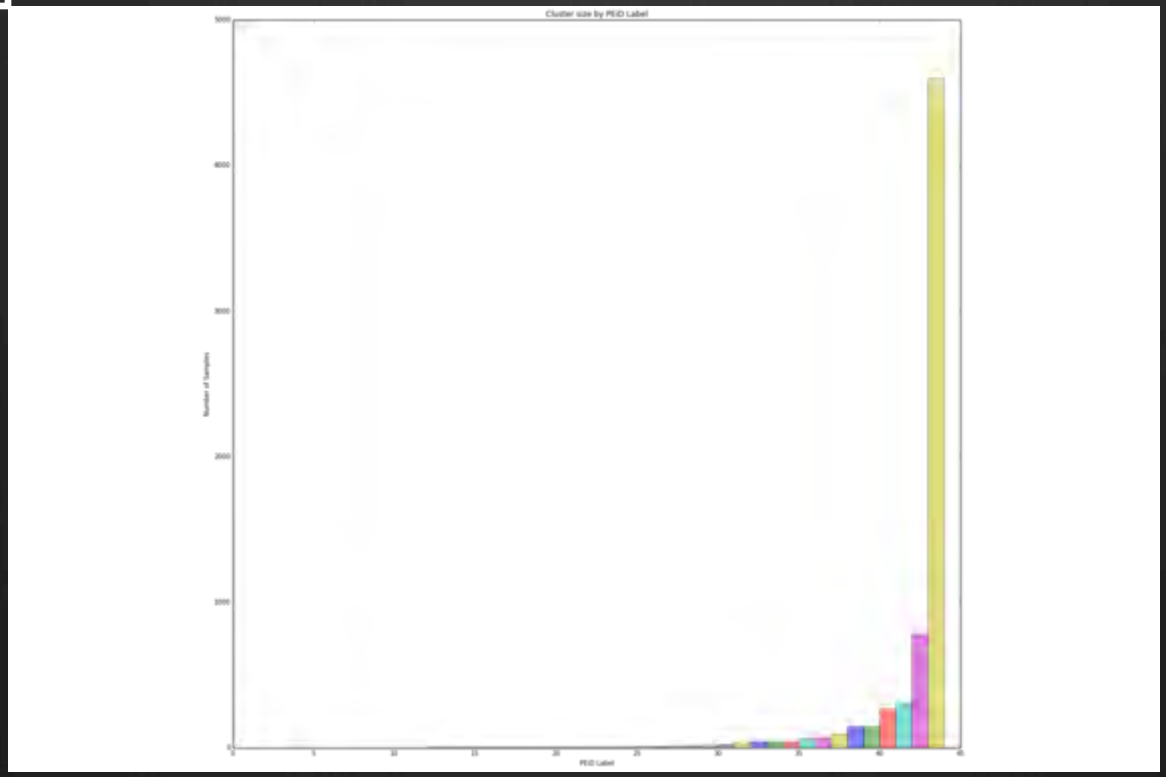
- 3977 PEiD signatures used for testing
- File sets
  - Sony – 9 Executables
  - Chthonic – 11 Executables
  - Backoff – 22 Executables
  - Volatile Ceader – 36 Executables
  - Carbanak – 74 Executables
  - APT1 – 281 Executables
  - ZeuS – 6774 Executables
  - Random – 411340 Executables

Data analysis

- Basic exploration of the ZeuS dataset
- Some of the possible attributes/features we can look at
- Clustering

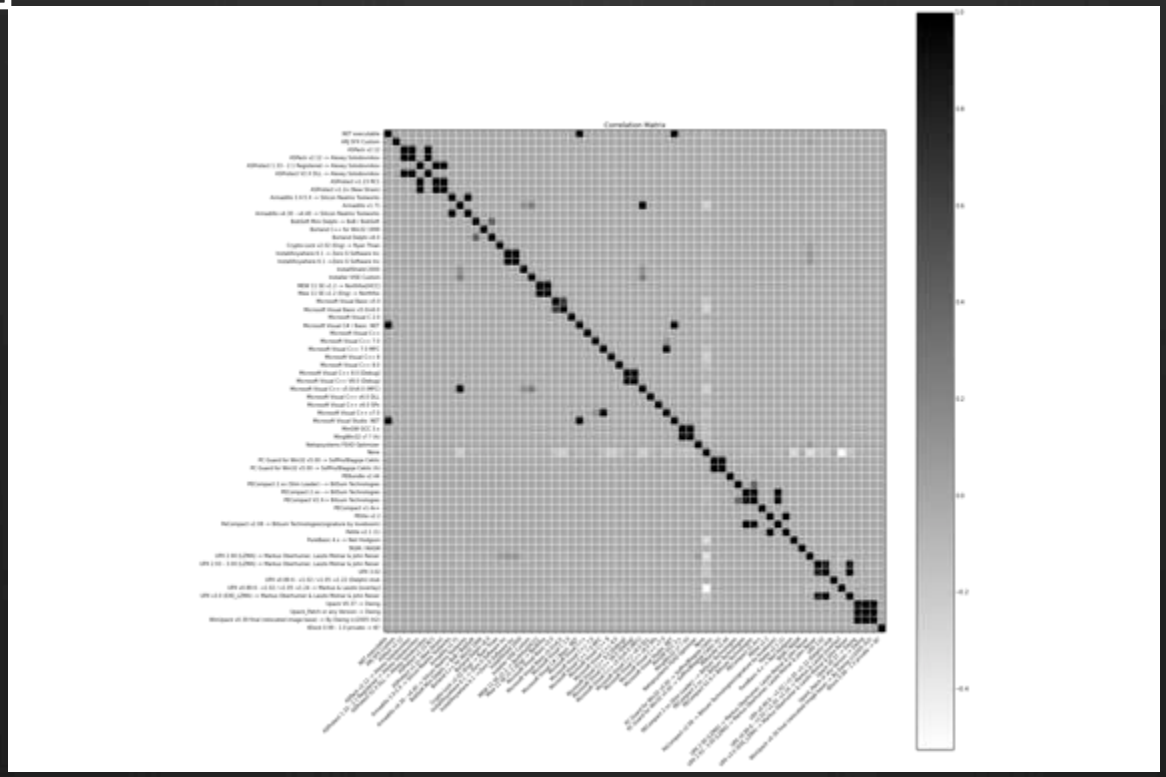
ZeuS + PEiD

PEiD Label	Count
None	4600
UPX v0.89.6 - v1.02 / v1.05 -v1.24 -> Markus & Laszlo [overlay]	781
UPX 2.90 [LZMA] -> Markus Oberhumer, Laszlo Molnar & John Reiser	318
PureBasic 4.x -> Neil Hodgson	267
Microsoft Visual Basic v5.0/v6.0	166
Armadillo v1.71	164
Microsoft Visual C++ 8	148
UPX 2.93 - 3.00 [LZMA] -> Markus Oberhumer, Laszlo Molnar & John Reiser	61
MingWin32 v?.? (h)	45
Microsoft Visual C++ 7.0 MFC	44



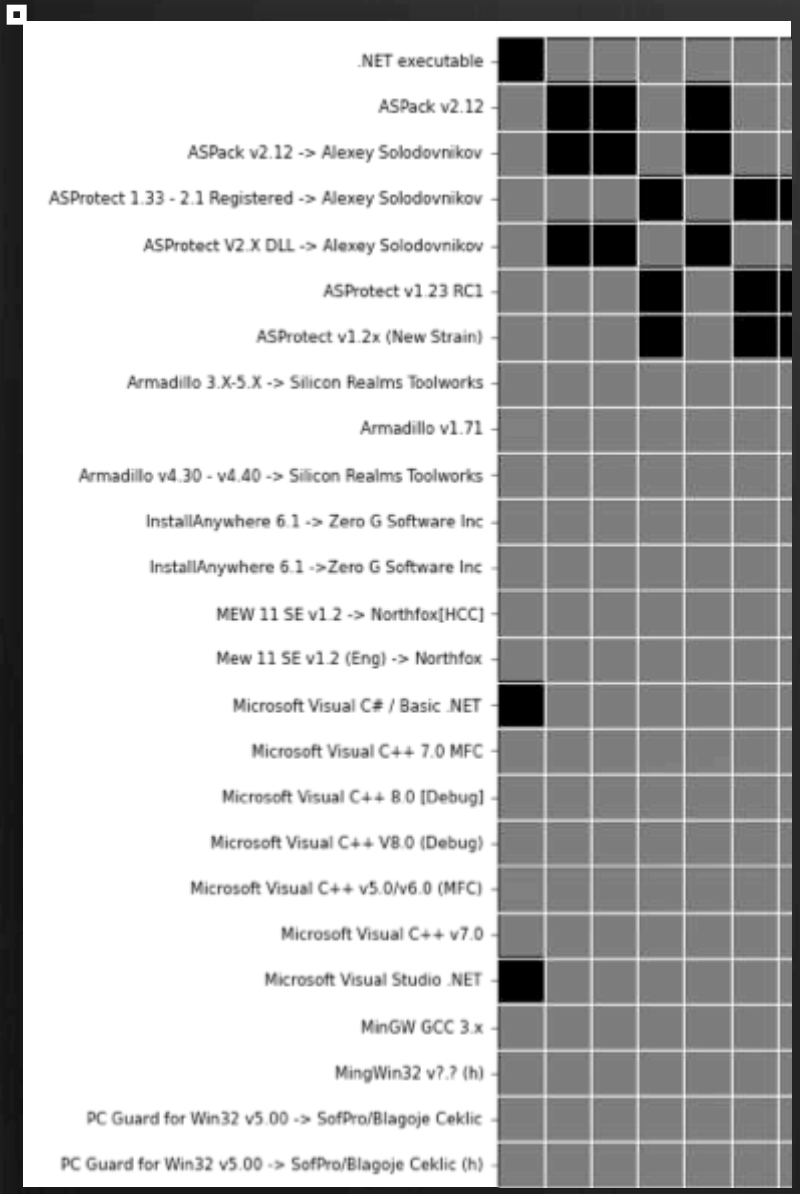
ZeuS + PEiD

Samples per PEiD signature



ZeuS + PEiD

Correlation between the PEiD signatures



Highly correlated

ASPack v2.12	ASPack v2.12 -> Alexey Solodovnikov	1.0
ASPack v2.12	ASProtect V2.X DLL -> Alexey Solodovnikov	1.0

PDB strings

- C:\Users\Samim\Desktop\Stab\stb\Release\stb.pdb
- Y:\DnijJVgd\pitWxRX\ctoerrwx\RtpjVeb.pdb
- H:\RJmq\HYkAuHH\lsvyudBS\yMgpHF\obzwwn.pdb
- C:\answer\record\These\Answer\Dry\Lay\since\Since\mean\Tree\Music.pdb
- X:\DEVELOPMENT\VC++\Cryptor_Evolution_old\release\main.pdb
- c:\temp\debug.pdb
- F:\zmapHjyf\tGQkckQ\UrmgircgraBwwX\nAjaGbB.pdb
- C:\Users\M4x\Documents\Programmieren\PECRYPT\Client\EXECUTABLE\Stub\Release\Stub.pdb

Linker versions

Major.Minor Linker Versions	Count
2.50	2067
10.0	1064
9.0	793
6.0	717
5.0	235
5.12	231
8.0	201
7.10	155
0.0	85
1.1	73

ASM mnemonics

Symbolic name for a machine instruction

- add
- mov
- nop
- xor
- …

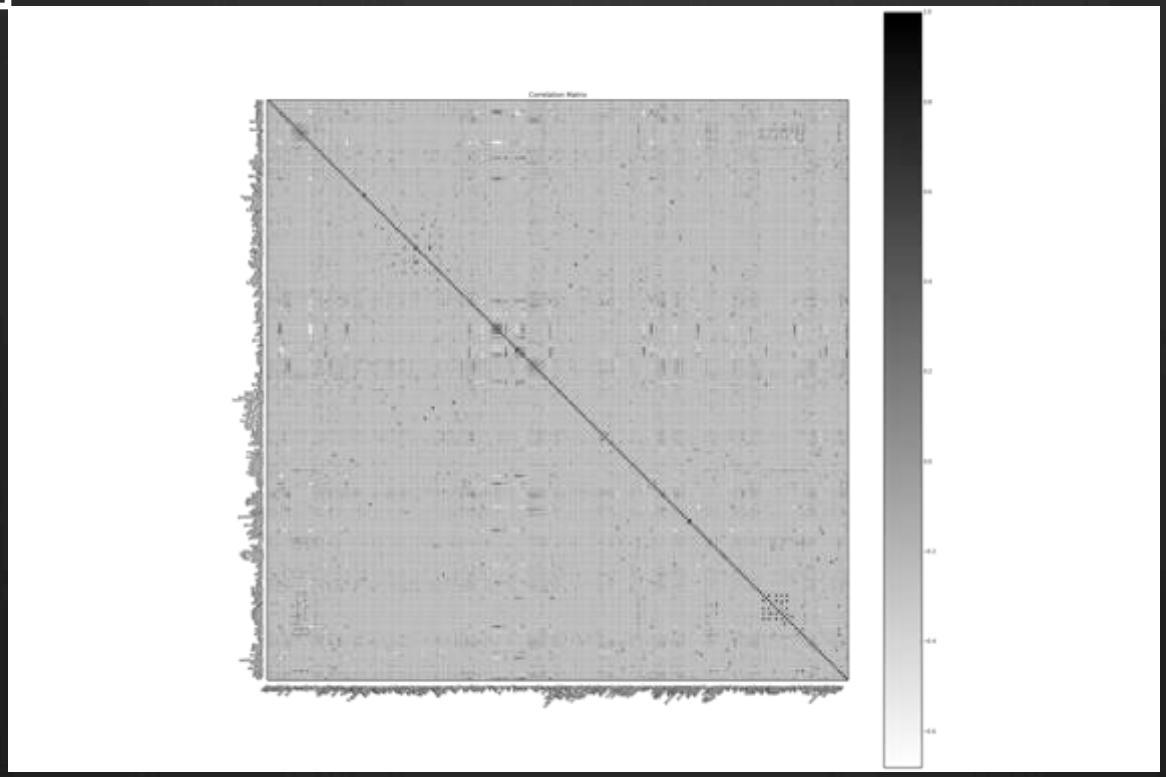
All mnemonics are treated equally from this point forward

Johnny 5 is alive!

