Misleading Worm Signature Generators Using Deliberate Noise Injection

Roberto Perdisci^+, David Dagon^, Wenke Lee^,
Prahlad Fogla^, Monirul Sharif^

^Georgia Institute of Technology, Atlanta, GA, USA
^+University of Cagliari, ITALY

Presented by: Roberto Perdisci
Outline

- Introduction
- Syntactic-based Signature Generators
- "Traffic-based" flow classifiers
- Noise Injection Attack
- Case study: Misleading POLYGRAPH
- Experimental Results
- Conclusion
Introduction

Automatic signature generation look for invariants in polymorphic worms

- **Syntactic-based:**

- **Semantic-based:**

**Our contribution:** **Syntactic-based** signature generators are **vulnerable to Noise Injection Attack**
Syntactic-based Signature Generators

Signature generator

Flow classifier → Worm flows → Signature generation → Firewall/NIDS

Live Traffic → Network tap → Protected Network
Syntactic-based Signature Generators

Signature generator

Flow classifier → Worm flows → Signature generation → Firewall/NIDS

Live Traffic

Protected Network
Syntactic-based Signature Generators

- **Network tap**
- **Live Traffic**
- **Flow classifier**
- **Worm flows**
- **Signature generation**
- **Firewall/NIDS**
- **Protected Network**
Syntactic-based Signature Generators

Signature generator

Flow classifier → Worm flows → Signature generation → Firewall/NIDS

Live Traffic → Network tap

Protected Network
Syntactic-based Signature Generators

Network tap

Live Traffic

Signature generator

Flow classifier

Worm flows

Signature generation

Firewall/NIDS

Protected Network

Live Traffic

Network tap
Syntactic-based Signature Generators

Signature generator

Flow classifier → Worm flows → Signature generation → Firewall/NIDS

Live Traffic

Network tap

Protected Network

Stop

Monday, September 6, 2010
Syntactic-based Signature Generators

Flow classifier → Worm flows → Signature generation → Firewall/NIDS

- Live Traffic
- Network tap
- Worm
- Stop

- Simulated Honeynet
- Double Honeynet (sim. Honeynet + “real” honeynet)
- Port-scanning detector
- Anomaly IDS (e.g., byte frequency-based classifiers)

Protected Network

Monday, September 6, 2010
Traffic-based flow classifiers
Traffic-based flow classifiers

Simulated Honeynet

A

Suspicious Flow Pool
Traffic-based flow classifiers

Simulated Honeynet

Suspicious Flow Pool
Traffic-based flow classifiers

Simulated Honeynet

A

Suspicious Flow Pool
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

A

Suspicious Flow Pool

GW

Layer-2

Layer-1

Suspicious Flow Pool

A
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

GW

Layer-2

Layer-1

Suspicious Flow Pool

Suspicious Flow Pool

A

A
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

GW

Layer-2

Suspicious Flow Pool

Layer-1

Suspicious Flow Pool
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

GW

A

Suspicious Flow Pool

Layer-1

X

Suspicious Flow Pool

Layer-2

A

Monday, September 6, 2010
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

Monday, September 6, 2010
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

Port-scanning detection
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

Port-scanning detection

GW

Layer-1

Layer-2
Traffic-based flow classifiers

Simulated Honeynet

A

Suspicious Flow Pool

Double Honeynet

GW

A

Suspicious Flow Pool

Layer-1

X

Layer-2

Suspicious Flow Pool

Port-scanning detection

A

Suspicious Flow Pool

Monday, September 6, 2010
Traffic-based flow classifiers

Simulated Honeynet

A

GW

Suspicious Flow Pool

Suspicious Flow Pool

Double Honeynet

A

GW

Layer-2

Layer-1

Suspicious Flow Pool

Port-scanning detection

A

Suspicious Flow Pool

Suspicious Flow Pool

Monday, September 6, 2010
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

Port-scanning detection
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

Port-scanning detection
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

Port-scanning detection

Anomaly IDS

Monday, September 6, 2010
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

Port-scanning detection

Anomaly IDS

Monday, September 6, 2010
Traffic-based flow classifiers

Simulated Honeynet

Double Honeynet

Port-scanning detection

Anomaly IDS

Monday, September 6, 2010
Noise Injection Attack

Noise Injection Attack “poisons” the suspicious flow pool dataset with fake anomalous flows

Signature generator

Flow classifier → Worm flows → Signature generation

Live Traffic → tap
Noise Injection Attack

- Noise Injection Attack “poisons” the suspicious flow pool dataset with fake anomalous flows

Signature generator

Worm + F.A.F.
Noise Injection Attack

- Noise Injection Attack “poisons” the suspicious flow pool dataset with fake anomalous flows

Signature generator

Flow classifier

Worms + F.A.F.

Useless signatures

Live Traffic

Worm + F.A.F.

tap

Monday, September 6, 2010
Noise Injection Attack

Noise Injection Attack “poisons” the suspicious flow pool dataset with fake anomalous flows

Signature generator

Flow classifier \[ \rightarrow \] Worms + F.A.F. \[ \rightarrow \] Signature generation

Useless signatures

too many FP and/or FN

Live Traffic

Worm + F.A.F.
Noise Injection Attack

- Noise Injection Attack “poisons” the suspicious flow pool dataset with fake anomalous flows

Signature generator

Flow classifier \rightarrow \text{Worms + F.A.F.} \rightarrow \text{Signature generation} \rightarrow \text{Useless signatures}

\downarrow \text{too many FP and/or FN}

Worm propagation

\[\text{A} \rightarrow \text{Internet} \rightarrow \text{B}\]
Noise Injection Attack

Noise Injection Attack “poisons” the suspicious flow pool dataset with fake anomalous flows

Signature generator

Flow classifier → Worms + F.A.F. → Signature generation

Useless signatures

too many FP and/or FN

Worm propagation

A → Worm → B

Internet
Noise Injection Attack

- Noise Injection Attack “poisons” the suspicious flow pool dataset with fake anomalous flows.

Signature generator

Flow classifier → Worms + F.A.F. → Signature generation

Useless signatures:
- too many FP and/or FN

Fake anomalous flows

Worm propagation

A → Worm → Fake anomalous flows → B

Internet
Noise Injection Attack

- Noise Injection Attack “poisons” the suspicious flow pool dataset with fake anomalous flows

Signature generator

Flow classifier → Worms + F.A.F. → Signature generation

Fake anomalous flows do not need to exploit the vulnerability

Worm propagation

Useless signatures: too many FP and/or FN

Internet

Monday, September 6, 2010
Noise Injection Attack
Noise Injection Attack

Honeynet

Suspicious Flow Pool
Noise Injection Attack

Honeynet

[Diagram showing connections between Honeynet and Suspicious Flow Pool]

Suspicious Flow Pool
Noise Injection Attack

Honeynet

A

Suspicious Flow Pool
Noise Injection Attack

Honeynet

A

Suspicious Flow Pool
Noise Injection Attack

Honeynet

Suspicious Flow Pool
Noise Injection Attack

Honeynet

Double Honeynet

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Double Honeynet

A

Suspicious Flow Pool

GW

Layer-1

Suspicious Flow Pool

Layer-2

A
Noise Injection Attack

Honeynet

Double Honeynet

A

Suspicious Flow Pool

GW

Layer-1

Layer-2

Suspicious Flow Pool

A

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Double Honeynet

A

Suspicious Flow Pool

GW

Layer-2

A

Layer-1

Suspicious Flow Pool
Noise Injection Attack

Honeynet

Double Honeynet

GW

Layer-1

Suspicious Flow Pool

Suspicious Flow Pool

A

Layer-2
Noise Injection Attack

Honeynet

Double Honeynet

A
Suspicious Flow Pool

GW
A
Layer-1
Suspicious Flow Pool

Layer-2
Noise Injection Attack

Honeynet

Double Honeynet

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Double Honeynet

A

Suspicious Flow Pool

GW

Layer-1

Layer-2

A

Suspicious Flow Pool
Noise Injection Attack

Honeynet

Double Honeynet

A

Suspicous Flow Pool

A

GW

Layer-1

X

Layer-2

Suspicous Flow Pool

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Double Honeynet

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Double Honeynet

A
Suspicious Flow Pool

GW
Layer-1
Suspicious Flow Pool

Layer-2

A
Noise Injection Attack

Honeynet

Double Honeynet

Port-scanning detection

A
Suspicious Flow Pool

GW
Layer-1
X
Layer-2
Suspicious Flow Pool

A
Suspicious Flow Pool

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Port-scanning detection

Double Honeynet

GW

Layer-1

Layer-2
Noise Injection Attack

Honeynet

Double Honeynet

Port-scanning detection
Noise Injection Attack

Honeynet

Double Honeynet

Port-scanning detection

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Double Honeynet

Port-scanning detection
Noise Injection Attack

Honeynet

Double Honeynet

Port-scanning detection
Noise Injection Attack

Honeynet

Double Honeynet

Port-scanning detection

Anomaly IDS

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Double Honeynet

Port-scanning detection

Anomaly IDS

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Double Honeynet

Port-scanning detection

Anomaly IDS

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Double Honeynet

Port-scanning detection

Anomaly IDS

Monday, September 6, 2010
Noise Injection Attack

Honeynet

Suspicious Flow Pool

Double Honeynet

GW

Layer-2

Suspicious Flow Pool

Port-scanning detection

Anomaly IDS
Noise Injection Attack

Note:
1. The worm could send many fake anomalous flow (roughly) at the same time.
2. The worm variant and all its fake anomalous flows will be stored into the suspicious flow pool.

Monday, September 6, 2010
Case study: POLYGRAPH

- Signature generation for polymorphic worms

- The flow classification technique is not specified
- However, the authors assume that the flow classifier is not perfect and that innocuous flows (noise) could be stored into the suspicious pool
- **POLYGRAPH seems to be resilient to noise** into the suspicious pool (up to 80%)
- This is **not true if the noise is deliberately well-...**
Case study: POLYGRAPH

- Signature generation for polymorphic worms

- The flow classification technique is not specified
- However, the authors assume that the flow classifier is not perfect and that innocuous flows (noise) could be stored into the suspicious pool
- **POLYGRAPH seems to be resilient to noise** into the suspicious pool (up to 80%)
- This is **not true if the noise is deliberately well-**
Case study: POLYGRAPH

- Polygraph generates 3 different types of signatures:
  - Conjunction, Token-subsequence, Bayes

Conjunction Signature
\{PF, TI-1, TI-2, TI-3\}

Token-subsequence Signature
PF.*TI-1.*TI-3
Case study: POLYGRAPH

Conjunction and Token-subsequence signatures are not resilient to noise in the Suspicious flow pool.

Suspicious flow pool

- Worm A
- Innocuous flow
- Worm B
- Worm C

\[\{\text{without clustering}\}\]

= Conjunction Signature

= Token-subsequence Signature

Monday, September 6, 2010
Case study: POLYGRAPH

Conjunction and Token-subsequence signatures are not resilient to noise in the Suspicious flow pool.

- Without clustering
- Conjunction Signature
- Token-subsequence Signature
- Too many FP
- The signatures will be disregarded
Hierarchical Clustering

Suspicious flow pool

Clustering

Worm A

Innocuous flow

Worm B

Worm C
Hierarchical Clustering

Suspicious flow pool

Worm A

Innocuous flow

Worm B

Worm C

Clustering

Suspicious Cluster 1

Innocuous flow

Suspicious Cluster 2

Worm A

Worm B

Worm C

= Conjunction Signature

= Token-subsequence Signature

Monday, September 6, 2010
Hierarchical Clustering

Good signatures: can match **new worm varints**!

- = Conjunction Signature
- = Token-subsequence Signature
Misleading Conjunction and Token-Subsequence Signatures

Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives
Misleading Conjunction and Token-Subsequence Signatures

Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives
Misleading Conjunction and Token-Subsequence Signatures

Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives

- Worm body
- Protocol Framework
- True Invariants

Monday, September 6, 2010
Misleading Conjunction and Token-Subsequence Signatures

Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives

- Worm body
- Protocol Framework
- True Invariants
- Permuted bytes
Misleading Conjunction and Token-Subsequence Signatures

Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives

Worm
- Worm body
- Protocol Framework
- True Invariants

Fake anomalous flow
- Permuted bytes
- Fake Invariants
Misleading Conjunction and Token-Subsequence Signatures

Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives

- Worm body
- Protocol Framework
- True Invariants
- Permuted bytes
- Fake Invariants

Monday, September 6, 2010
Misleading Conjunction and Token-Subsequence Signatures

Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives
Misleading Conjunction and Token-Subsequence Signatures

- Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives

![Diagram showing Worm and Fake anomalous flow with labels for Worm body, Protocol Framework, True Invariants, Permuted bytes, and Fake Invariants]
Misleading Conjunction and Token-Subsequence Signatures

Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives

- Worm
- Fake anomalous flow

Legend:
- Worm body
- Protocol Framework
- True Invariants
- Permutated bytes
- Fake Invariants
Misleading Conjunction and Token-Subsequence Signatures

Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives

- Worm body
- Protocol Framework
- True Invariants
- Permuted bytes
- Fake Invariants

Monday, September 6, 2010
Objective: Mislead the Hierarchical Clustering algorithm so that signature will produce False Negatives

\[ P(FI \mid \text{innocuous flow}) < P(TI \mid \text{innocuous flow}) \]

\[ = \]

\[ P(\text{false positive} \mid \text{sig}(FI)) < P(\text{false positive} \mid \text{sig}(TI)) \]
Hierarchical Clustering

Suspicious flow pool

- Worm A
- FAF A
- Worm B
- FAF B
- Worm C
- FAF C

Clustering
Hierarchical Clustering
Hierarchical Clustering

Suspicious flow pool

Worm A
FAF A

Worm B
FAF B

Worm C
FAF C

Clustering

Worm A
FAF A

Worm B
FAF B

Worm C
FAF C

Conjunction Signature

Token-subsequence Signature

Monday, September 6, 2010
Hierarchical Clustering

Useless Signatures:
- The signatures do not contain the True Invariants
- Fake Invariants will not match new worm variants

The attack causes False Negatives!

\[
\begin{align*}
\text{• Conjunction Signature} & : \quad \text{ }
\text{• Token-subsequence Signature} & : \quad \text{ }
\end{align*}
\]
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j | \text{Suspicious Flow})$
  - $P_{if} = P(t_j | \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}$
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j | \text{Suspicious Flow})$
  - $P_{if} = P(t_j | \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$$\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}$$
Case study: POLYGRAH

**Bayes signatures:** All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

\[
\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}> < TI-3, \lambda_{TI-3}>\}
\]
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token \( t_j \)
  - \( P_{sf} = P(t_j \mid \text{Suspicious Flow}) \)
  - \( P_{if} = P(t_j \mid \text{Innocuous Flow}) \)
  - \( \lambda_j = \log \left( \frac{P_{sf}}{P_{if}} \right) \)

\{ <PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}> \}
Case study: POLYGRAH

**Bayes signatures:** All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$$\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>, \}$$
Case study: POLYGRAH

**Bayes signatures:** All the tokens common to at least $K$ out of the total number of suspicious flows $N$ are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$$\{<PF, \lambda_{PF}> , <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}$$
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token \( t_j \)
  - \( P_{sf} = P(t_j \mid \text{Suspicious Flow}) \)
  - \( P_{if} = P(t_j \mid \text{Innocuous Flow}) \)
  - \( \lambda_j = \log(P_{sf} / P_{if}) \)

\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$$\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>, \}$$
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}$
Case study: POLYGRAH

**Bayes signatures:** All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$$\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}$$
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least $K$ out of the total number of suspicious flows $N$ are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j | \text{Suspicious Flow})$
  - $P_{if} = P(t_j | \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

\{
\langle PF, \lambda_{PF} \rangle, \langle TI-1, \lambda_{TI-1} \rangle, \langle TI-2, \lambda_{TI-2} \rangle, \langle TI-3, \lambda_{TI-3} \rangle \}\}

If $\Lambda = \sum_k \lambda_k > \theta$

The flow is a worm!
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

\[\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}\]
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>, \}$
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j | \text{Suspicious Flow})$
  - $P_{if} = P(t_j | \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j | \text{Suspicious Flow})$
  - $P_{if} = P(t_j | \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}
Case study: POLYGRAH

**Bayes signatures**: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

\[
\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>)\}
\]
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j \mid \text{Suspicious Flow})$
  - $P_{if} = P(t_j \mid \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}$
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j | \text{Suspicious Flow})$
  - $P_{if} = P(t_j | \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

$\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}>, <TI-3, \lambda_{TI-3}>\}$
Case study: POLYGRAH

Bayes signatures: All the tokens common to at least K out of the total number of suspicious flows N are extracted

- For each token $t_j$
  - $P_{sf} = P(t_j | \text{Suspicious Flow})$
  - $P_{if} = P(t_j | \text{Innocuous Flow})$
  - $\lambda_j = \log(P_{sf} / P_{if})$

\[
\{<PF, \lambda_{PF}>, <TI-1, \lambda_{TI-1}>, <TI-2, \lambda_{TI-2}> < TI-3, \lambda_{TI-3}>\}
\]

\[\Lambda = \sum_k \lambda_k < \theta\]

If $\Lambda < \theta$ then

The flow is Innocuous!
Consider a string \( \nu \) of length \( n \) that is present in the Innocuous pool with probability

\[
0.05 < P(\nu | \text{Innocuous Flow}) < 0.20
\]

If all the fake anomalous flows contain \( \nu \), the string will be present in 50\% of the suspicious flows.

Thus, the extracted signature will contain \( \nu \) and the related score \( \lambda_{\nu} \) will be

\[
\log(0.5/0.20) < \lambda_{\nu} < \log(0.5/0.05)
\]

This means that an innocuous flow containing \( \nu \) will receive a total score \( \Lambda \geq \lambda_{\nu} \).
Misleading Bayes Signatures

- E.g., $\nu$ = “Pragma: no-cache”
  - Suppose $P(\nu \mid \text{Innocuous Flow}) = 0.094$
  - $\lambda_\nu = \log(0.5/0.094)$

- Suppose signature is
  - $<$GET,$\lambda_1$>; $<$HTTP/1.1,$\lambda_2$>; ... ; $<$Pragma: no-cache,$\lambda_\nu$>$>$

- Given an Innocuous flow:
  - GET .*$HTTP/1.1 .*$Pragma: no-cache .*$
  - $\Lambda = \sum_k \lambda_k + \lambda_\nu \geq \lambda_\nu$
Misleading Bayes Signatures

- E.g., $\nu = \text{"Pragma: no-cache"}$
  - Suppose $P(\nu \mid \text{Innocuous Flow}) = 0.094$
  - $\lambda_\nu = \log(0.5/0.094)$

- Suppose signature is
  - $\langle \text{GET}, \lambda_1 \rangle; \langle \text{HTTP/1.1}, \lambda_2 \rangle; \ldots ; \langle \text{Pragma: no-cache}, \lambda_\nu \rangle$

- Given an Innocuous flow:
  - $\Lambda = \Sigma_k \lambda_k + \lambda_\nu \geq \lambda_\nu$
Misleading Bayes Signatures

- E.g., $\nu = \text{"Pragma: no-cache"}$
  - Suppose $P(\nu \mid \text{Innocuous Flow}) = 0.094$
  - $\lambda_\nu = \log(0.5/0.094)$

- Suppose signature is
  - $\langle \text{GET}, \lambda_1 \rangle; \langle \text{HTTP/1.1}, \lambda_2 \rangle; \ldots; \langle \text{Pragma: no-cache}, \lambda_\nu \rangle$

- Given an Innocuous flow:
  - $\Lambda = \sum_k \lambda_k + \lambda_\nu \geq \lambda_\nu$
Misleading Bayes Signatures

- E.g., $\nu$ = “Pragma: no-cache”
  - Suppose $P(\nu \mid \text{Innocuous Flow}) = 0.094$
  - $\lambda_{\nu} = \log(0.5/0.094)$

- Suppose signature is
  - $<\text{GET},\lambda_{1}>; <\text{HTTP/1.1},\lambda_{2}>; \ldots; <\text{Pragma: no-cache},\lambda_{\nu}>;$

- Given an Innocuous flow:
  - $\Lambda = \lambda_{1} + \lambda_{2} + \lambda_{\nu}$
  - $\Lambda = \sum_{k} \lambda_{k} + \lambda_{\nu}$

  If $\Lambda < \theta$ then No false positives
“Score multiplier” effect

Spliting \( \nu \) into all the substrings of length \( m < n \)
- E.g., “Pragma: no”, “pragma: no-”, “agma: no-c”, etc.
- If \(! (m < n)\) then \( p(\nu_{i,i+m} | IF) \approx p(\nu | IF), \forall i\)

- Injecting all the substring \( \nu_{i,i+m} \) of \( \nu \) into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
  - \( \lambda_{\nu_{i,i+m}} \approx \lambda_\nu \)

- An innocuous flow that contains \( \nu \) contains also all the \( \nu_{i,i+m} \) tokens

- A **score multiplier effect** is obtained for the innocuous flows which contain \( \nu \)
  - GET .* HTTP/1.1 .* Pragma: no-cache .*
  - \( \Lambda = \Sigma_k \lambda_k + \Sigma_i \lambda_{\nu_i} \gg \lambda_\nu \)
“Score multiplier” effect

Spliting $\nu$ into all the substrings of length $m < n$
  - E.g., “Pragma: no”, “Pragma: no-”, “Pragma: no-c”, etc.
  - If !(m<<n) then $p(\nu_{i,i+m} \mid IF) \sim= p(\nu \mid IF), \forall i$

Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
  - $\lambda_{\nu_{i,i+m}} \sim= \lambda_{\nu}$

An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

A **score multiplier effect** is obtained for the innocuous flows which contain $\nu$
  - $\Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_{\nu}$

```
GET .* HTTP/1.1 .* Pragma: no-cache .*
```

Monday, September 6, 2010
“Score multiplier” effect

Spliting $\nu$ into all the substrings of length $m<n$
- E.g., “Pragma: no”, “pragma: no-”, “agma: no-c”, etc.
- If !(m<<n) then $p(\nu_{i,i+m} | \text{IF}) \sim p(\nu | \text{IF}), \forall i$

Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
- $\lambda_{\nu_{i,i+m}} \sim \lambda_{\nu}$

An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

A **score multiplier effect** is obtained for the innocuous flows which contain $\nu$

- GET .* HTTP/1.1 .* Pragma: no-cache .*
- $\Lambda = \Sigma_k \lambda_k + \Sigma_i \lambda_{\nu_i} \gg \lambda_{\nu}$
“Score multiplier” effect

Spliting $\nu$ into all the substrings of length $m<n$

- E.g., “Pragma: no”, “ pragma: no-”, “agma: no-c”, etc.
- If !(m<<n) then $p(\nu_{i,i+m} | IF) \sim p(\nu | IF)$, $\forall i$

- Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)

- $\lambda_{\nu_{i,i+m}} \sim= \lambda_\nu$

- An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

- A **score multiplier effect** is obtained for the innocuous flows which contain $\nu$

  - $\Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_\nu$

Monday, September 6, 2010
“Score multiplier” effect

- Spliting $\nu$ into all the substrings of length $m<n$
  - E.g., “Pragma: no”, “ragma: no-”, “agma: no-c”, etc.
  - If !(m<<n) then $p(\nu_{i,i+m} | \text{IF}) \sim p(\nu | \text{IF}), \forall i$

- Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
  - $\lambda_{\nu_{i,i+m}} \sim \lambda_{\nu}$
  - An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

- A score multiplier effect is obtained for the innocuous flows which contain $\nu$
  - GET .* HTTP/1.1 .* Pragma: no-cache .*
  - $\Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_{\nu}$

Monday, September 6, 2010
“Score multiplier” effect

Spliting ν into all the substrings of length m<n

- E.g., “Pragma: no”, “pragma: no-”, “agama: no-c”, etc.
- If !(m<n) then p(νi,i+m | IF) ~ p(ν | IF), ∀i

Injecting all the substring νi,i+m of ν into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)

- λνi,i+m ~ λν

An innocuous flow that contains ν contains also all the νi,i+m tokens

A score multiplier effect is obtained for the innocuous flows which contain ν

- GET .* HTTP/1.1 .* Pragma: no-cache .* 
- Λ = Σk λk + Σi λνi >> λν
“Score multiplier” effect

Spliting $\nu$ into all the substrings of length $m<n$

- E.g., “Pragma: no”, “pragma: no-”, “agama: no-c”, etc.
- If !(m<<n) then $p(\nu_{i,i+m} \mid \text{IF}) \sim p(\nu \mid \text{IF}), \forall i$

- Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)

- $\lambda_{\nu_{i,i+m}} \sim= \lambda_{\nu}$

- An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

- A score multiplier effect is obtained for the innocuous flows which contain $\nu$

- $\Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_{\nu}$
"Score multiplier" effect

- Spliting \( \nu \) into all the substrings of length \( m < n \)
  - E.g., “Pragma: no”, “ragma: no-”, “agma: no-c”, etc.
  - If \( !(m < n) \) then \( p(\nu_{i,i+m} | IF) \sim p(\nu | IF), \forall i \)

- Injecting all the substring \( \nu_{i,i+m} \) of \( \nu \) into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
  - \( \lambda_{\nu_{i,i+m}} \sim= \lambda_{\nu} \)

- An innocuous flow that contains \( \nu \) contains also all the \( \nu_{i,i+m} \) tokens

- A **score multiplier effect** is obtained for the innocuous flows which contain \( \nu \)
  - \( \Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_{\nu} \)
“Score multiplier” effect

Spliting $\nu$ into all the substrings of length $m<n$

- E.g., “Pragma: no”, “pragma: no-”, “agma: no-c”, etc.
- If !(m<<n) then $p(\nu_{i,i+m} | IF) \sim p(\nu | IF), \forall i$

Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)

- $\lambda_{\nu_{i,i+m}} \sim= \lambda_{\nu}$

- An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

- A **score multiplier effect** is obtained for the innocuous flows which contain $\nu$

- \[ \Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_{\nu} \]
“Score multiplier” effect

Spliting $\nu$ into all the substrings of length $m<n$
- E.g., “Pragma: no”, “pragma: no-”, “agma: no-c”, etc.
- If $!(m<\nu n)$ then $p(\nu_{i,i+m} \mid \text{IF}) \sim p(\nu \mid \text{IF}), \forall i$

Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
- $\lambda_{\nu_{i,i+m}} \sim \lambda_{\nu}$

An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

A **score multiplier effect** is obtained for the innocuous flows which contain $\nu$
- $\Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_{\nu}$
“Score multiplier” effect

Spliting \( \nu \) into all the substrings of length \( m < n \)
- E.g., “Pragma: no”, “pragma: no-”, “agma: no-c”, etc.
- If \(! (m << n)\) then \( p(\nu_{i,i+m} \mid \text{IF}) \approx p(\nu \mid \text{IF}), \forall i\)

Injecting all the substring \( \nu_{i,i+m} \) of \( \nu \) into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
- \( \lambda_{\nu_{i,i+m}} \approx \lambda_{\nu} \)

An innocuous flow that contains \( \nu \) contains also all the \( \nu_{i,i+m} \) tokens

A **score multiplier effect** is obtained for the innocuous flows which contain \( \nu \)
- \( \Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_{i,i+m}} \gg \lambda_{\nu} \)
“Score multiplier” effect

Spliting \( \nu \) into all the substrings of length \( m<n \)
- E.g., “Pragma: no”, “pragma: no-”, “agma: no-c”, etc.
- If \( !(m<n) \) then \( p(\nu_{i,i+m} \mid \text{IF}) \sim p(\nu \mid \text{IF}), \forall i \)

Injecting all the substring \( \nu_{i,i+m} \) of \( \nu \) into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
- \( \lambda_{\nu_{i,i+m}} \sim \lambda_{\nu} \)
- An innocuous flow that contains \( \nu \) contains also all the \( \nu_{i,i+m} \) tokens
- A **score multiplier effect** is obtained for the innocuous flows which contain \( \nu \)

\[
\Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_{\nu}
\]
“Score multiplier” effect

Spliting ν into all the substrings of length m<n
- E.g., “Pragma: no”, “pragma: no-”, “agma: no-c”, etc.
- If !(m<<n) then p(ν_{i,i+m} | IF) \sim p(ν | IF), ∀i

Injecting all the substring ν_{i,i+m} of ν into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
- λ_{ν_{i,i+m}} \sim λ_{ν}

An innocuous flow that contains ν contains also all the ν_{i,i+m} tokens

A score multiplier effect is obtained for the innocuous flows which contain ν
- \textbf{GET} .* \textbf{HTTP/1.1} .* Pragma: no-cache .*
- Λ = \sum_k λ_k + \sum_i λ_{ν_{i,i+m}} \gg λ_{ν}
“Score multiplier” effect

Spliting $\nu$ into all the substrings of length $m<n$
- E.g., “Pragma: no”, “ragma: no-”, “agma: no-c”, etc.
- If !(m<<n) then $p(\nu_{i,i+m} | IF) \sim p(\nu | IF), \forall i$

Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
- $\lambda_{\nu_{i,i+m}} \sim \lambda_{\nu}$
- An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

A **score multiplier effect** is obtained for the innocuous flows which contain $\nu$
- $\Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_{i}} >> \lambda_{\nu}$
“Score multiplier” effect

Spliting $\nu$ into all the substrings of length $m<n$

- E.g., “Pragma: no”, “pragma: no-”, “agama: no-c”, etc.
- If !(m<<n) then $p(\nu_{i,i+m} | IF) \sim= p(\nu | IF), \forall i$

- Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
  - $\lambda_{\nu_{i,i+m}} \sim= \lambda_{\nu}$

- An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

- A score multiplier effect is obtained for the innocuous flows which contain $\nu$
  - $\Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_{\nu}$
“Score multiplier” effect

Spliting $\nu$ into all the substrings of length $m<n$
- E.g., “Pragma: no”, “rgba: no-”, “agma: no-c”, etc.
- If $!(m<<n)$ then $p(\nu_{i,i+m} \mid IF) \sim p(\nu \mid IF), \forall i$

- Injecting all the substring $\nu_{i,i+m}$ of $\nu$ into the fake anomalous flows, they will be all considered as tokens in the signature (50% occurrence freq.)
  - $\lambda_{\nu_{i,i+m}} \sim \lambda_{\nu}$

- An innocuous flow that contains $\nu$ contains also all the $\nu_{i,i+m}$ tokens

- A **score multiplier effect** is obtained for the innocuous flows which contain $\nu$
  - $\Lambda = \sum_k \lambda_k + \sum_i \lambda_{\nu_i} \gg \lambda_{\nu}$
  - If $\Lambda > \theta$ \quad **False positive**

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    
    $0.05 < p(\nu \mid \text{“normal” flow}) < 0.20$
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$
Crafting the Noise

- Find candidate strings \( \nu \) to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings \( \nu \) of length between \( l_1 \) and \( l_2 \) whereby
    \[
    0.05 < p(\nu \mid \text{“normal” flow}) < 0.20
    \]
  - Split \( \nu \) into the score multiplier substrings \( \nu_{i,i+m} \)
Crafting the Noise

- Find candidate strings $\nu$ to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    $$0.05 < p(\nu \mid \text{"normal" flow}) < 0.20$$
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

Worm

Fake anom. flow

- Worm body
- Protocol framework
- True Invariant

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce **score multiplier effect**
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby $0.05 < p(\nu \mid \text{“normal” flow}) < 0.20$
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

- Worm body
- Protocol framework
- True Invariant
- Permuted bytes
Crafting the Noise

- Find candidate strings $\nu$ to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    \[ 0.05 < p(\nu | \text{“normal” flow}) < 0.20 \]
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

Worm

Fake anom. flow

<table>
<thead>
<tr>
<th>Worm body</th>
<th>Protocol framework</th>
<th>True Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permuted bytes</td>
<td>Fake Invariant</td>
<td></td>
</tr>
</tbody>
</table>

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    $0.05 < p(\nu \mid \text{“normal” flow}) < 0.20$
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$
Crafting the Noise

- Find candidate strings $\nu$ to produce **score multiplier effect**
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    \[ 0.05 < p(\nu \mid \text{“normal” flow}) < 0.20 \]
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

- Worm
  - Worm body
  - Protocol framework
  - True Invariant

- Fake anom. flow
  - Permuted bytes
  - Fake Invariant

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    $$0.05 < p(\nu \mid \text{“normal” flow}) < 0.20$$
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

- Worm body
- Protocol framework
- True Invariant
- Permuted bytes
- Fake Invariant

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce **score multiplier effect**
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    \[ 0.05 < p(\nu \mid \text{“normal” flow}) < 0.20 \]
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

Worm

Fake anom. flow

- True Invariant
- Protocol framework
- Worm body
- Permuted bytes
- Fake Invariant

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    \[0.05 < p(\nu \mid \text{"normal" flow}) < 0.20\]
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce **score multiplier effect**
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    \[
    0.05 < p(\nu \mid \text{"normal" flow}) < 0.20
    \]
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

![Diagram showing worm and fake anom. flow with various segments colored differently to represent Worm body, Protocol framework, True Invariant, Permuted bytes, Fake Invariant, and Score multip. strings.]

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    \[ 0.05 < p(\nu | \text{"normal" flow}) < 0.20 \]
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$
Crafting the Noise

- Find candidate strings $\nu$ to produce **score multiplier effect**
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    \[ 0.05 < p(\nu \mid \text{“normal” flow}) < 0.20 \]
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

---

<table>
<thead>
<tr>
<th>Worm body</th>
<th>Protocol framework</th>
<th>True Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permuted bytes</td>
<td>Fake Invariant</td>
<td>Score multip. strings</td>
</tr>
</tbody>
</table>

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    \[ 0.05 < p(\nu \mid \text{"normal" flow}) < 0.20 \]
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

Monday, September 6, 2010
Crafting the Noise

- Find candidate strings $\nu$ to produce score multiplier effect
  - Analyze normal traffic samples
  - Look for all the strings $\nu$ of length between $l_1$ and $l_2$ whereby
    \[0.05 < p(\nu \mid \text{“normal” flow}) < 0.20\]
  - Split $\nu$ into the score multiplier substrings $\nu_{i,i+m}$

- The Fake Invariants are specific for each worm and its fake anomalous flows
- The score multiplier strings have to be common to all the fake anomalous flows
Experimental results

- **Experimental Setup**
  - We implemented POLYGRAPH according to the description in [1]
  - “Apache-Knacker” Worm:
    ```
    GET .* HTTP/1.1\r\n.*\r\nHost: .*\r\n.*\r\nHost: .*\xFF\xBF.*\r\n    ```

- **Training dataset**
  - Suspicious flow pool = 10 worm variants
  - