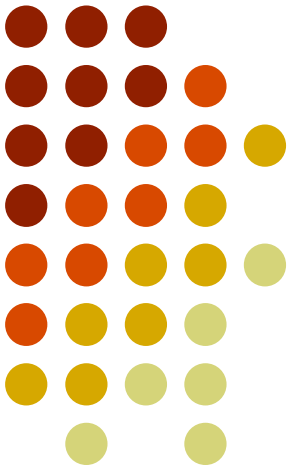
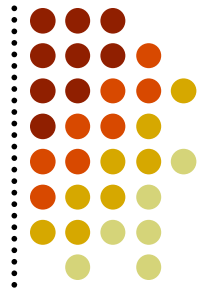
Analyzing All That Data

Techniques for Sifting Haystacks and Finding Needs



Jose Nazario <jose@monkey.org>

Payload Analsysis Problem Statement



The problem is not gathering data

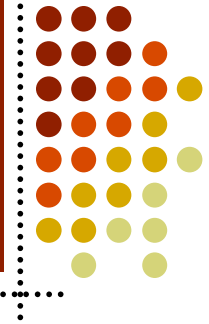
- We have instrumented over 1.2% of allocated IPv4
- Approximately 2 /8 networks worth
- We can collect spam, phishing mails in builk

The problem is in reliably detecting signal

- New attacks
- Reliably classify spam

Needle and haystack problem

Improving Payload Analysis



Allows for the identification of new worms

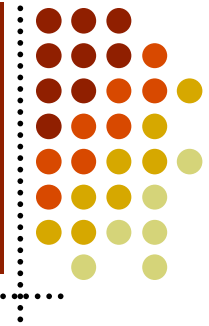
IMS calculates an MD5 hash for all payloads

- Similar payloads look wildly different
- See similar payloads … hash explosion

Similar is the trick

Motivation: improve new attack detection and characterization

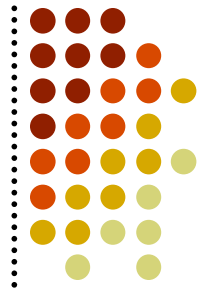
String Analysis Methods



Works on lists of bytes

Great for plain text (ie mail)

Classic Ways of Analysis



Look for substrings

Regular expressions

Hashes (SHA1, MD5)

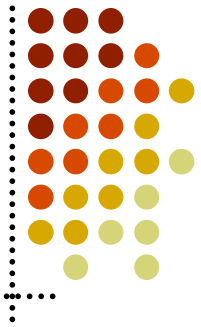
Problems

- Point changes cause match failure
- Regular expressions are boolean, not fuzzy
- Small changes fail to be detected as insignificant
- Complexity of expressions

The Viagra Example

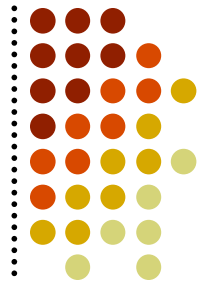

```
Vi--agra
Vi-ag-ra
VIAGR @
Via-gra
Vi.agr-a
vi__@gr@
Via-gra
Viagra
Via-gra
Vi--agra
Viagr,a
Viaqra
```

MD5 Difference Exaggeration



```
 Vi--agra -> eda81bbc0e8bb2bf76ecef1244d8de7e
Vi-ag-ra  -> ee686a6abfd4b52f784efc7c7c14add2
  VIAGR @ -> 73d2f57ff0bd46f4145c3c5a78654d94
  Via-gra -> 2a00142f56610cf610cffec99fba5b25
 Vi.agr-a -> 659f3ce1e8dc223d87e42d74b7390363
 vi__@gr@ -> 5e5fb2d7497c7448c753d24d1db4c46b
   Viagra -> 324cfde55522050027f396041089c2c7
  Via-gra -> 2a00142f56610cf610cffec99fba5b25
 Vi--agra -> eda81bbc0e8bb2bf76ecef1244d8de7e
  Viagr,a -> f206d1467274d8f68018b5ba69bdf497
   Viaqra -> 7f99a0d0db3fad96afbea4e973c5673c
```

Finding Similar Payloads



Multiple ways to discover similar payloads

Main ones:

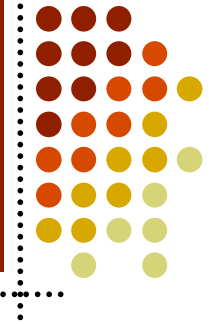
- Suffix trees
- String distance functions

Each has weaknesses and strengths

Bioinformatics Inspiration

- Use sequence alignment techniques
- Taking a cue from bioinformatics
  - Treat the payload as a sequence of bytes
  - Align similar inputs
  - Cluster groups of like inputs
  - Find consensus sequence

Suffix Trees



Builds up a tree of strings starting from string start (the root of the tree)

Branch creation at first difference

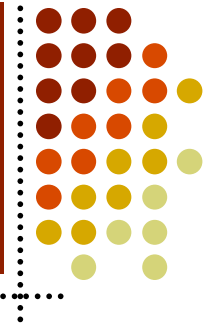
Implemented as a library from Christian Kreibich (libstree)

http://www.cl.cam.ac.uk/~cpk25/libstree/index.html

Dan Gusfield, *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, 1997.

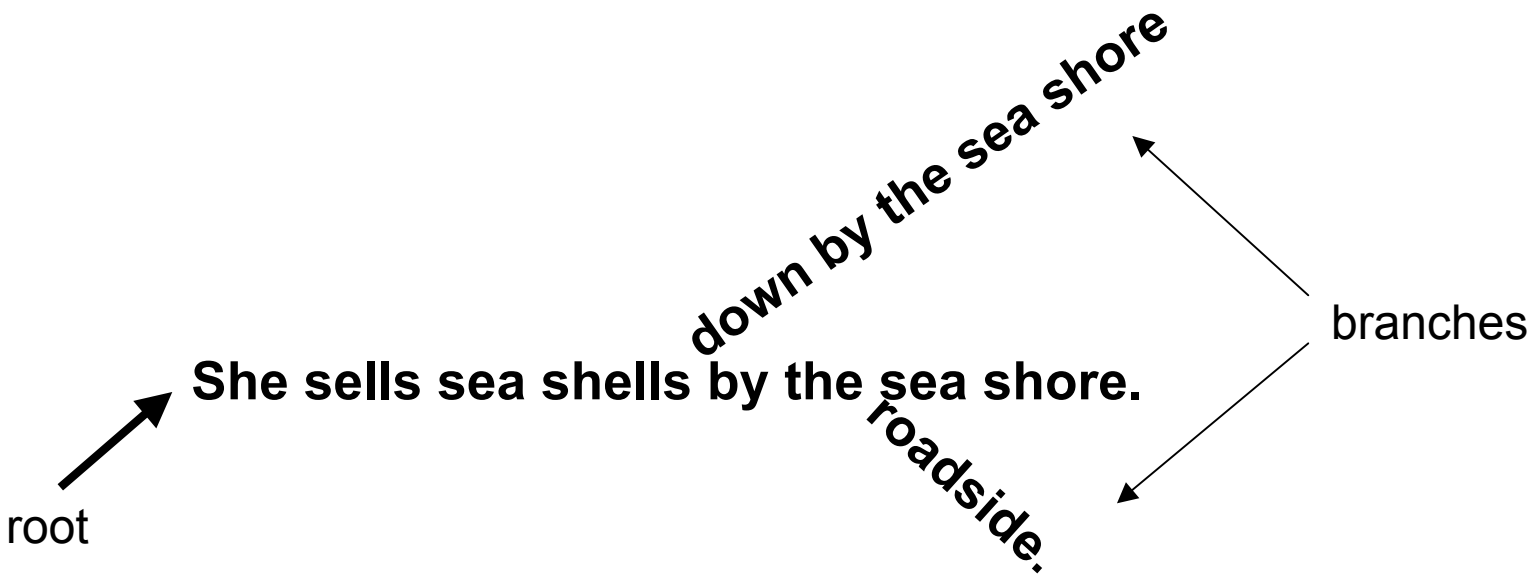
Esko Ukkonen, *On-line Construction of Suffix Trees*, Algorithmica, 14, 1995, 249-260.