Yes. Disassemble.

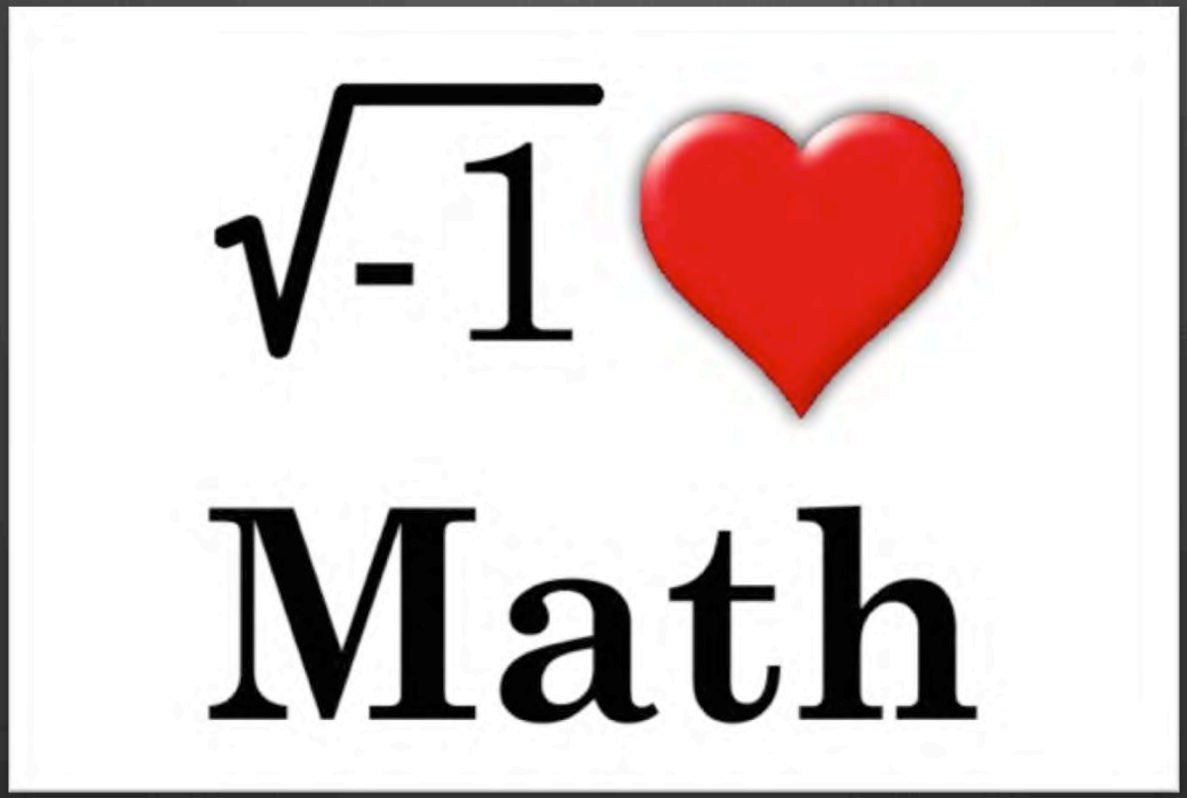
Capstone Engine

- Standardize on one disassembler for consistent results
- Free
- Awesome
- Multi-Language support
- Cross Platform



ZeuS + ASM

Just because we can, doesn't mean we should



It's important

ASM

- Mnemonics describe program behavior
- Mnemonics at AddressOfEntryPoint describe initial program behavior
  - Compiler setup
  - Unpacker stub
- Use this as the basis for a signature

Sets

- Correlation
  - Doesn't take order into account
  - Doesn't really help with distance or similarity
- Jaccard Distance
  - Doesn't take order into account
  - Distance is based on set membership
- Levenshtein Distance
  - Edits determine distance
  - Position is important

Jaccard

['pushal', 'mov', 'lea', 'push', 'jmp', 'nop', 'mov', 'inc', 'mov', 'inc']

['push', 'mov', 'add', 'push', 'mov', 'call', 'mov', 'mov', 'call', 'mov']

Total #of shared elements/Total # of unique elements

[mov push] / [pushal mov lea jump nop inc push add call]

2/8 = .25

Levenshtein

['pushal', 'mov', 'lea', 'push', 'jmp', 'nop', 'mov', 'inc', 'mov', 'inc’]

['push', 'mov', 'add', 'push', 'mov', 'call', 'mov', 'mov', 'call', 'mov']

How many things have to change to make the bottom into the top.

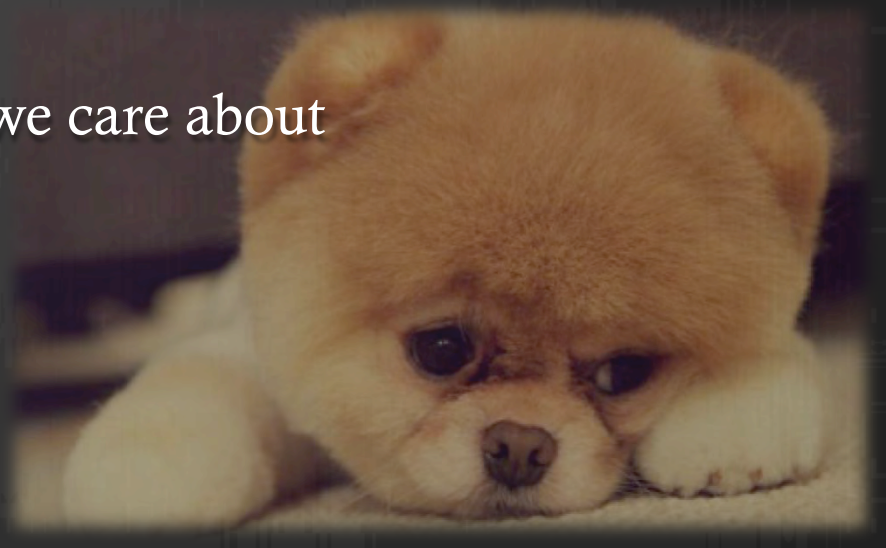
[yes no yes no yes yes no yes yes yes]

yes = 7

Distance = 7

But…

- Code is executed in order
- There might be branches
- Shouldn't the ASM mnemonics to the 'left' be worth more than the ones on the 'right'
- Where's the cutoff
- How many instructions should we care about
- What's the size of the stub



Enter our superhero

(Tapered Levenshtein)

Tapered levenshtein

Position dependent, left edits have a higher weight than right edits

['pushal', 'mov', 'lea', 'push', 'jmp', 'nop', 'mov', 'inc', 'mov', 'inc'

['push', 'mov', 'add', 'push', 'mov', 'call', 'mov', 'mov', 'call', 'mov']

1 - (position/len(set))

1, 0, .8, 0, .6, .5, 0, .3, .2, .1

3.5 (vs. 7 on the non-tapered version)

Now we're ready to science

- PE files
- MajorLinkerVersion
- MinorLinkerVersion
- Assembly mnemonics
- Fancy algorithms

Workflow

1. Gather samples
2. Static analysis
    1. PEiD
    2. Disassemble
    3. Header features
3. Cluster
4. Closeness according to distance metric (> 90% similar)
    1. Use banded minhash for < $O(n^2)$ comparisons
5. Analyze based on cluster groups using 30 mnemonics
    1. 30 mnemonic chain length is based on prior research and exploration
6. Create Signatures

Demo

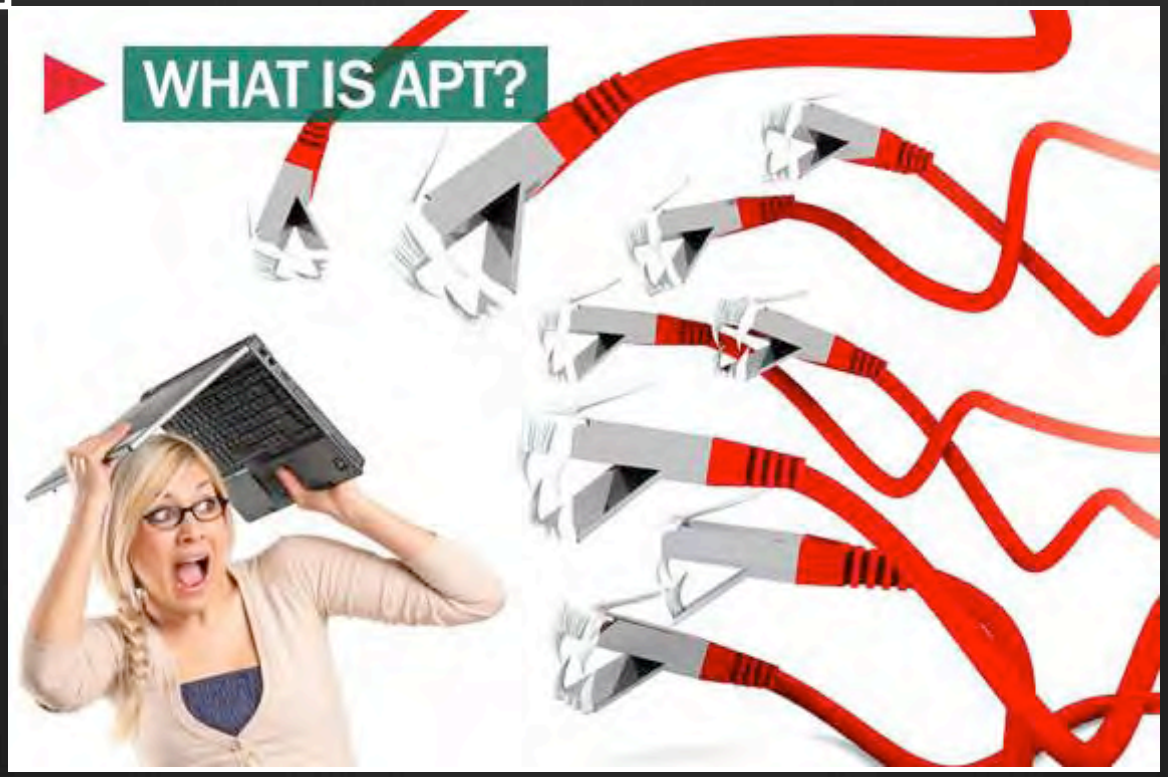
Please accept our chicken sacrifice, Demo Gods.

```
mac:packerid          $ python ./mmpes.py -s ./test.sig -v -t 0.0 ~/data/APT1/VirusShare_01e0dc079d4e33d8edd050c4900818da
[*] Processing: /Users/          /data/APT1/VirusShare_01e0dc079d4e33d8edd050c4900818da
[TEST] (Edits: 4.66666666667 | Similarity: 0.844) (Minor Linker Version Match: True | Major Linker Version Match: True | Number Of Sections Match: False)
['push', 'mov', 'push', 'push', 'push', 'mov', 'push', 'mov', 'sub', 'push', 'push', 'push', 'mov', 'and', 'push', 'call', 'pop', 'or', 'or', 'call', 'mov', 'mov', 'call', 'mov', 'mov', 'mov', 'mov', 'mov', 'call', 'cmp']
['push', 'mov', 'push', 'push', 'push', 'mov', 'push', 'mov', 'sub', 'push', 'push', 'push', 'mov', 'call', 'xor', 'mov', 'mov', 'mov', 'and', 'mov', 'shl', 'add', 'mov', 'shr', 'mov', 'push', 'call', 'pop', 'test', 'jne']

mac:packerid          $ python ./mmpes.py -s ./test.sig -v -t 0.0 ~/data/APT1/VirusShare_002325a0a67fded0381b5648d7fe9b8e
[*] Processing: /Users/          /data/APT1/VirusShare_002325a0a67fded0381b5648d7fe9b8e
[TEST] (Edits: 0.0 | Similarity: 1.000) (Minor Linker Version Match: True | Major Linker Version Match: True | Number Of Sections Match: True)
['push', 'mov', 'push', 'push', 'push', 'mov', 'push', 'mov', 'sub', 'push', 'push', 'push', 'mov', 'and', 'push', 'call', 'pop', 'or', 'or', 'call', 'mov', 'mov', 'call', 'mov', 'mov', 'mov', 'mov', 'mov', 'call', 'cmp']
['push', 'mov', 'push', 'push', 'push', 'mov', 'push', 'mov', 'sub', 'push', 'push', 'push', 'mov', 'and', 'push', 'call', 'pop', 'or', 'or', 'call', 'mov', 'mov', 'call', 'mov', 'mov', 'mov', 'mov', 'mov', 'call', 'cmp']
```

```
XXX-mac:packerid XXXX$ python ./mmpes.py -s ./test.sig -v -t 0.0 ~/data/APT1/
VirusShare_01e0dc079d4e33d8edd050c4900818da
[*] Processing: /Users/XXX/data/APT1/VirusShare_01e0dc079d4e33d8edd050c4900818da
[TEST] (Edits: 4.66666666667 | Similarity: 0.844) (Minor Linker Version Match: True | Major Linker
Version Match: True | Number Of Sections Match: False)
['push', 'mov', 'push', 'push', 'push', 'mov', 'push', 'mov', 'sub', 'push', 'push', 'push',
'mov', 'and', 'push', 'call', 'pop', 'or', 'or', 'call', 'mov', 'mov', 'call', 'mov', 'mov',
'mov', 'mov', 'mov', 'call', 'cmp']
['push', 'mov', 'push', 'push', 'push', 'mov', 'push', 'mov', 'sub', 'push', 'push', 'push',
'mov', 'call', 'xor', 'mov', 'mov', 'mov', 'and', 'mov', 'shl', 'add', 'mov', 'shr', 'mov',
'push', 'call', 'pop', 'test', 'jne']

XXX-mac:packerid XXX$ python ./mmpes.py -s ./test.sig -v -t 0.0 ~/data/APT1/
VirusShare_002325a0a67fded0381b5648d7fe9b8e
[*] Processing: /Users/XXX/data/APT1/VirusShare_002325a0a67fded0381b5648d7fe9b8e
[TEST] (Edits: 0.0 | Similarity: 1.000) (Minor Linker Version Match: True | Major Linker Version
Match: True | Number Of Sections Match: True)
['push', 'mov', 'push', 'push', 'push', 'mov', 'push', 'mov', 'sub', 'push', 'push', 'push',
'mov', 'and', 'push', 'call', 'pop', 'or', 'or', 'call', 'mov', 'mov', 'call', 'mov', 'mov',
'mov', 'mov', 'mov', 'call', 'cmp']
['push', 'mov', 'push', 'push', 'push', 'mov', 'push', 'mov', 'sub', 'push', 'push', 'push',
'mov', 'and', 'push', 'call', 'pop', 'or', 'or', 'call', 'mov', 'mov', 'call', 'mov', 'mov',
'mov', 'mov', 'mov', 'call', 'cmp']
```

Signatures

- [Microsoft Visual Basic v5.0]
  - mnemonics = push,call,add,add,add,xor,add,inc,add,add,add,add,adc,dec,mov,adc,add,add,add,add,or,imul,push,and,and
- File
  - mnemonics = push,call,add,add,add,xor,add,inc,add,add,add,add,jmp,dec,mov,xor,add,add,add,add,add,add,dec,jnp,add
- Results
  - [Microsoft Visual Basic v5.0] (Edits: 2.93333333333 | Similarity: 0.902)

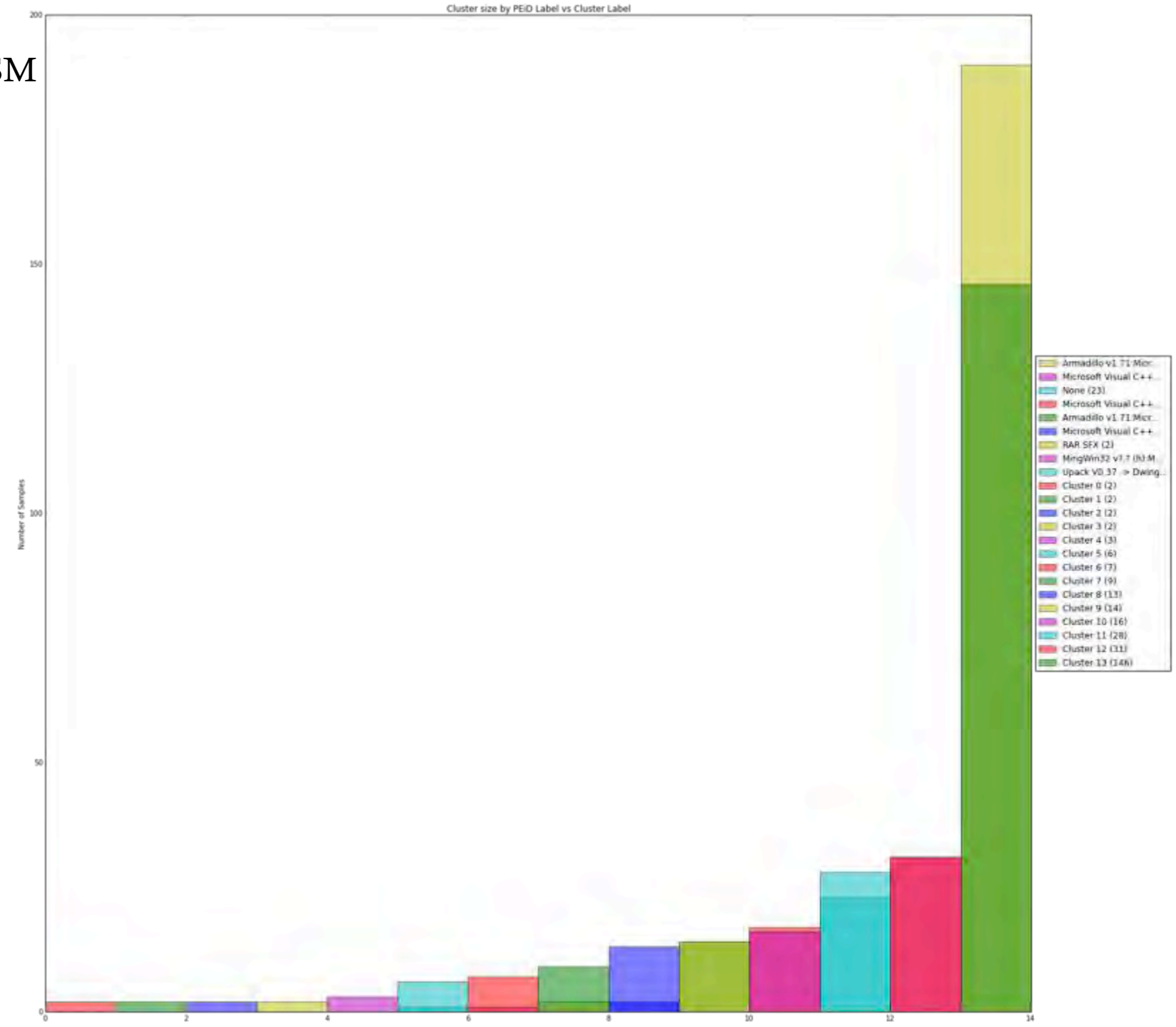


https://blog.kaspersky.com/files/2013/06/apt_title.jpg

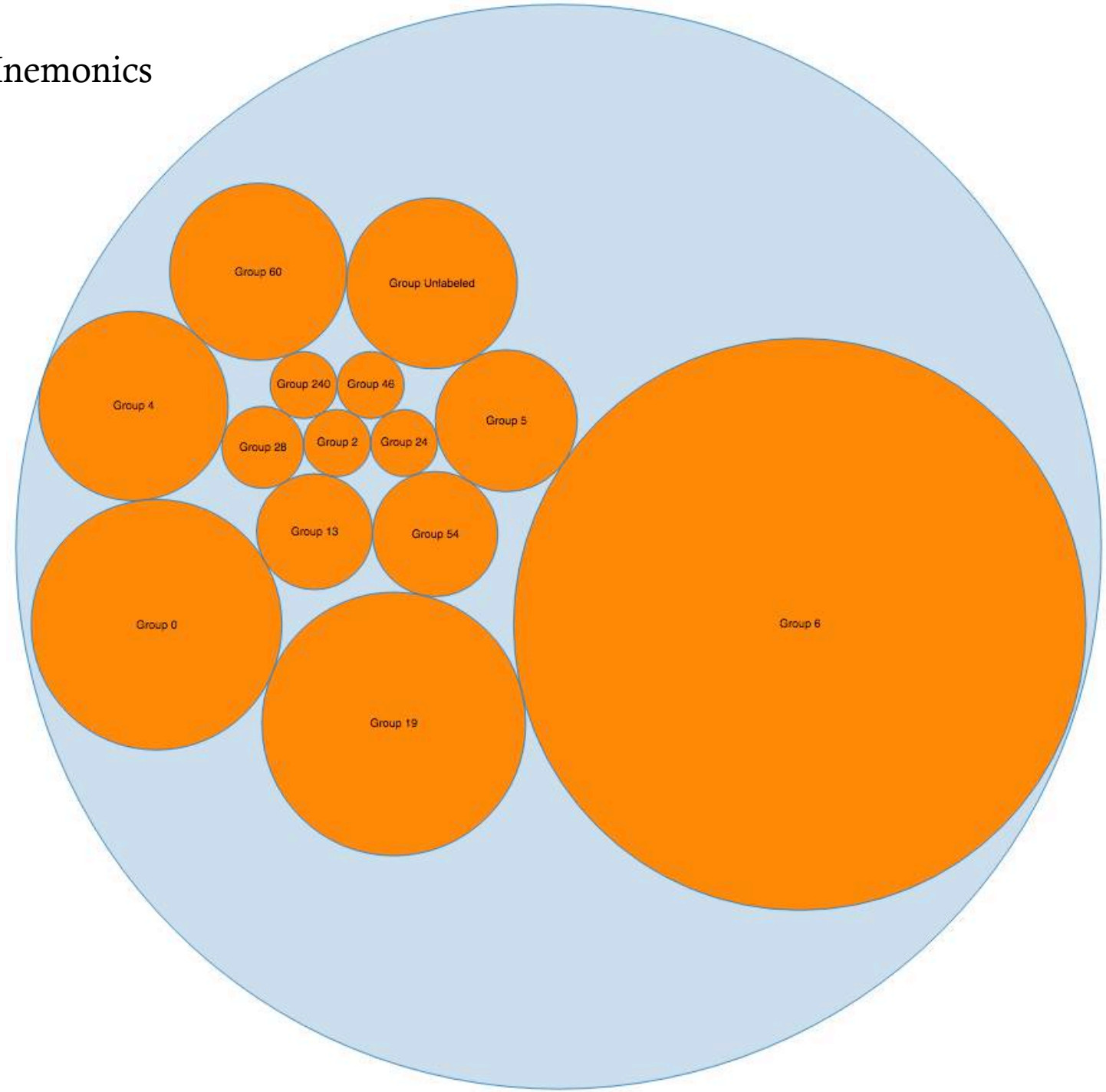
APT1

Because first APT is best APT

APT 1
PEiD vs. ASM

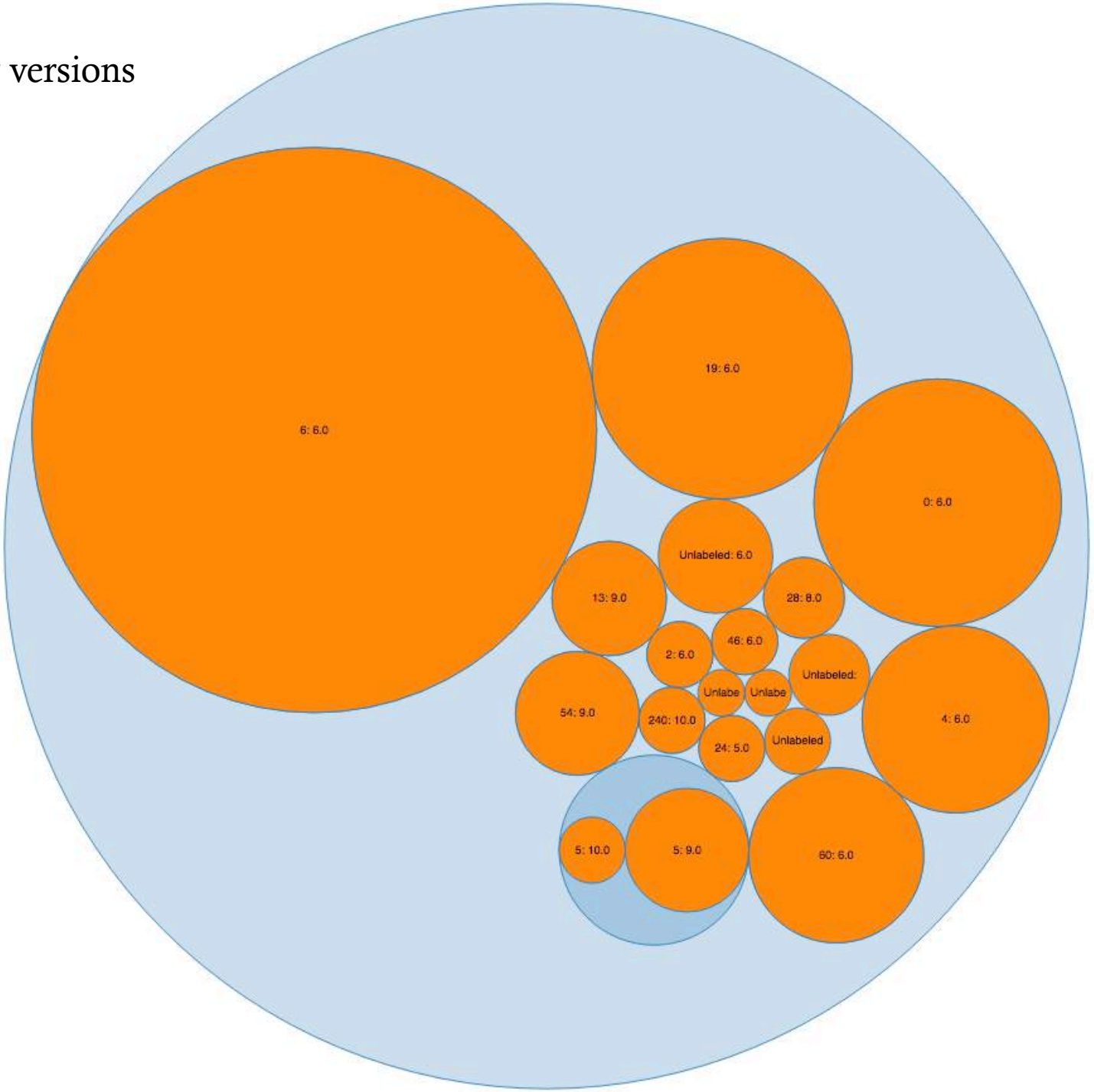


APT 1
Cluster on 30 ASM Mnemonics

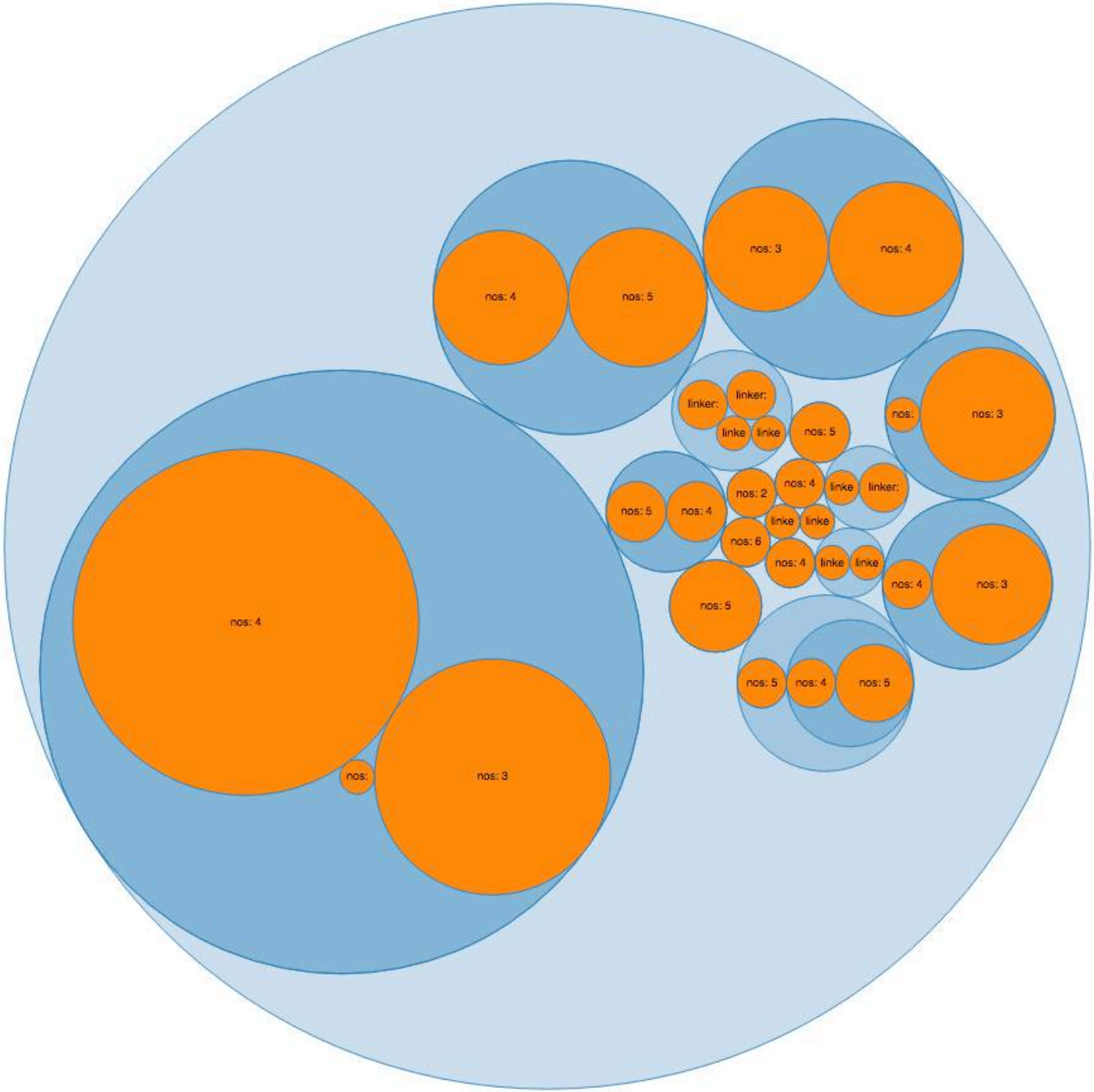


APT 1
Sub-clustered on number of sections

APT 1
Sub-clustered on linker versions



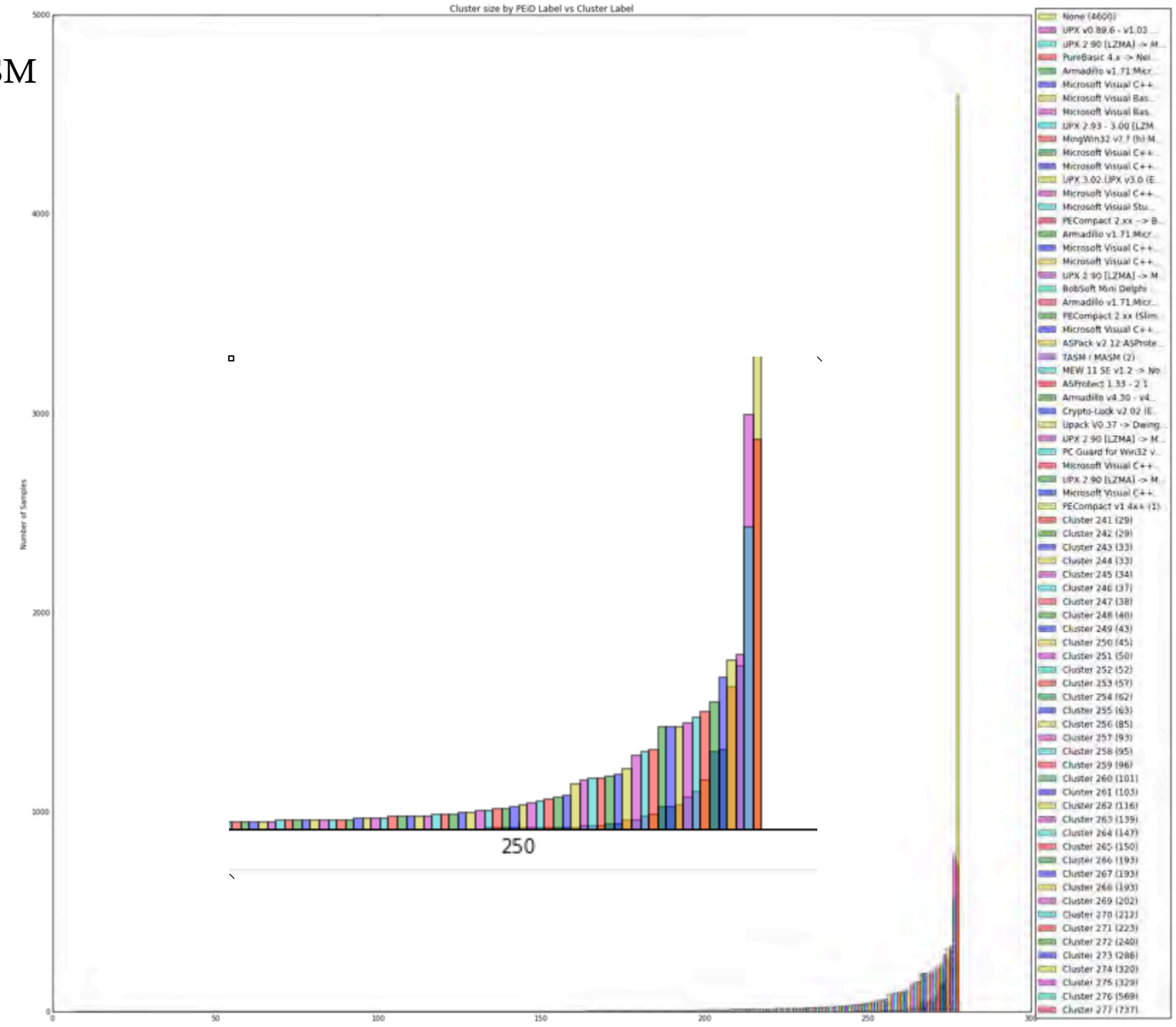
APT 1
Sub-clustered on both



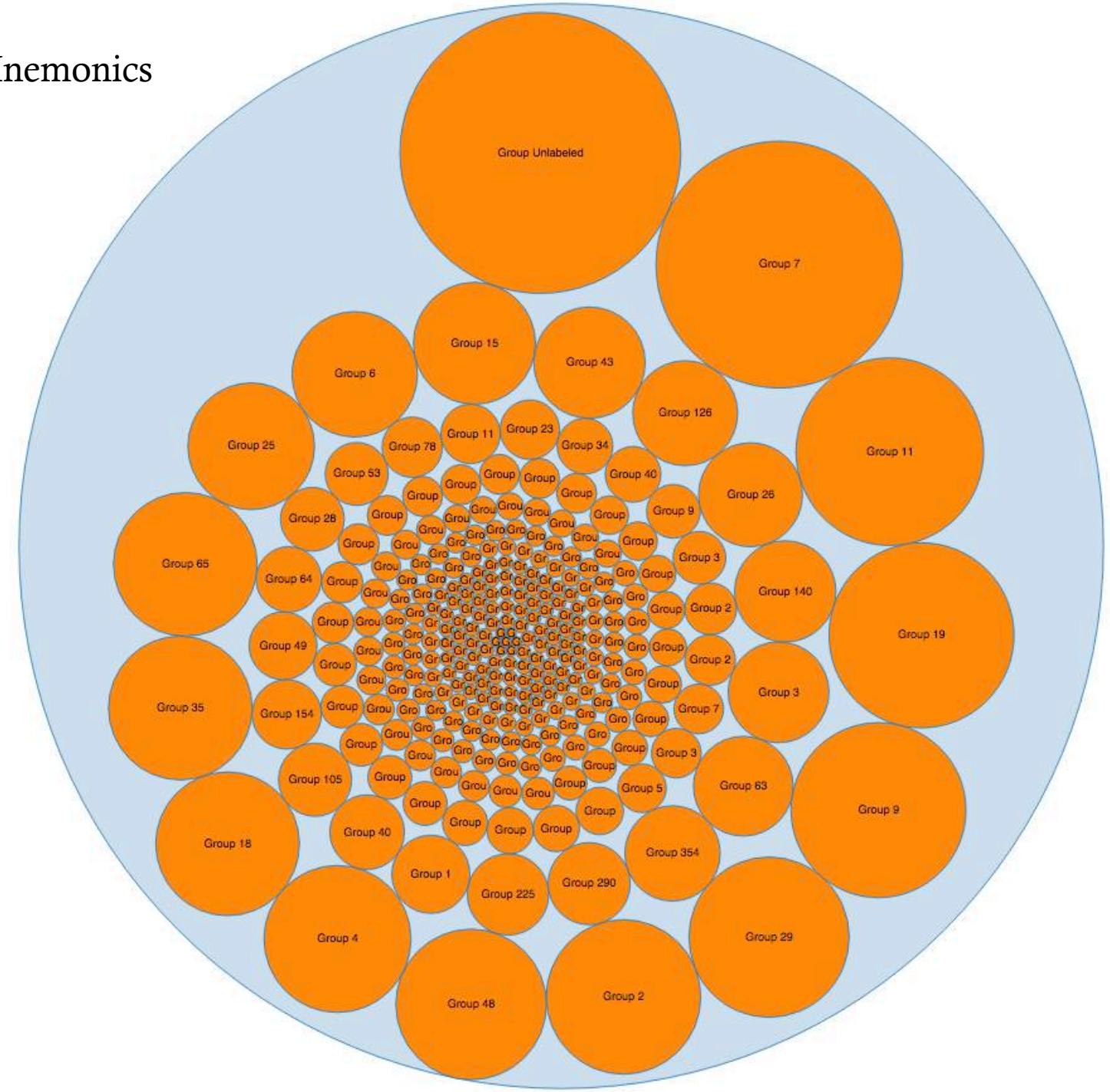
ZeuS

Malware or Greek God?

