Polygraph: Automatically Generating Signatures for Polymorphic Worms

The work of:

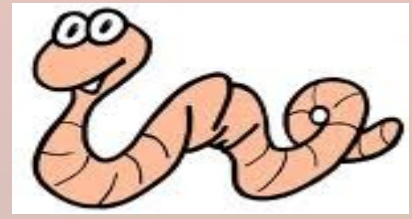
James Newsome, Brad Karp, and Dawn Song

Carnegie Mellon University

Proceedings of the 2005 IEEE Symposium on Security and Privacy

Origins

- The actual term "worm"' was first used in John Brunner's 1975 novel, The Shockwave Rider.



- Robert Morris, a Cornell U grad student, created a worm that disrupted 10% of the internet machines. (Nov 1988)
  - Estimated to have cost more than $100K to remove the worm.
  - He went to jail for it.

-wikipedia

What is a worm?

Worm: self replicates, it is passed from one machine to another, doesn't need to attach itself to an existing program.

Typical behavior: self replication, add a back door to the machine, send SPAM

Typical unwanted effect: absorb bandwidth, not necessarily intended to corrupt or modify the target machine.

-wikipedia

Polymorphic Code

Poly = many

Morph = change in shape

Polymorphic - able to have several shapes or forms.

Problem: How do you detect something that is always changing?

Worm detection

Worm detection with Intrusion Detection Systems (IDS)

- Signature generated manually
- Typical automated approach was to find a single substring match to identify the worm.
- Automatic signature generation systems: Honeycomb, Autograph, EarlyBird
- A signature from a polymorphic worm is too short to be on any use

Worm Properties

Behavior we can count on

What is its purpose?

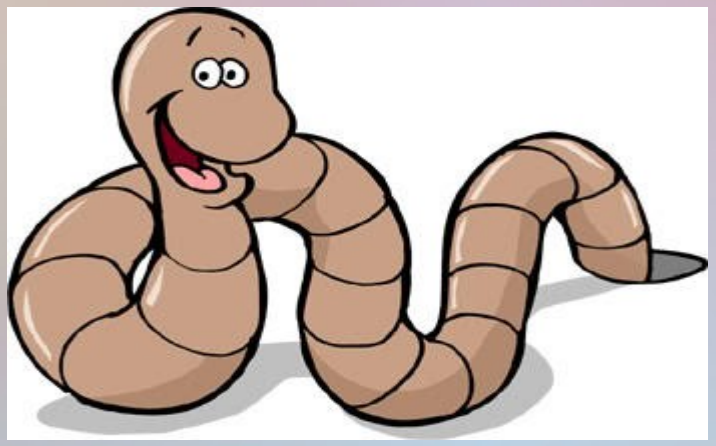
- Self replication

What stays the same?

- Fixed structure of the exploit
  - Attack method remains unchanged
- Imperfect obfuscation techniques
  - Polymorphic engines that do a poor job

Dissecting the worm

- Invariant bytes
  - Can't change or the exploit doesn't work
- Wildcard bytes
  - Any value
  - Has no effect on the worm's behavior
- Morphed bytes
  - Always changing
  - The code section



Tokens

- Forensics on worm exploits
  - Many worms contain may invariant bytes
  - Not all worms have invariant bytes (Limitation)
- Polymorphic worms have very small invariant strings
  - Too small to be of any use as a signature tool
  - Many invariant strings
- Tokens
  - A invariant string in the worm

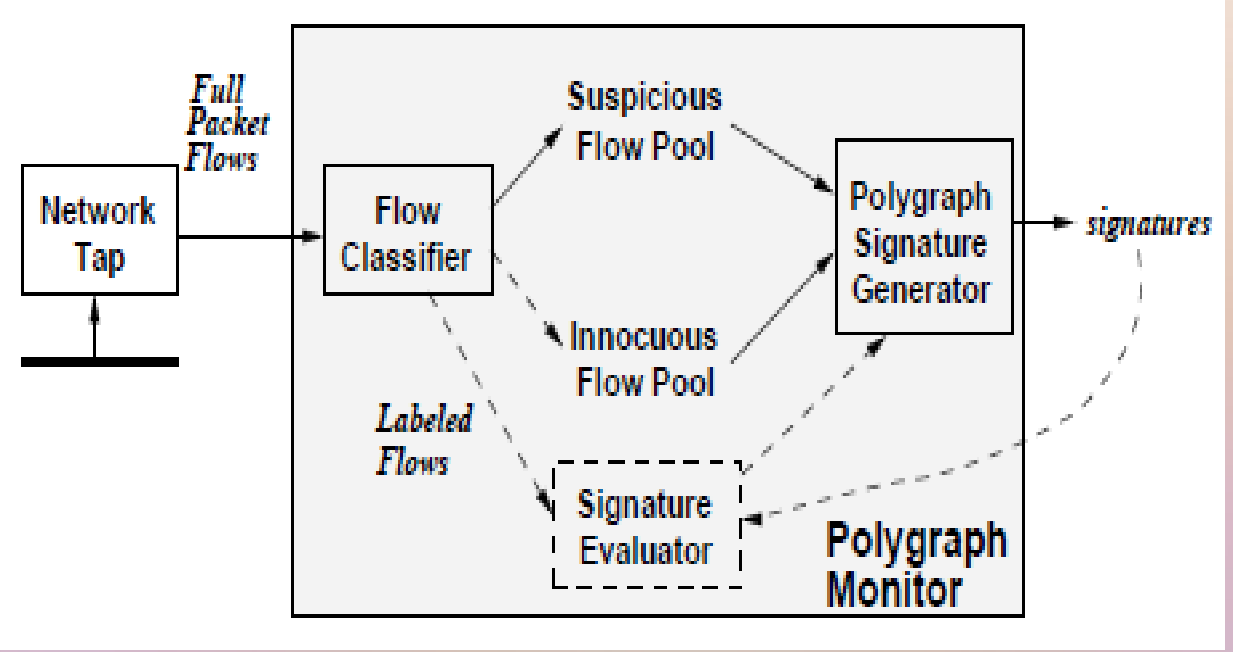
Signatures

- Tokens are building blocks for the signatures
  - Signature = set of tokens
- Conjunction Signatures
  - Fixed set of tokens found in any order (t3, t8, t2,..)
- Sequence Signatures
  - Fixed set of tokens in a set sequence (t1->t2 ->t3->...)
  - Regular expression (.t1.*t2.*T3.*,....)
  - Catches framing exploits

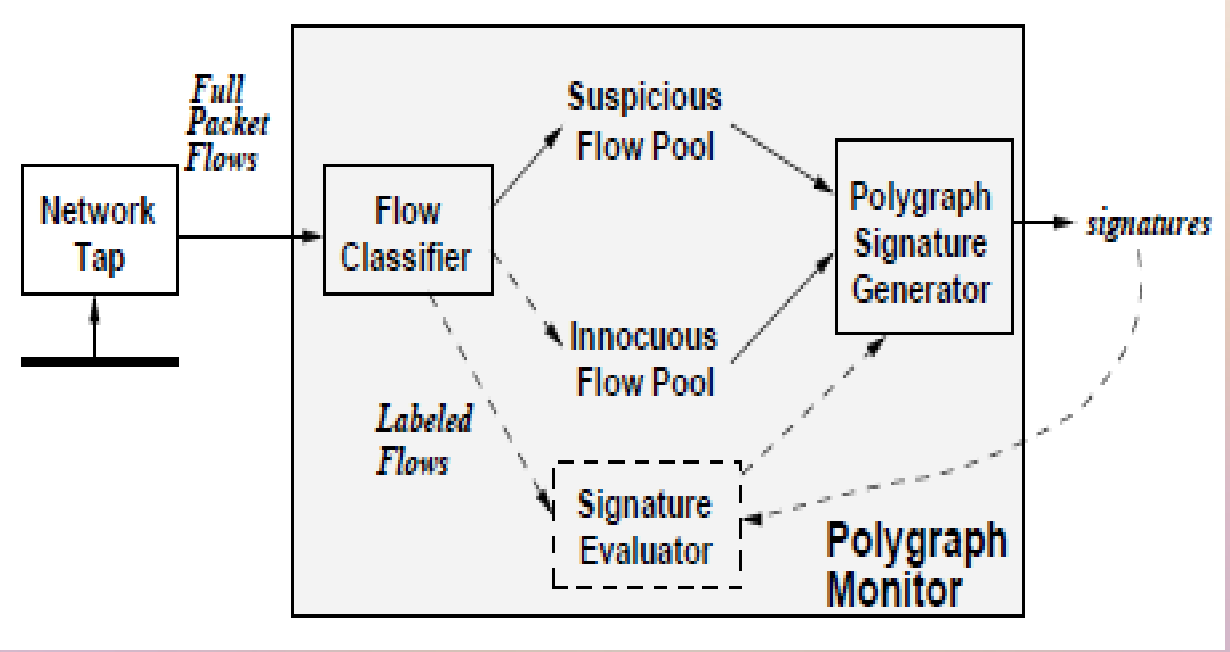
Signature

- Bayes Signatures
  - Match based on probabilities
    - Weights are assigned to each token
    - Match occurs when total weighting exceeds a threshold
  - Learned from a pool of unrelated worms as well as innocuous network activity
  - Higher computational cost
  - Reduces false negatives

Polygraph Architecture



- Flow classifier isn't perfect
  - Identifies suspicious traffic
  - This worm or that worm, it doesn't matter
  - Partitioned worms according to the destination port
  - Noise = misclassified innocuous traffic



