The 4th International Workshop on
Software Engineering for Secure Systems (SESS'08)
17-18 May 2008, Leipzig, Germany

LISABETH: Automated Content-Based Signature Generator for Zero-Day Polimorphic Worms

Lorenzo Cavallaro, Andrea Lanzi, Luca Mayer, and Mattia Monga

Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano, Italy

Worm

A *worm* is an independently replicating and autonomous infection agent, capable of seeking out new hosts and infecting them via the network

J. Nazario, 2006

Polymorphic worm

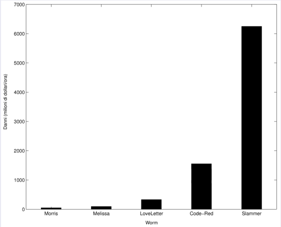
A worm is said to be *polymorphic* if able to change its binary representation during the spreading process using techniques like self-encryption or semantic-preserving code transformations

Typical infection pattern

1. **Scan:** the worm has to hunt out other network nodes to infect
2. **Compromise:** the worm has to launch an attack against an identified target
3. **Replicate:** the worm has to replicate and install itself on the victim

Weaknesses

With their typical scan, compromise and replicate pattern, worms can infect all the vulnerable hosts in a matter of few hours or even minutes



Signature-based approach

Toward defending against worms, the research community has proposed different kind of Intrusion Detection Systems (IDS).

Signature-based IDS filter incoming and outcoming network traffic for known *signatures* that correspond to maliciuos worms' flows samples

Speed up signature generation

To face low pace of manual signature generation process and to speed up this task, some automatic signature generation systems have been developed

Signature generation systems
Design goals

Required features

In order to work effectively, a signature generation system must meet several design goals:

- Automation
- Robustness against polymorphism
- Efficiency
- Network based
- Resilience to poisoning
- Effectiveness

Signature effectiveness

A good signature generation system must generate signatures that offer low false positive rate for innocuous traffic and low false negative rate for worm instances

Approach

- Use of an imperfect flow classifier able to separate innocuous and maliciuous network flows
- Content-based signature generation
- Invariant bytes presence assumption
- Greedy approach in signatures generation
- Generation of the most specialized signature for each worm

Multiset signatures

Multiset signatures are multi-set of *tokens*, and tokens are sequences of bytes.
We define a signature $\mathcal{S}$ as: $\mathcal{S} = \{(t_1, n_1), (t_2, n_2), \ldots, (t_k, n_k)\}$.
We say that a network flow $\mathcal{G}$ *matches* $\mathcal{S}$ if contains at least n_j copies of each t_j of $\mathcal{S}$, $\forall j \in [1, \ldots, k]$

Typical polymorphic worm structure

In a polymorphic worm sample we can classify three kind of bytes:

- **Invariant bytes:** with a fixed value in every possible instance and absolutely necessary for the exploit (protocol framework, exploit bytes)
- **Code bytes:** even if subjected to polymorphism and encription techniques some can be invariant (worm body, polymorphic decriptor)
- **Wildcard bytes:** may take any value

Assumption

Capitalize the invariant bytes presence to build an effective signature for a given worm

Generator weaknesses

- Best-local choice approach of the iterative greedy algorithm
- Strong constraints on partial signatures
- Assumption that all invariants byte are good invariant bytes
- Attempt to build the most specialized signature

Some assumptions and the approach used in algorithm design lead Hamsa to be effectiveless if exposed to some evasion attacks.

Effective attacks

- *Normal Pool Poisoning* attacks
- *Suspicious Pool Poisoning* attacks

Hamsa suffers some of well known normal and suspicious pool poisoning attacks.

Normal Pool Poisoning attacks

The adversary places specially crafted network flow samples in the innocuous pool to mislead signature generator. For example:

- Single Invariant poisoning attack
- Complete Signature poisoning attack

Suspicious Pool Poisoning attacks

The adversary places some network flow samples inside suspicious pool to mislead signature generator. For example:

- Dropped Red Herring attack
- Random Red Herring attack

Here, we present a new Suspicious Pool Poisoning attack

The new attack proceeds iteratively sending specially crafted suspicious network flows some with effective worms and others only to mislead signature generation.

Attack's phases

1. Poisoning of suspicious pool with worm and fake flows. Worm and fake flows contains fake invariants
2. If some trivial constraints are respected a single uneffective signature is returned. The signature contains only fake invariants
3. The adversary sends new worms and fake flows using a different set of fake invariants

This new attack offers better results than already known.