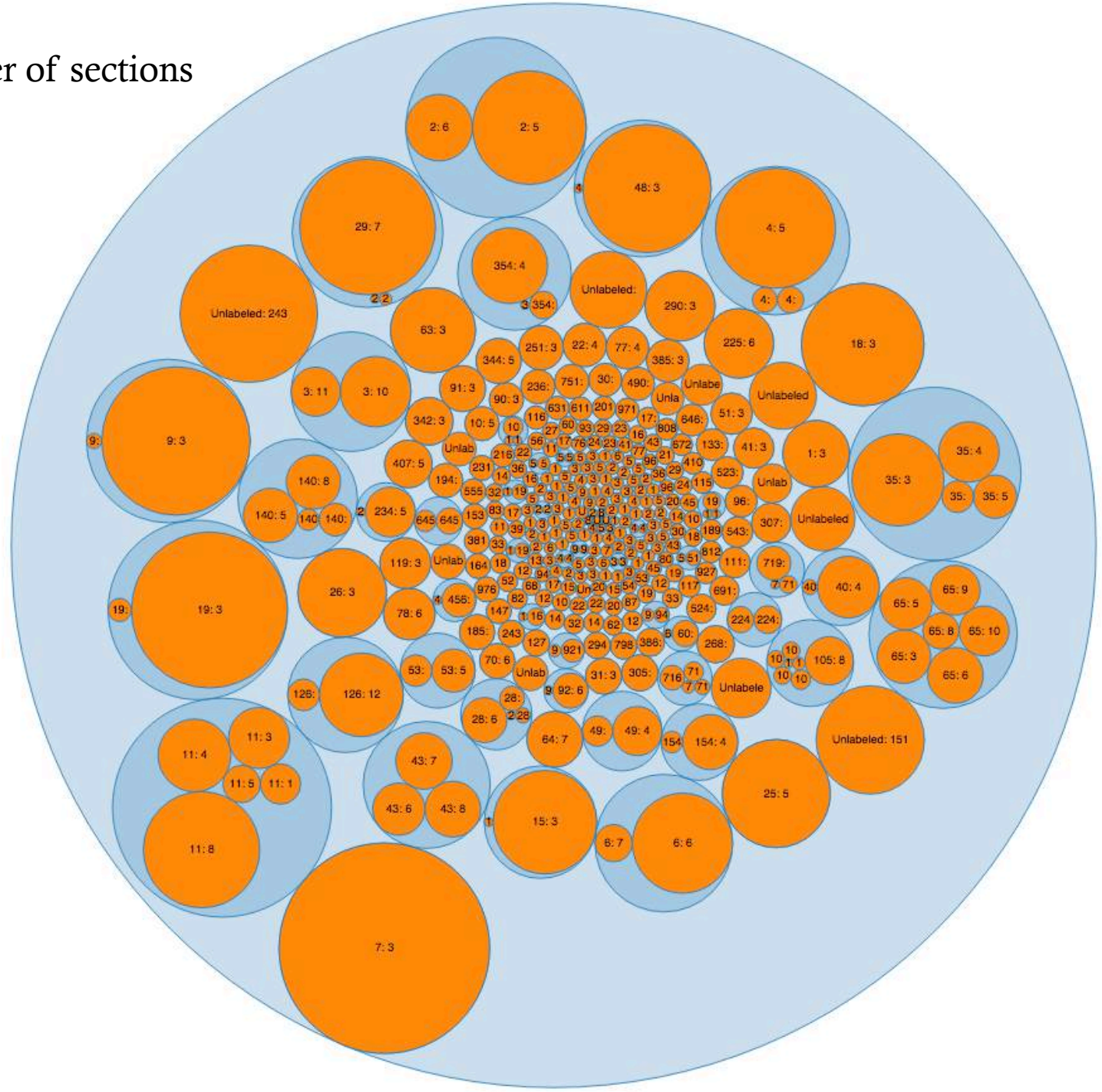
ZeuS
PEiD vs. ASM



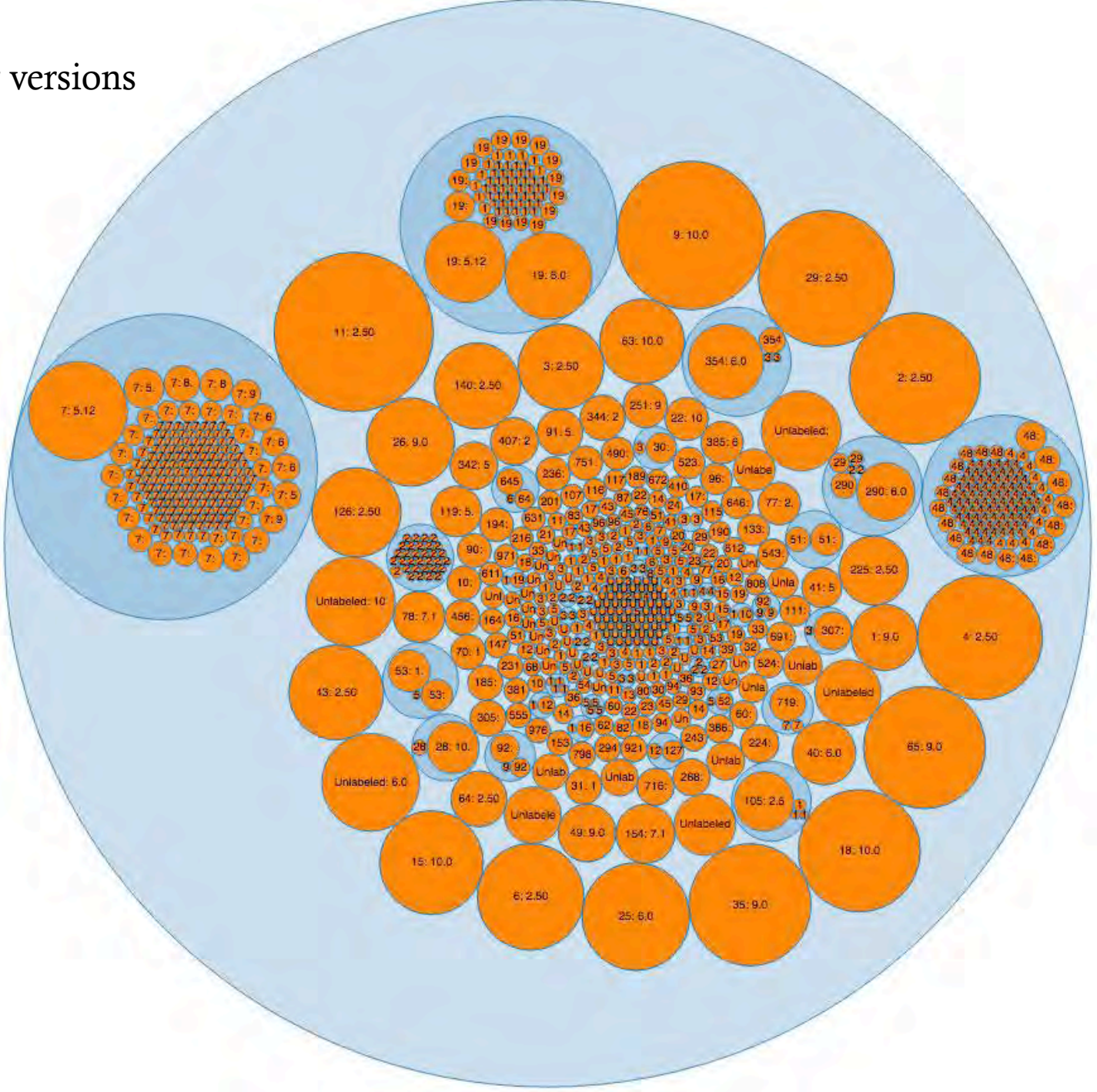
ZeuS
Cluster on 30 ASM Mnemonics



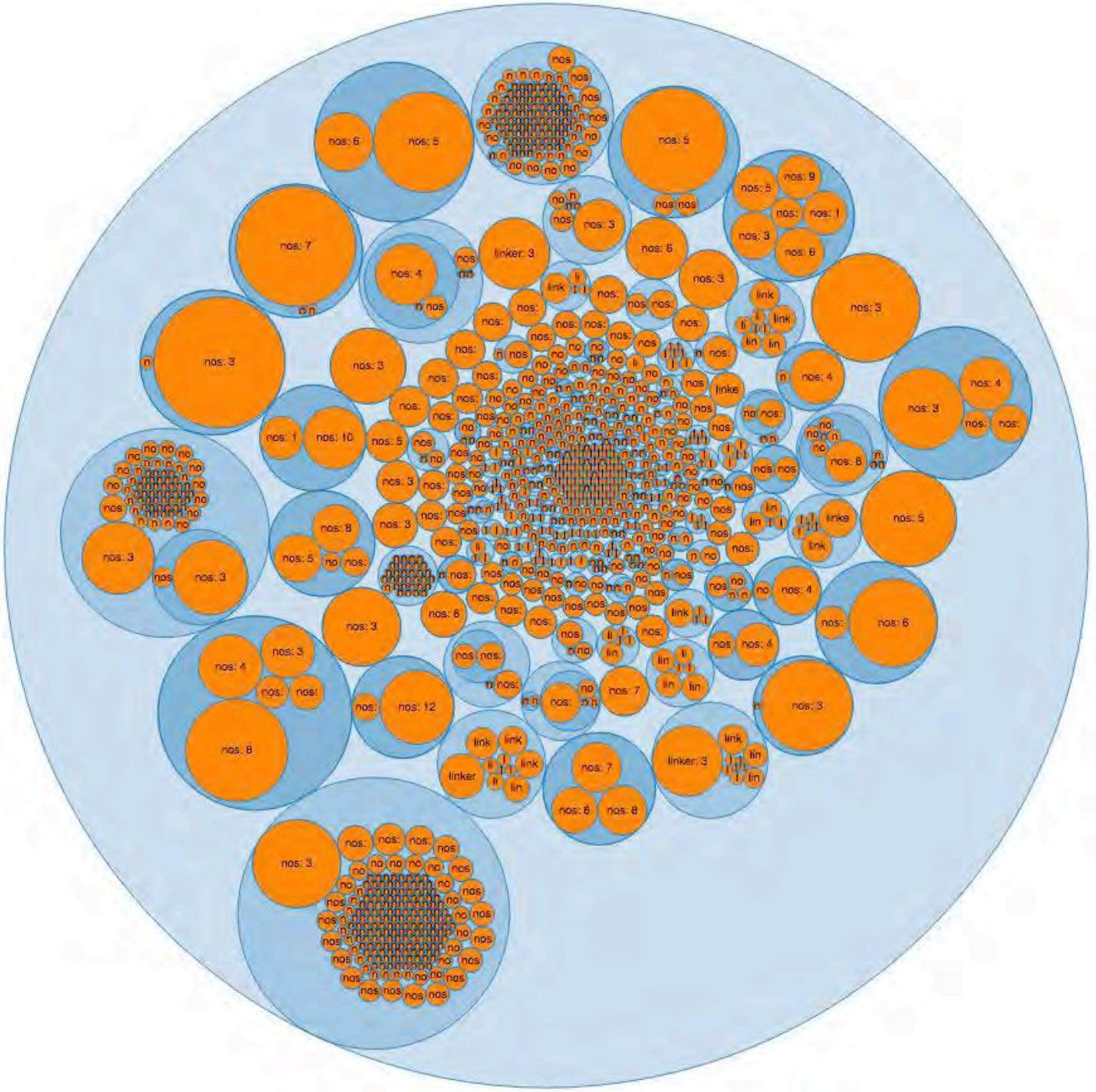
ZeuS
Sub-clusters on number of sections

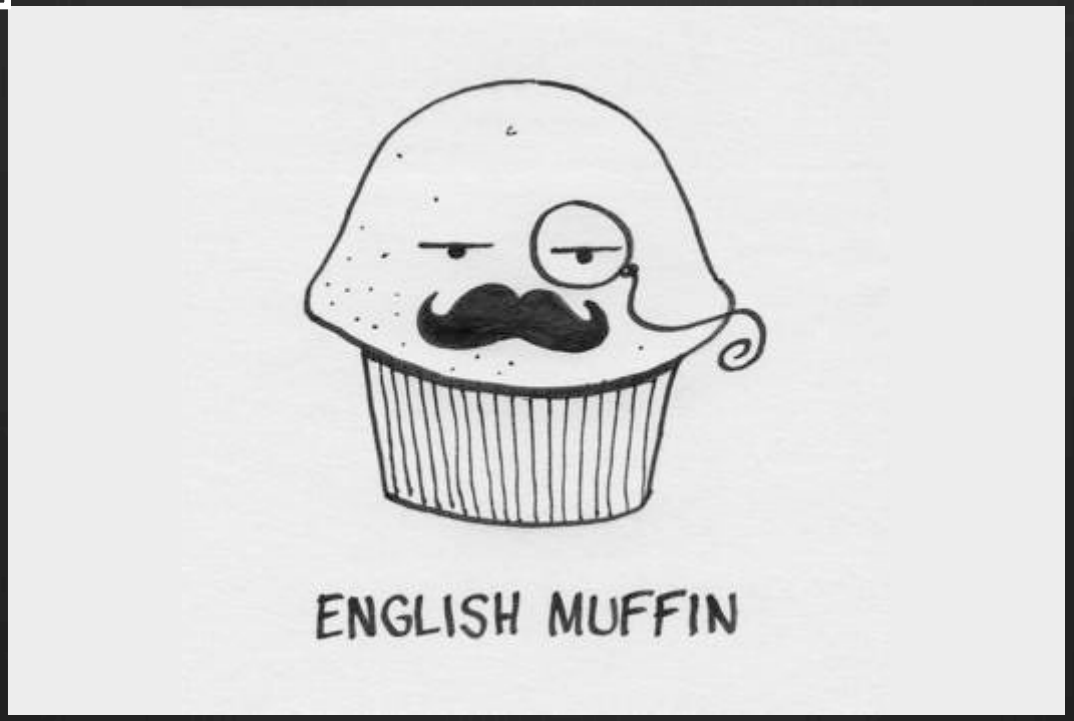


ZeuS
Sub-clustered on linker versions



ZeuS
Sub-clustered on both

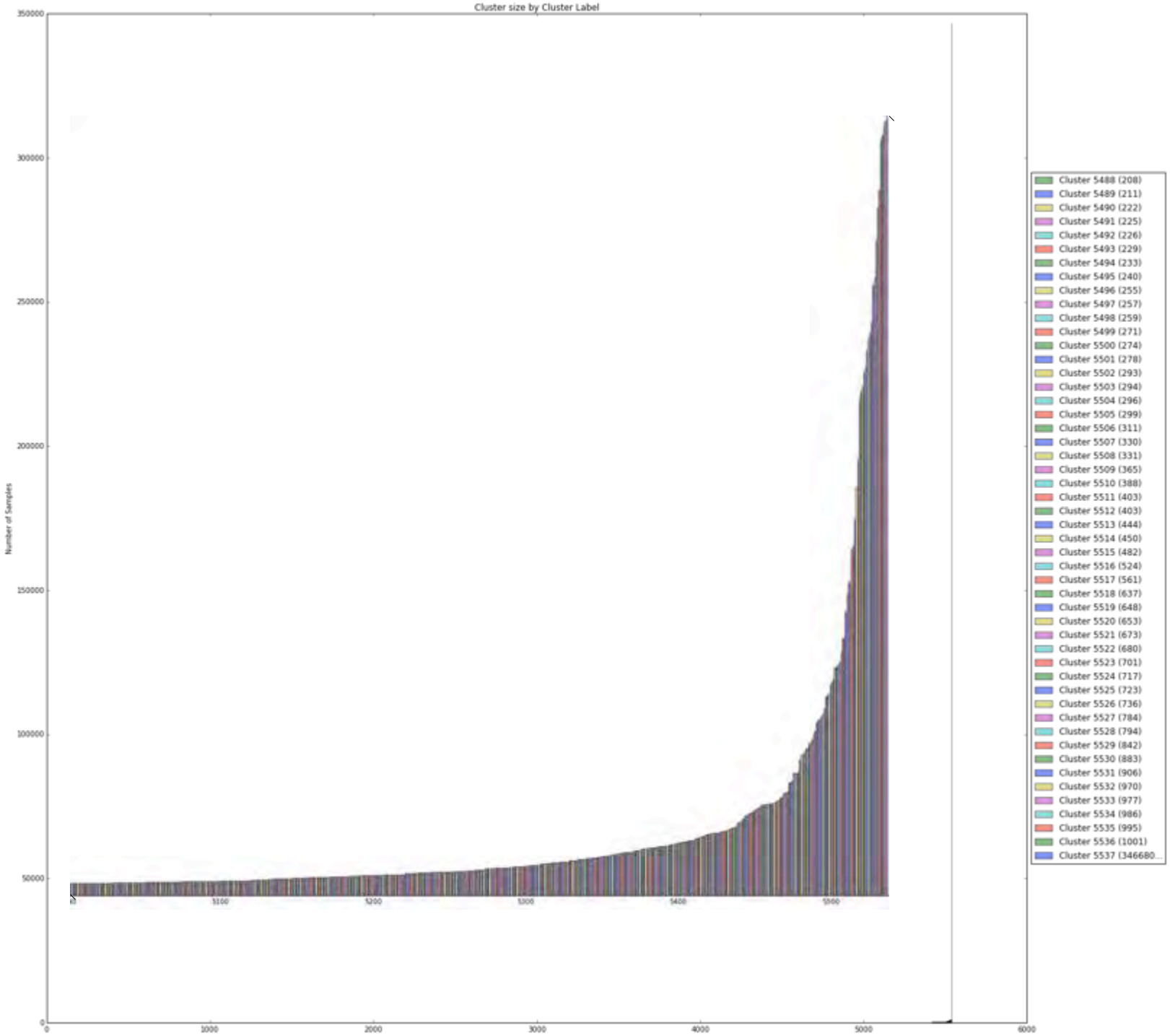




Random files

Are random

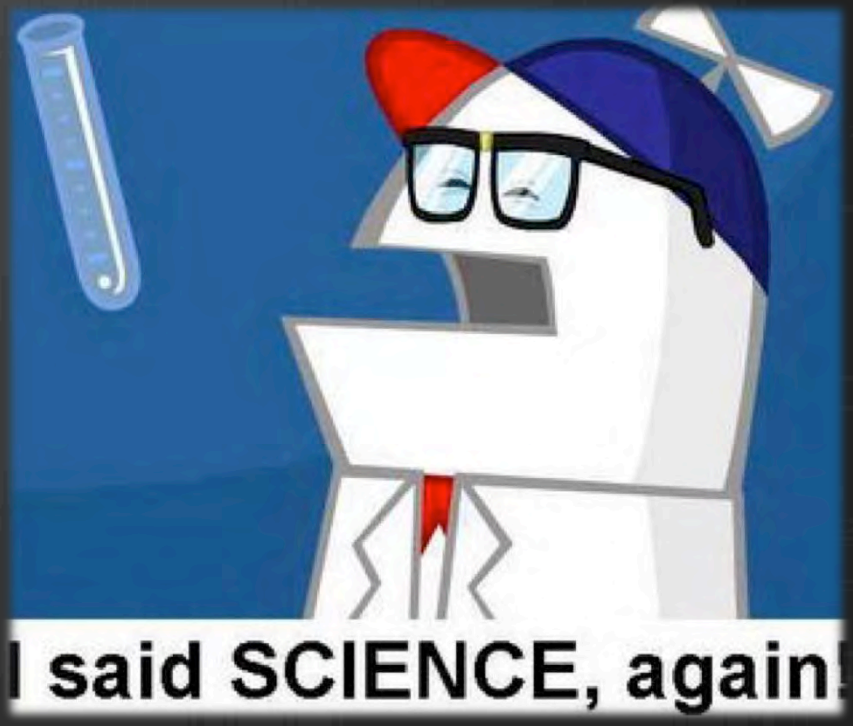
Random
ASM


Cluster size by Cluster Label
Number of Samples
0
50000
100000
150000
200000
250000
300000
350000
0
1000
2000
3000
4000
5000
6000
5100
5200
5300
5400
5500
Cluster 5488 (208)
Cluster 5489 (211)
Cluster 5490 (222)
Cluster 5491 (225)
Cluster 5492 (226)
Cluster 5493 (229)
Cluster 5494 (233)
Cluster 5495 (240)
Cluster 5496 (255)
Cluster 5497 (257)
Cluster 5498 (259)
Cluster 5499 (271)
Cluster 5500 (274)