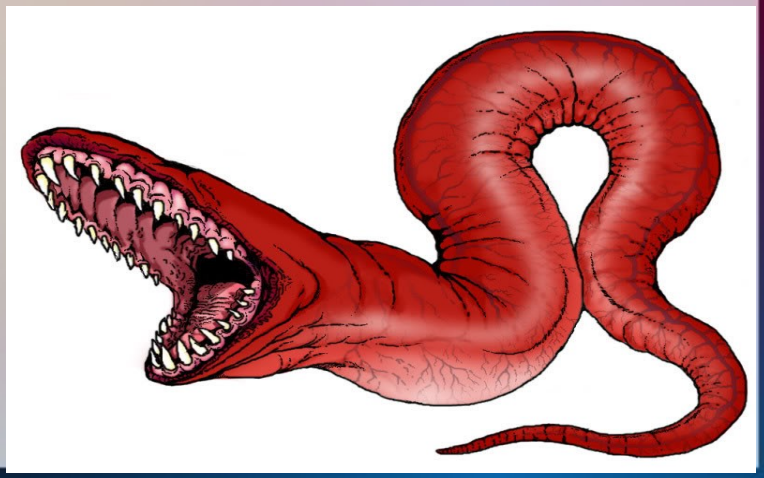
- Signature generator is the main focus
  - Innocuous pool is used to reduce false positives
  - Signature evaluator is future work

Design goals

- Low false positive / negative detections
- Minimize computational cost
  - Efficient algorithms
- Minimize the number of signature generated
  - One per morphed worm is unacceptable
  - Still maintain signature quality
- Unaffected by noise or mixed set or worms
  - A different worm family will generate a different signature

Conjunction Algorithm

- Generate the token list
  - Find the invariant strings
- Use the list to create the conjunction signature list
  - Which tokens are in “this” worm
    - Search the worm for tokens



Sequence Algorithm

- Sequence signature
  - Reduce the worm to token series
    - Ignore non-token strings
  - Need a string alignment algorithm that prefers subsequences with contiguous (large) substrings
    - Smith-Waterman alogrithm
      - Points are added for no-gaps (contiguous)
      - Points are subtracted for gaps (non contiguous)
      - The higher the sum, the more contiguous the match

Bayes Algorithm

- Bayes signature
  - Calculates the probability the at token is contained in the sample
  - Filters out the noise in the suspicious pool
  - Probability that a token is in a string is independent of other tokens in the string
  - Works best with a large number of tokens and moderate sized suspicion pool
  - Using the innocuous pool as input reduces the false positive detections
  - Calculation biased to reduce false positives

Clustering

- Noise is still a problem for the conjunction and sequence signatures
- Hierarchical Clustering is used to group common worm signatures
  - Set of suspicious flows and the signature that identifies the set
  - Merging the 2 clusters produces a more sensitive signature
  - Use the new signatures against the innocuous pool
  - Select the signature that produces the fewest false positives

Evaluation

- Data set was created with known worms and scanned network traffic
  - Suspicious tool size > 2
  - HTTP: incoming and out going
  - DNS traffic
- Tested with only one worm in the suspicious pool
- Add noise and retest
- Test with multiple worms and noise

Results

Class	False +	False −	Signature
Longest Substring	92.5%	0%	HTTP/1.1\r\n
Best Substring	.008%	0%	\xFF\xBF
Conjunction	.0024%	0%	'GET ', ' HTTP/1.1\r\n', ': ', '\r\nHost: ', '\r\n', ': ', '\r\nHost: ', '\xFF\xBF', '\r\n'
Token Subsequence	.0008%	0%	GET .* HTTP/1.1\r\n.*: .* \r\nHost: .* \r\n.*: .*\r\nHost: .*\xFF\xBF.*\r\n
Bayes	.008%	0%	'\r\n': 0.0000, ': ': 0.0000, '\r\nHost: ': 0.0022, 'GET ': 0.0035, ' HTTP/1.1\r\n': 0.1108, '\xFF\xBF': 3.1517. Threshold: 1.9934

HTTP worm

Class	False +	False −	Signature
Longest Substring	.3279%	0%	\x00\x00\xFA
Best Substring	.0023%	0%	\xFF\xBF
Conjunction	0%	0%	'\xFF\xBF', '\x00\x00\xFA'
Token Subsequence	0%	0%	\xFF\xBF.*\x00\x00\xFA
Bayes	.0023%	0%	'\x00\x00\xFA': 1.7574, '\xFF\xBF': 4.3295 Threshold: 4.2232

DNS worm

Cost of Computation

- Hardware
  - Single desktop Linux machine
    - Pent III running at 1.4GHz
- Conjunction, Sequence, Bayes signatures generation is quick
  - 10 sec each for a 100 sample set
- Cluster refinement is time consuming
  - Depends on sample size
    - O(#samples^2)
  - Signatures generated in <10 min for a 25 sample set

Conclusion

- The IDS approach toward polymorphic worm detection is feasible
- High quality worm signatures can be generated automatically
- Reliable even if the sample pool contains noise and multiple worm families

Questions?