3 Similar Sentences

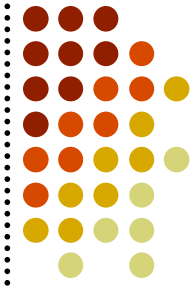


She sells sea shells down by the sea shore.
She sells sea shells by the sea shore.
She sells sea shells by the roadside.

Suffix Tree Example



Suffix Tree Problems



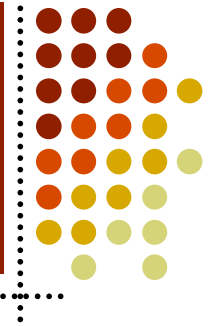
down by the sea shore

She sells sea shells by the sea shore.

roadside.

Infinite branching with different inputs
Branches never return to the root

String Distance Functions



Several popular functions

Calculate the “cost” of changing one string to another

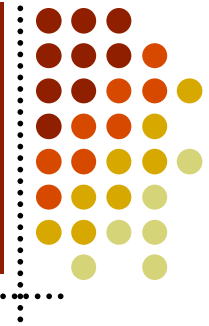
Cost is calculated differently

Not all distance functions work for all scenarios

Several are implemented in libdistance

http://monkey.org/~jose/software/libdistance/

Levenshtein Distance Function



Every change has a uniform cost of 1

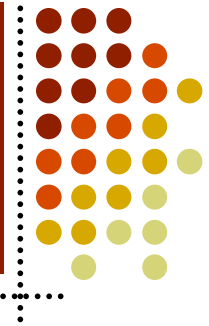
Changes are character replacements or insertions

She sells sea shells by the sea shore.
She sells sea shells down by the sea shore.

Cost is 5 to insert 5 spaces.

V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals", Doklady Akademii Nauk SSSR, 4, 163, 845-848, 1965.

Levenshtein Weaknesses



Not all conversions are equal

Compare:

`Viagara`

`V1@gara`

***LD(x,y)* = 2**

Not all insertion costs should scale linearly

`V i a g r a`

`V i a g r a`

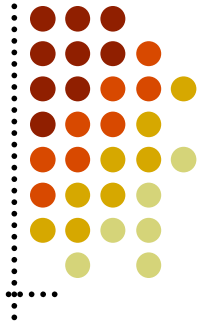
***LD(x,y)* = 20**

`Viagra`

`Welcome to KL Malaysia`

***LD(x,y)* = 20**

Hamming Distance Function



Position by position comparisons

Differences yield 1 point in cost of conversion

Inputs must be of same length

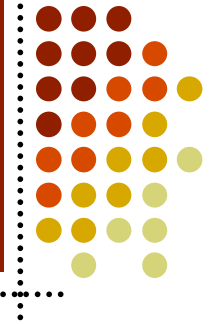
```
She sells sea shells by the sea shore.
She sells sea smells by the sea shore.
```

***HD(x,y)* = 1**

Useful for detecting minute changes in a large data set

R. W. Hamming, "Error Detecting and Error Correcting Codes", Bell System Tech Journal, 9, 147-160, April 1950.

Hamming Weaknesses



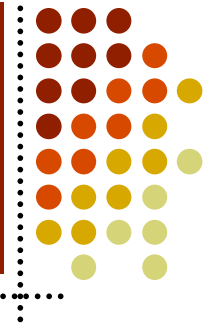
An earlypoint change can create a high HD() cost

```
She sells sea shells by the sea shore.
He sells sea shells by the sea shores.
```

***HD(x,y)* = 35**

Doesn't deal well with similar characters, misspellings

Damerau Distance Function



Similar to Levenshtein distance

Tolerates adjacent character swaps

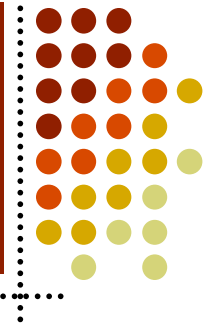
```
She sells sea shells by the sea shore.
She sells se ashells by teh sea shore.
```

DD(x,y) **= 0 (*LD(x,y)* = 4)**

Great for mis-spellers.

Fred J. Damerau, "A technique for computer detection and correction of spelling ", Communications of the ACM, 3, 7, 171-176, March 1964.

Needleman Wunsch Distance



Corrects some of the deficiencies of Levenshtein

Uses variable costs to calculate conversions and insertions

Conversion costs
a -> @: 0.1
a -> b: 1.0

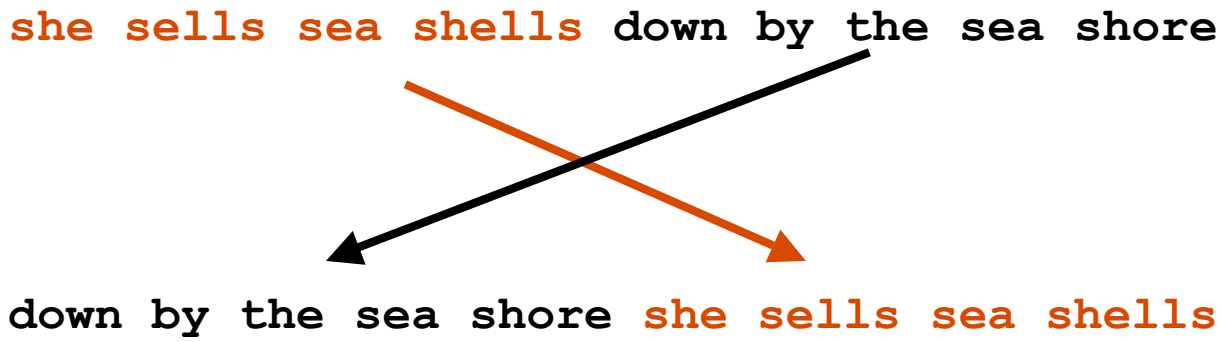
`Viagara`
`V1@gar@`
***DD(x,y)* = 0.3**

S. B. Needleman, and C. D. Wunsch, "A general method applicable to the search for similarities", Jrnl Molec Biol, 48, 443-453, 1970.

These Method Still Fail

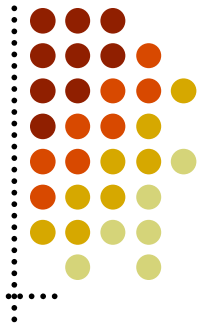
Transposed segments of payloads

Use the BLAST algorithm to detect this



Polymorphic attack payloads

Outputs of these Algorithms



Suffix trees return the longest common substring in a set

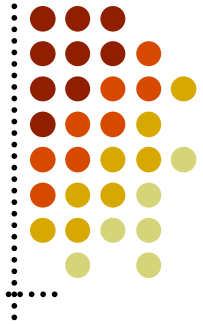
- Useful only if the starting set is closely related

```
She sells seashells along the roadside.
She sells seashells down by the sea shore
```

String distance algorithms return relative distances between inputs

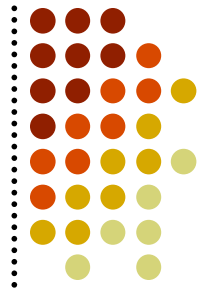
- Useful to cluster like payloads together

Current Use of These Algorithms



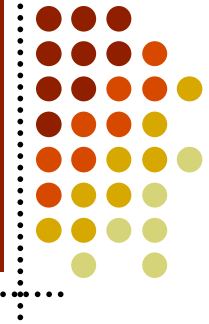
- Longest common substring analysis of payloads
  - Find similar looking payloads, look for substantial LCS
  - Reveals similar exploit usage across different attack sources
- Distances between payloads indicate mutants of malware, spam

Viagra Edit Distances



```
viagra ->   vi--agra: 2
viagra ->  vi-ag-ra : 3
viagra ->    viagr @: 2
viagra ->    via-gra: 1
viagra ->   vi.agr-a: 2
viagra ->   vi__@gr@: 4
viagra ->     viagra: 0
viagra ->    via-gra: 1
viagra ->   vi--agra: 2
viagra ->    viagr,a: 1
viagra ->     viaqra: 1
```

A Spam Friendly Cost Matrix



Default costs in points:

conversion costs 1

insertion costs 0.2

Cost to change case: 0

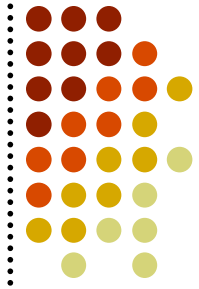
Cost to change from a to @: 0.1

Cost to change from i to 1: 0.1

Cost to insert - after a: 0.1

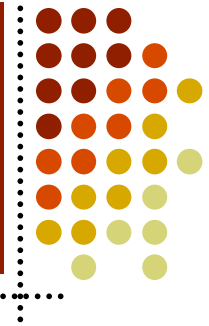
Cost to insert - after i: 0.1

NW Distances vs Edit Distances



Superviagra ->	V1AGRA:	0.3	11
Superviagra ->	Via-gra:	0.3	7
Superviagra ->	V,iagra:	0.4	6
Superviagra ->	Via-gra:	0.3	7
VIAGR @ ->	Superviagra:	0.8	11
VIAGR @ ->	Vi--agra:	0.8	7
VIAGR @ ->	via'gra:	0.8	7
VIAGR @ ->	V"iagra:	0.8	6
VIAGR @ ->	Vi-ag-ra:	0.9	7
VIAGR @ ->	"viag.ra:	1.0	8
Via-gra ->	V"iagra:	0.4	2
Via-gra ->	"viag.ra:	0.6	4
Via-gra ->	Viagra:	0.2	1

Clustering Analysis

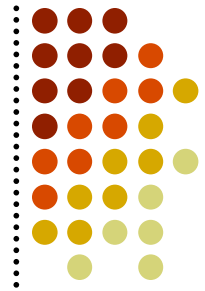


Find all pairs distances between payloads

```
for x in payloads:
    for y in payloads:
        d[x][y] = distance(x, y)
```

Next step is dependent on clustering algorithm

Clustering Data



K-means clustering

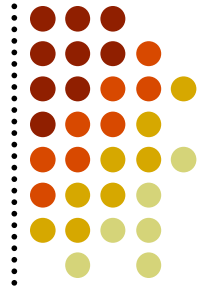
- Pick *k* cluster centers
- Calculate distances
- Recompute centers
- Repeat until convergence

Hierarchical clustering

- Root group
- Find furthest points, split group
- Recompute, repeat until all distances within tolerance

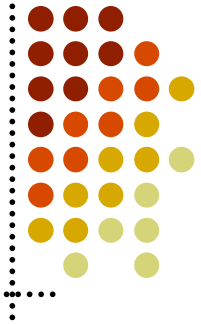
V. Guralnik and G. Karypis, Workshop on Data Mining in Bioinformatics (2001) 73-80.

Results from Clustering



- Reduced data complexity
  - From 1000's of points to dozens
  - Makes future classification easy
- Distill features from a cluster
  - Derive a consensus pattern
- Time dependent clustering can reveal mutations
- Phylogeny tree analysis

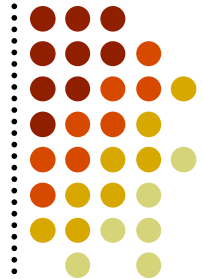
Initial Experiments



- Using spam
  - Rich data source, problem of interest
  - Have years worth of spam payloads
- Problems
  - Implementation inefficiencies
  - Runtime memory footprint
  - Allocating time to complete this

- Currently do payload analysis on popular payloads
- Haven't built up NW cost matrices for binary payloads

Code Availability



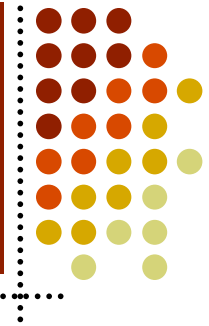
Libdistance has been released

- http://monkey.org/~jose/software/libdistance/
- Jose Nazario and Evan Cooke
- Current version: 0.2.0
- Implements these algorithsm
- 0.2.1 will have improved Python bindings
  - (written in the past day!)

LibStree 0.4, Christian Kreibach

- http://www.cl.cam.ac.uk/~cpk25/libstree/

Uses of Libdistance

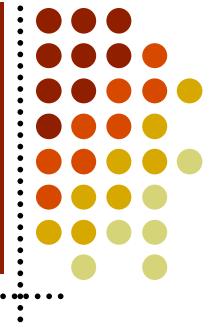


Logfile analysis

Mail sorting

HTTP request analysis

Using Libdistance (Python)

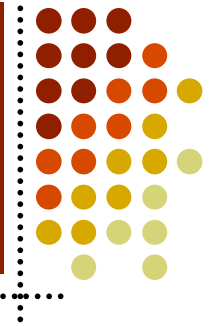


```
#!/usr/bin/env python

import distance

s1 = 'Viagra'
s2 = 'V    i    a    g    r    a'
s3 = 'Welcome to KL Malaysia'
print distance.levenshtein(s1, s2)
print distance.levenshtein(s1, s3)
```

More Libdistance (Python)

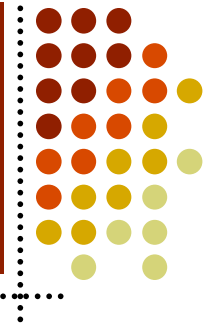


```
import distance

s1 = 'Viagra'
s2 = 'V    i    a    g    r    a'

m = distance.nw_matrix(insertion = 0.1, \
    conversion = 10.0)
m.setInsertion('a', ' ', 0.1)
m.setInsertion('i', ' ', 0.1)
print '%s -> %s: %.2f' % (s1, s2, \
      distance.needleman_wunsch(s1, s2, m))
del(m)
```

Graph Comparison Algorithms

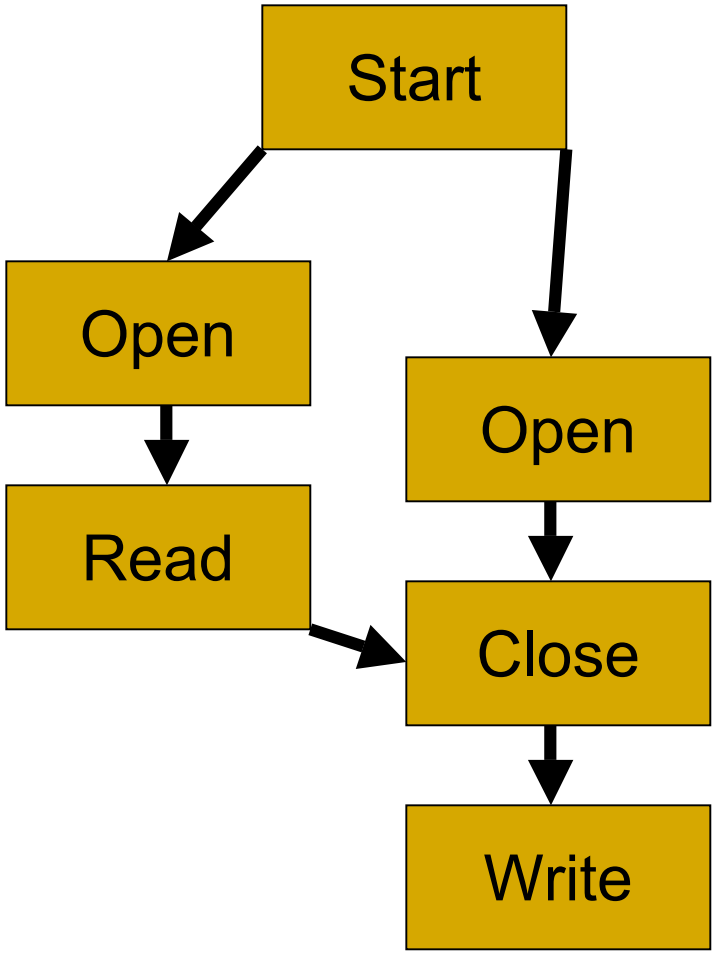


Typically built on a callgraph

Executables have specific patterns

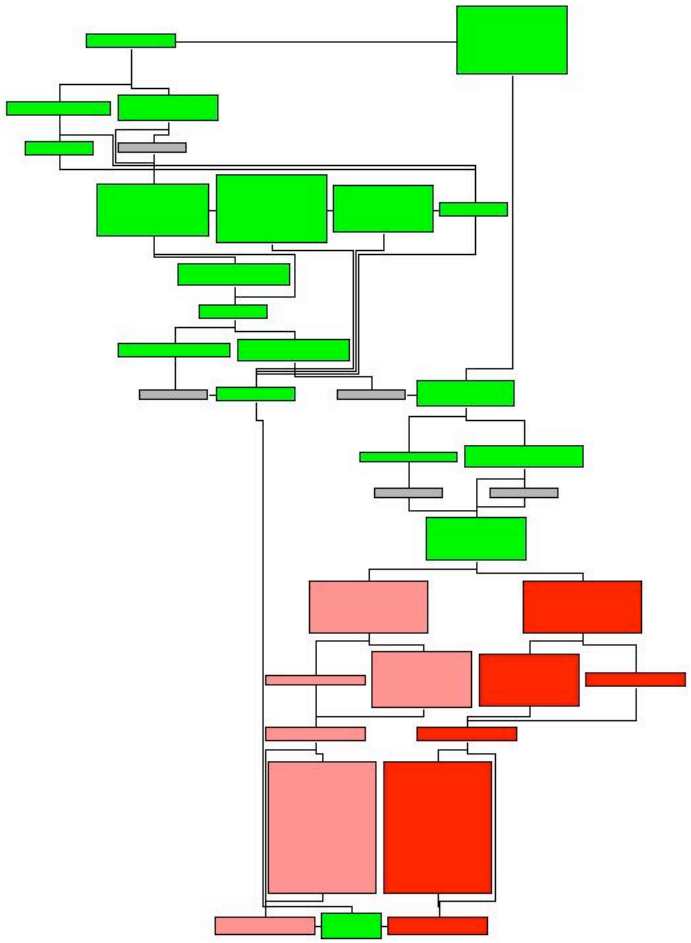
Families of executables have patterns

Example Callgraph



- Directed Graph
- Based on CALL, RETURN structure
- Can be obtained by tracing a process
- IDA Pro can also generate callgraphs

Graph Isomorphism Comparisons



- Lets you highlight only differences
- Useful in malware analysis, binary patch analysis
- Method for binaries described by Todd Sabin

COMPARING BINARIES WITH GRAPH ISOMORPHISMS
Todd Sabin, Bindview Research
http://www.bindview.com/Services/Razor/Papers/2004/comparing_binaries.cfm