Cluster 5501 (278)
Cluster 5502 (293)
Cluster 5503 (294)
Cluster 5504 (296)
Cluster 5505 (299)
Cluster 5506 (311)
Cluster 5507 (330)
Cluster 5508 (331)
Cluster 5509 (365)
Cluster 5510 (388)
Cluster 5511 (403)
Cluster 5512 (403)
Cluster 5513 (444)
Cluster 5514 (450)
Cluster 5515 (482)
Cluster 5516 (524)
Cluster 5517 (561)
Cluster 5518 (637)
Cluster 5519 (648)
Cluster 5520 (653)
Cluster 5521 (673)
Cluster 5522 (680)
Cluster 5523 (701)
Cluster 5524 (717)
Cluster 5525 (723)
Cluster 5526 (736)
Cluster 5527 (784)
Cluster 5528 (794)
Cluster 5529 (842)
Cluster 5530 (883)
Cluster 5531 (906)
Cluster 5532 (970)
Cluster 5533 (977)
Cluster 5534 (986)
Cluster 5535 (995)
Cluster 5536 (1001)
Cluster 5537 (346680...

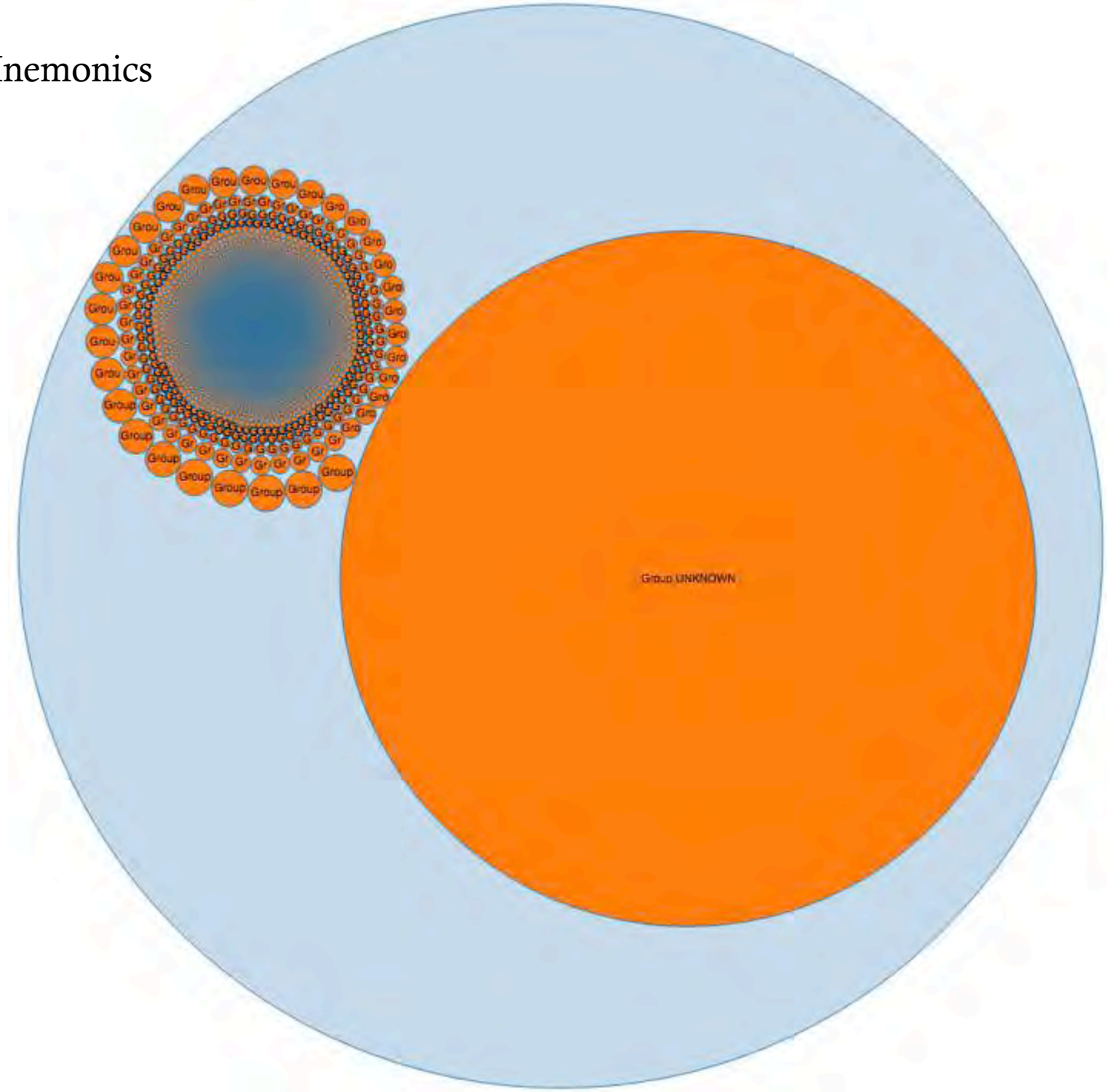
Fun fact

- 346680 out of 411340 files don't meet the similarity requirement*
  - No 2 files are at least 90% similar

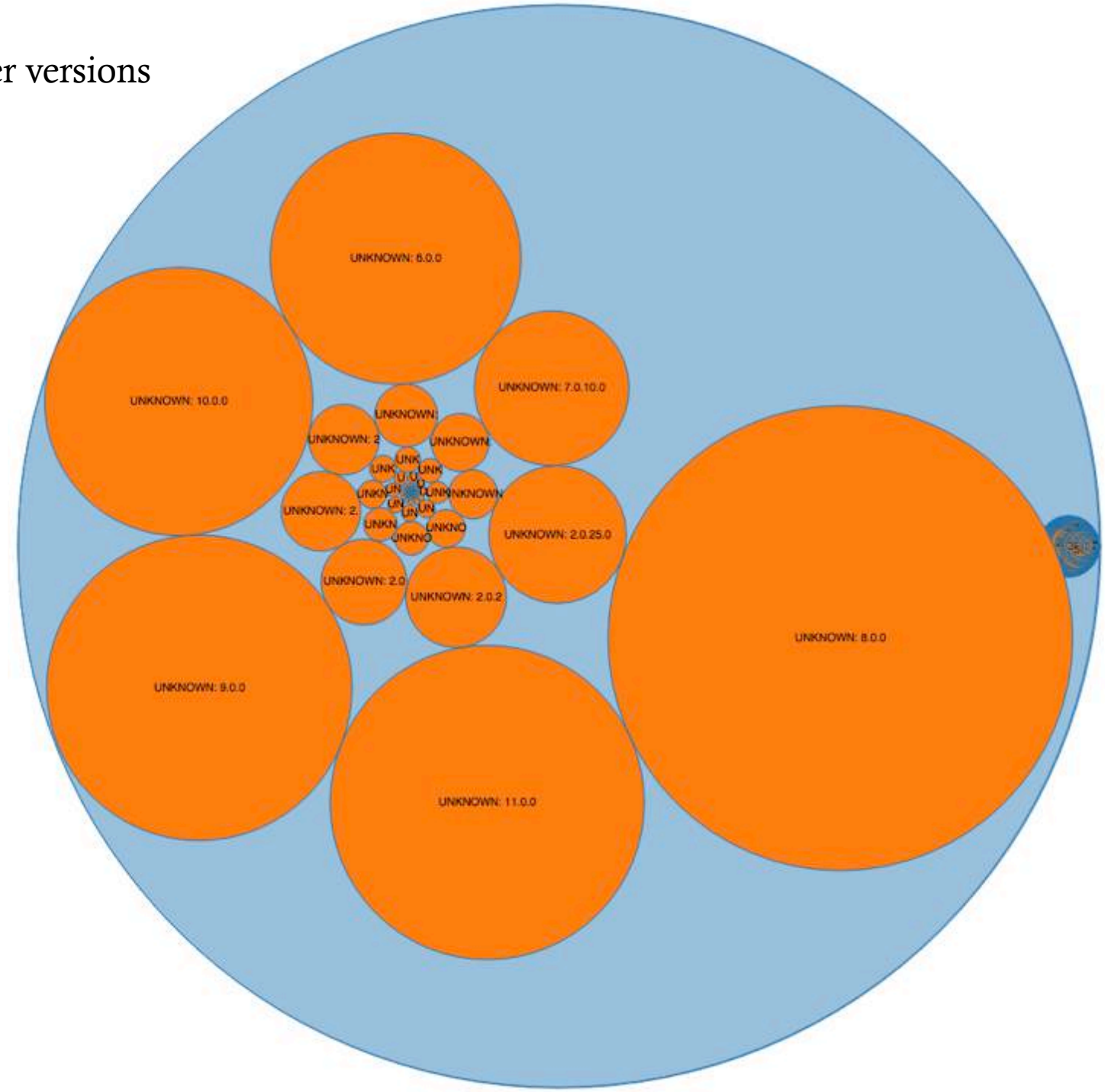
*Actual number may be lower



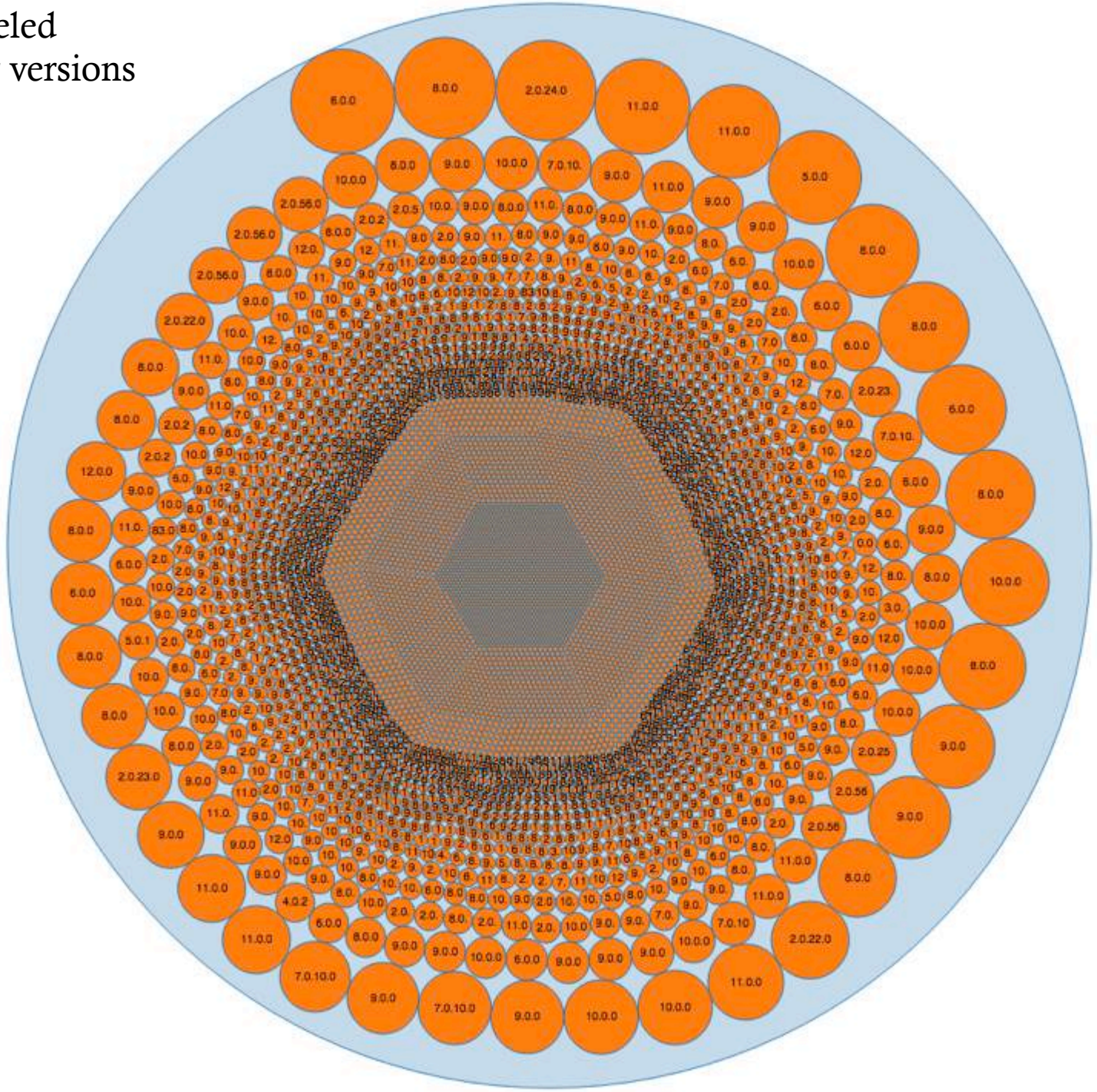
Random
Cluster on 30 ASM Mnemonics



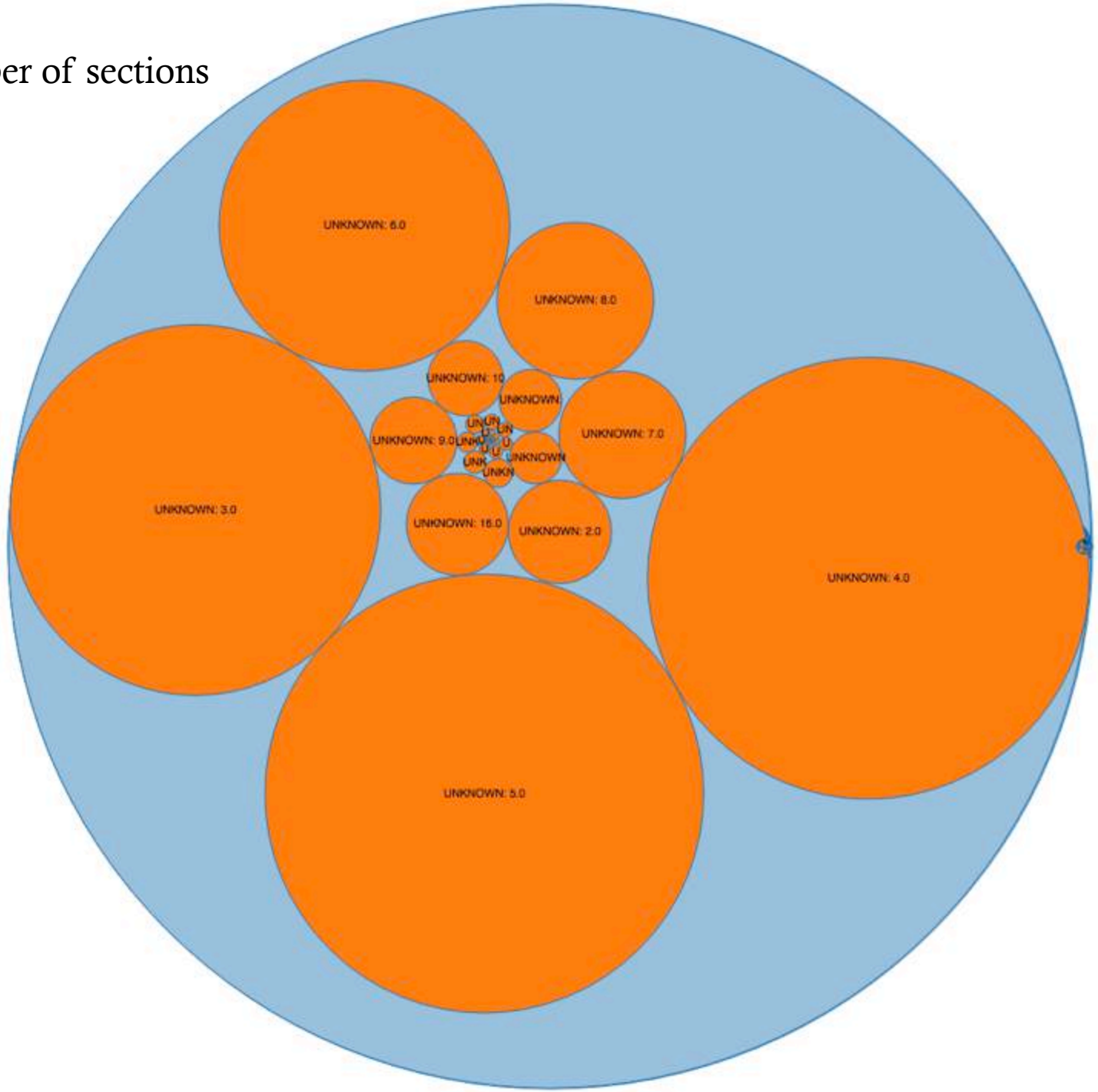
Random
Sub-clustered on linker versions



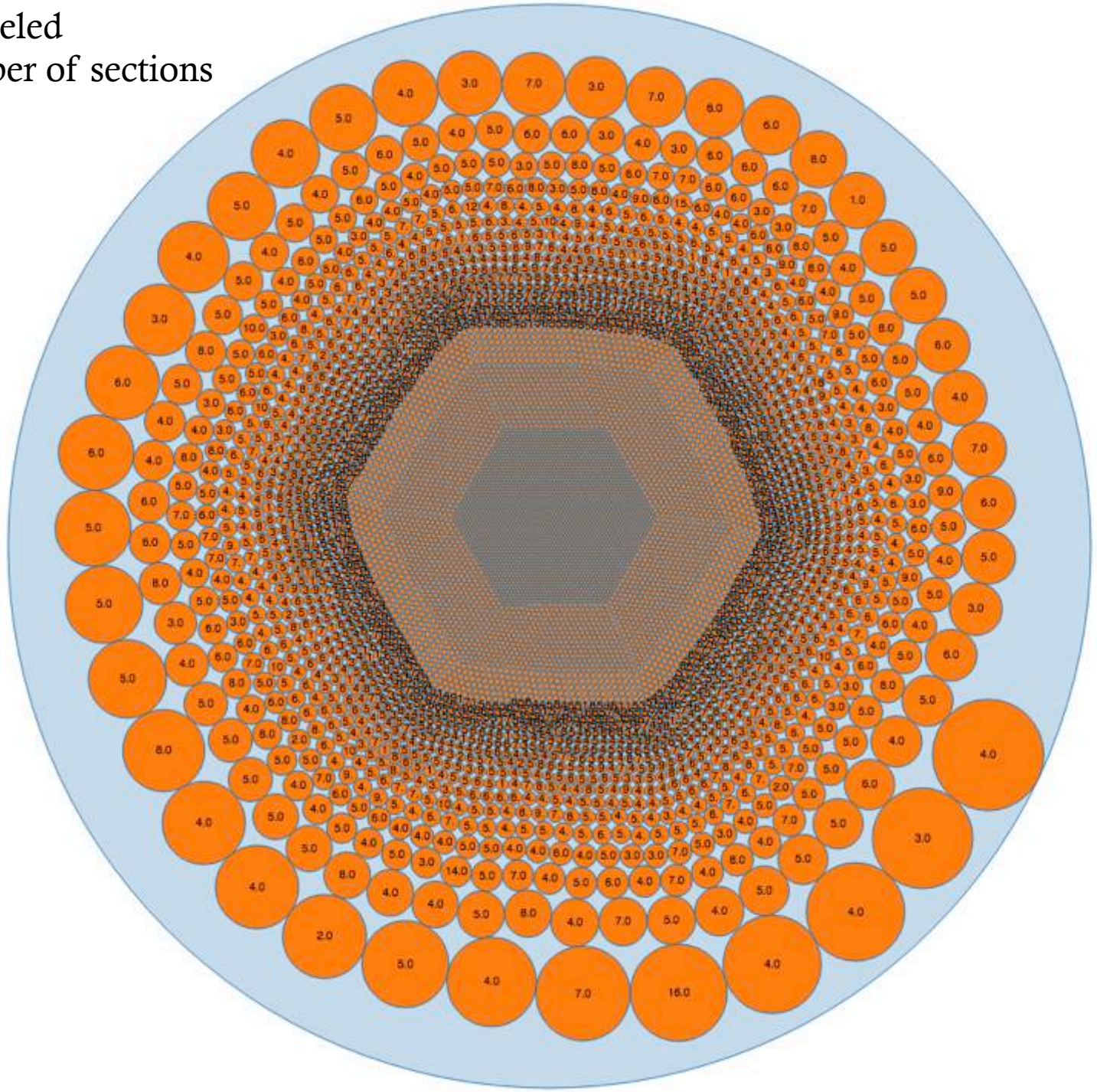
Random – Non-Unlabeled
Sub-clustered on linker versions



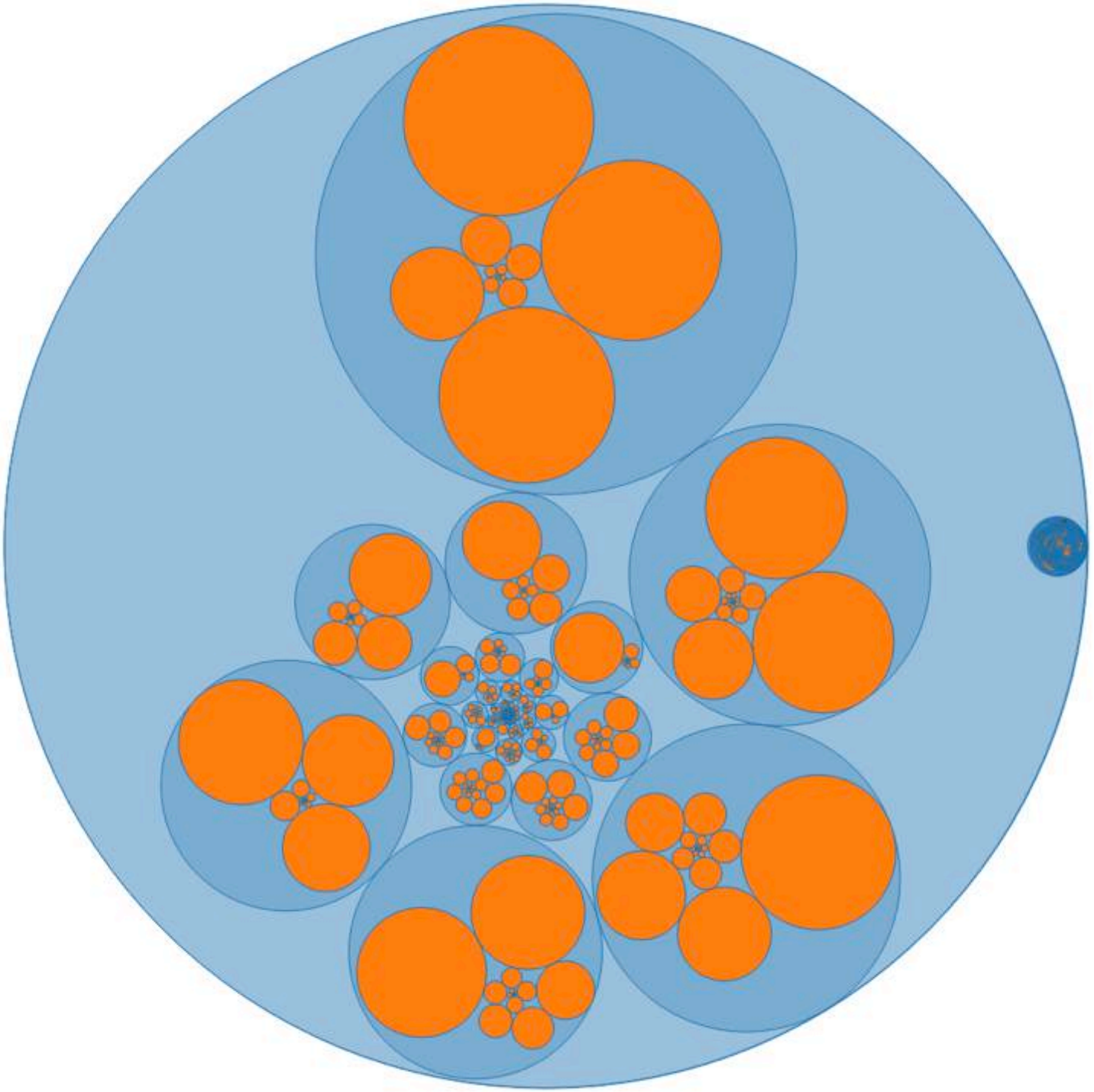
Random
Sub-clustered on number of sections



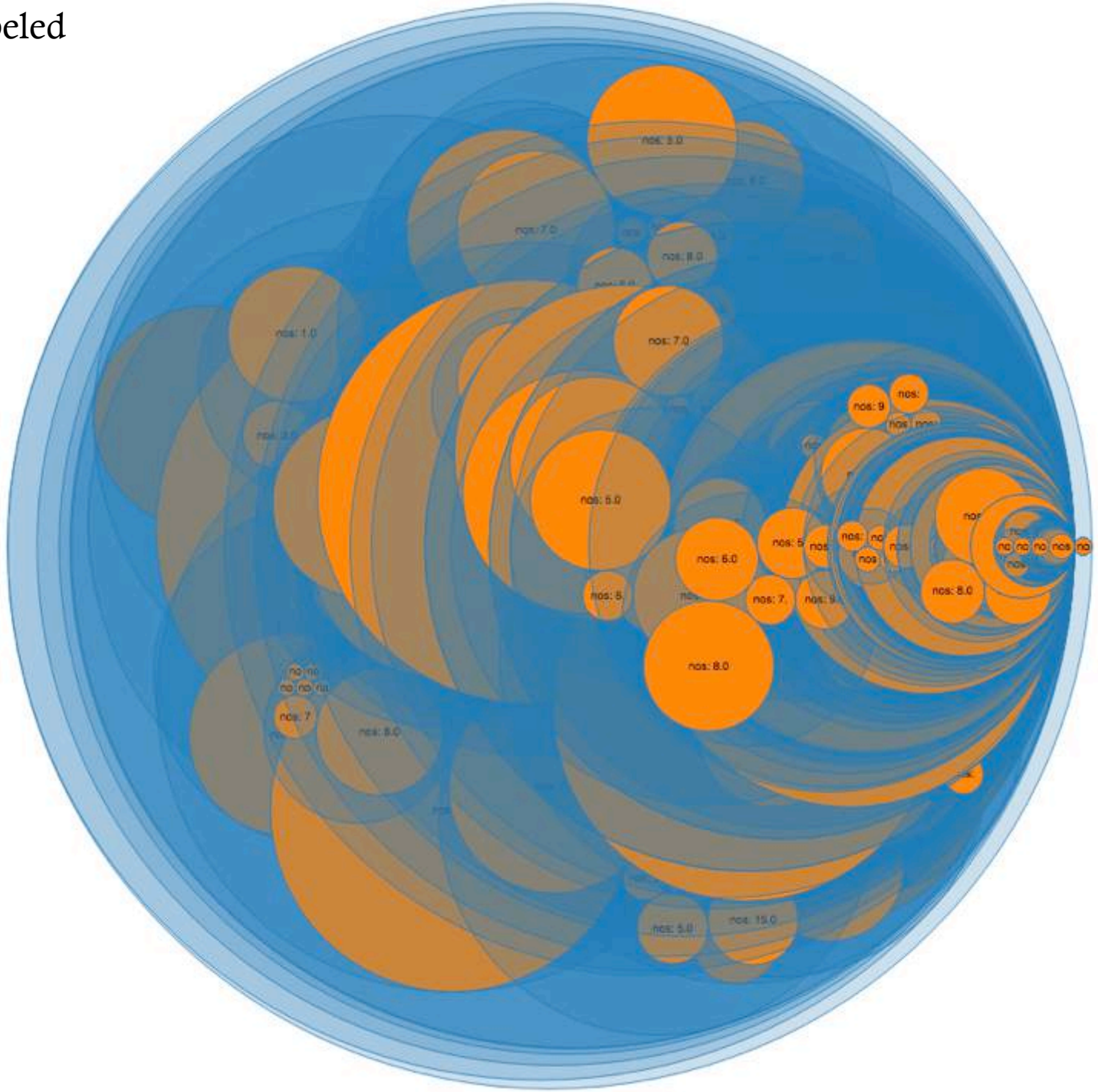
Random – Non-Unlabeled
Sub-clustered on number of sections



Random
Sub-clustered on both



Random – Non-Unlabeled
Sub-clustered on both



Google Chrome

AKA Why linkers and sections are important

- 97 files match the ASM signature
  - ['call', 'jmp', 'int3', 'int3', 'int3', 'int3', 'int3', 'int3', 'int3', 'int3', 'int3', 'int3', 'int3', 'int3', 'cmp', 'jae', 'cmp', 'jae', 'shrd', 'shr', 'ret', 'mov', 'xor', 'and', 'shr', 'ret', 'xor', 'xor', 'ret', 'int3']
- 95 have the same linker version: 10.0
  - 2 have a linker version of 11.0
- 94 have the same number of sections: 6
  - Two have 4 sections, one has 5 sections
- 94 binaries have BOTH a linker version of 10.0 AND 6 sections
  - All 94 are signed Google Chrome binaries

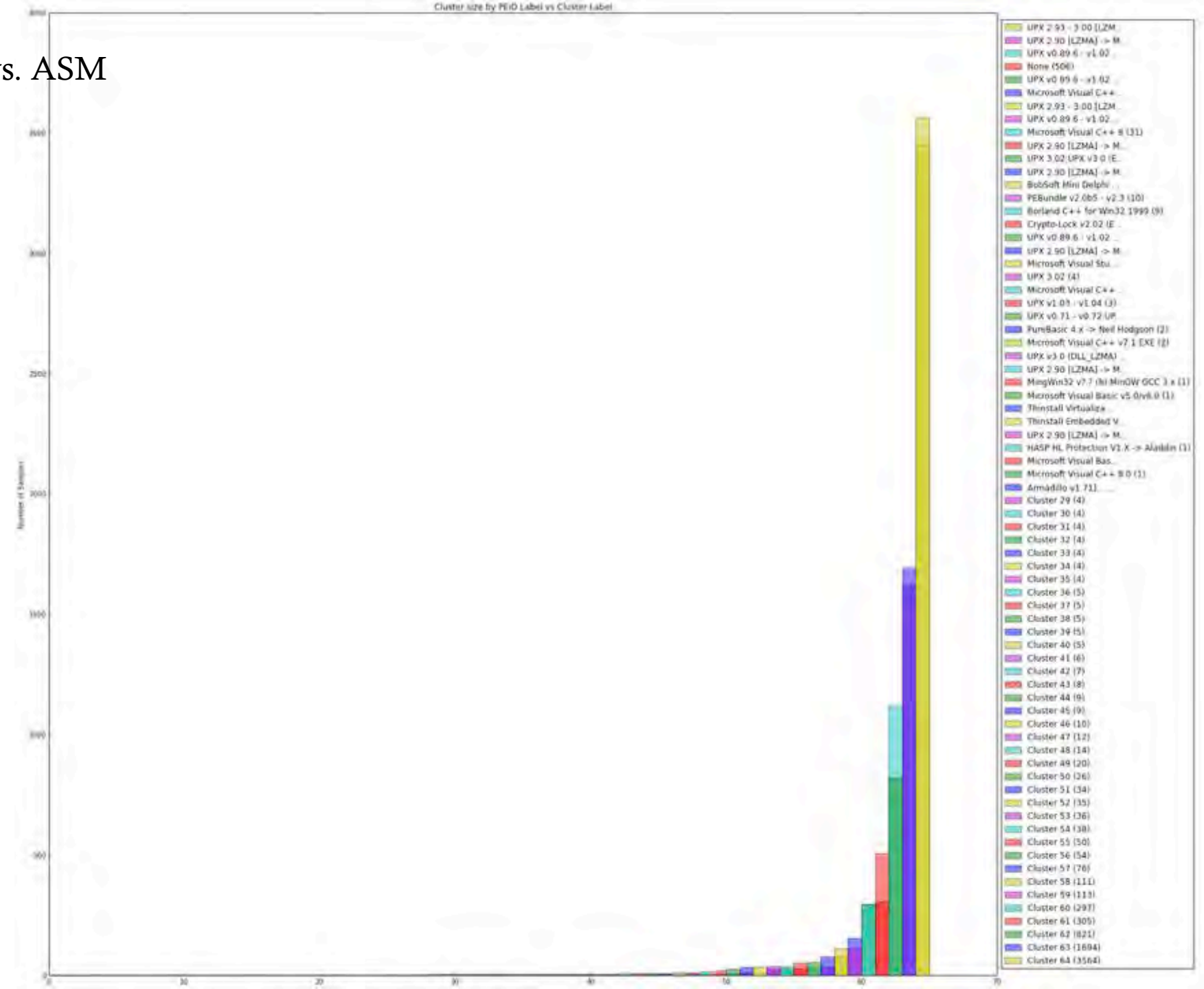
UPX

Because somebody was going to ask

- 7469 files packed with UPX (looked for UPX0, UPX1, or UPX! in the file)
- 65 different groups identified
  - Includes everything that didn't match anything else (catch all) – 76 samples
- 31 unique linker versions
- 13 unique number of sections
  - 6902 have 3 sections

Group Label	Count
2	3564
1	1694
0	821
25	305
24	297
83	113
123	111

UPX
PEiD vs. ASM



UPX numbers

PEiD Label	Count
UPX 2.93 - 3.00 [LZMA] -> Markus Oberhumer, Laszlo Molnar & John Reiser:UPX 3.02:UPX v3.0 (EXE_LZMA) -> Markus Oberhumer & Laszlo Molnar & John Reiser	3453
UPX 2.90 [LZMA] -> Markus Oberhumer, Laszlo Molnar & John Reiser	1626
UPX v0.89.6 - v1.02 / v1.05 -v1.24 -> Markus & Laszlo [overlay]	1121
None	506
UPX v0.89.6 - v1.02 / v1.05 -v1.22 (Delphi) stub	297

Group Label	Count
2	3564
1	1694
0	821
25	305
24	297

Solution recap

- Easy to generate signatures
  - Python script with minimal dependencies
- It involves Math, who doesn't love Math?
- Cross platform.
  - Python works everywhere, right?
- Easy to understand…ish
- It Works!

Future work

- The int3s are a side effect of the compiler adding bytes between functions or keeping aligned addresses (http://hooked-on-mnemonics.blogspot.com/2013/08/exploring-functions-with-undefinderpy.html)

Questions

I’m done

References

- Capstone Engine
- PEFile
- Tapered Levenshtein
- ZeuS dataset
- APT1 dataset