New attack benefit

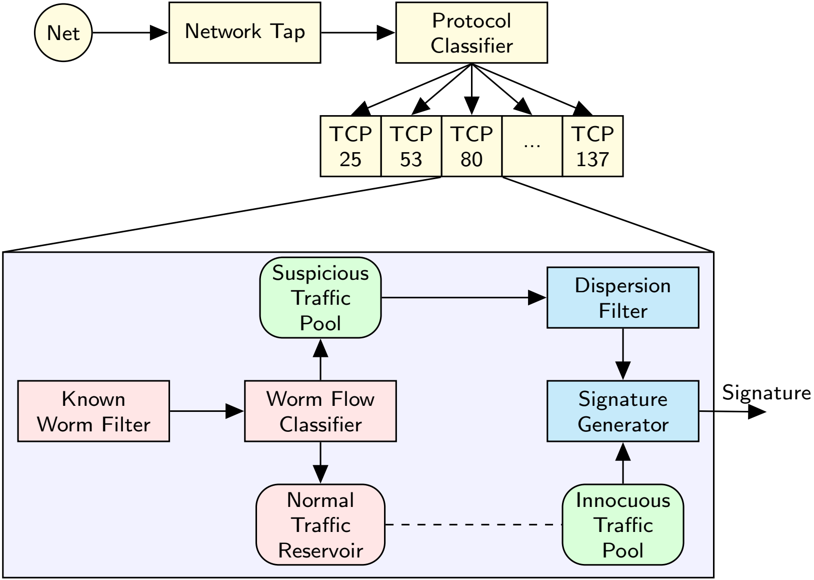
- No true invariants presence in any signatures
- No synchronization required between different worm instances attacking the same network
- Lesser fake network traffic generation required
- No links between different uneffective signatures
- No limitations on the number of attack iterations

The high level architecture of Lisabeth is very similar to Hamsa's and Polygraph but with some important changes

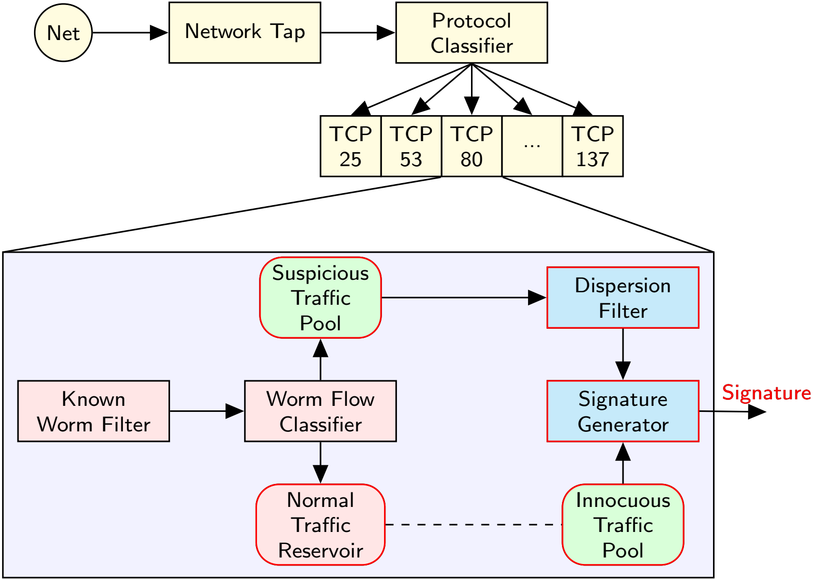
Design guide lines

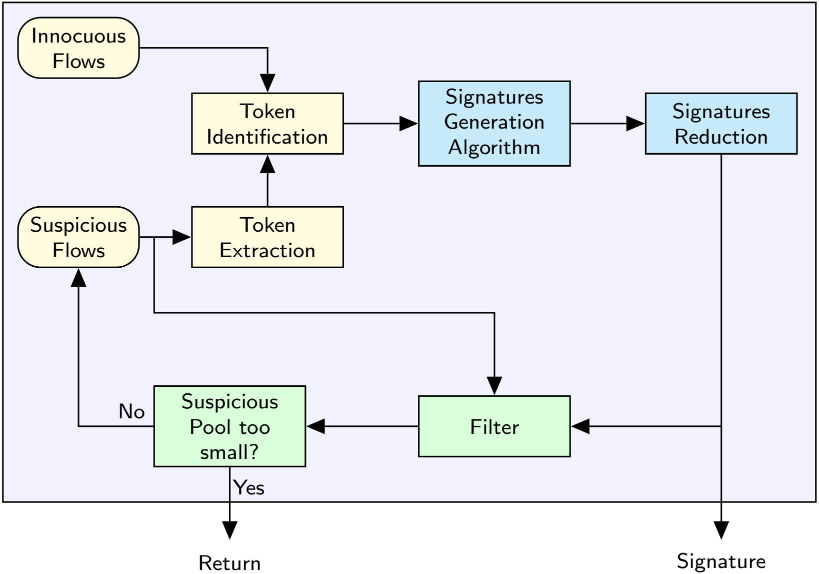
The idea is to minimize the assumption set on the attacker behaviour. So we assume that:

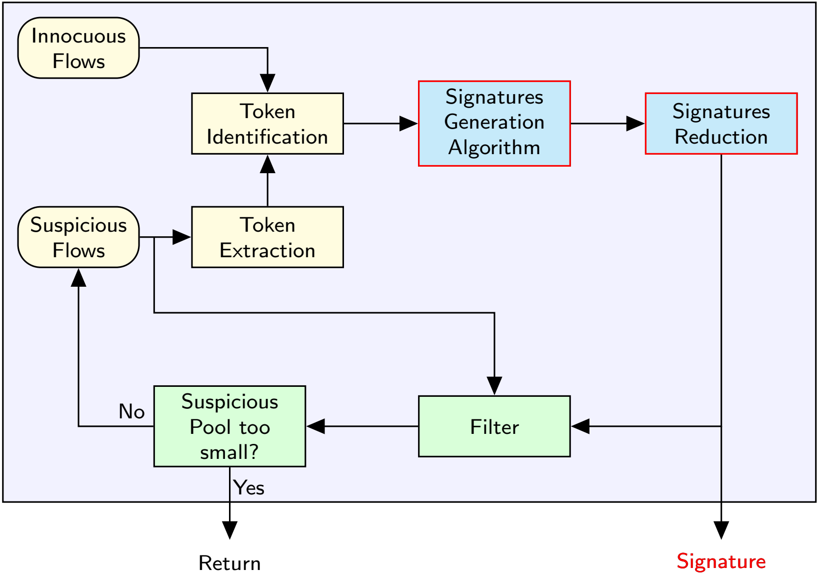
- Exists a set of recurrent invariant bytes
- Generate many signatures if in doubt
- Mantain global vision during signature generation process
- Limit false positive rate of signatures



Net
Network Tap
Protocol Classifier
TCP 25
TCP 53
TCP 80
...
TCP 137
Suspicious Traffic Pool
Dispersion Filter
Known Worm Filter
Worm Flow Classifier
Signature Generator
Signature
Normal Traffic Reservoir
Innocuous Traffic Pool


Net
Network Tap
Protocol Classifier
TCP 25
TCP 53
TCP 80
...
TCP 137
Suspicious Traffic Pool
Dispersion Filter
Known Worm Filter
Worm Flow Classifier
Signature Generator
Signature
Normal Traffic Reservoir
Innocuous Traffic Pool


Innocuous Flows
Token Identification
Signatures Generation Algorithm
Signatures Reduction
Suspicious Flows
Token Extraction
No
Suspicious Pool too small?
Filter
Yes
Return
Signature

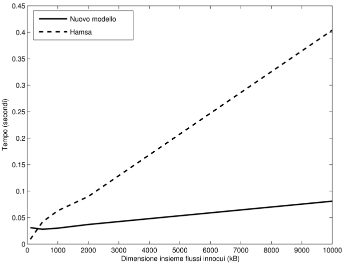

Innocuous Flows
Token Identification
Signatures Generation Algorithm
Signatures Reduction
Suspicious Flows
Token Extraction
No
Suspicious Pool too small?
Filter
Yes
Return
Signature

Effectiveness

- Noise tollerant (until 50%)
- Low false positive rate (0.095%) and false negative
- Resilient against many known *Suspicious Pool Poisoning attacks* and high resilience against *Normal Pool Poisoning attacks*

Efficiency

- More efficient than Hamsa working with big innocuous flows pools
- Equally sensible on suspicious flows pools size



Main contributions

- Discovery of a new attack against Hamsa
- Design of a new signature generation model more efficient, effective and resilient than old ones
- Realization of an experimental prototype of Hamsa, augmented with some of the proposed improvements for that model
- Integration of the proposed model in the prototype

Future enhancements

- Use of a distributed environment to collect network flows and split algorithm's computational costs
- Use of the same approach to classify unsolicited email messages (SPAM)

THANK YOU FOR YOUR ATTENTION

Questions?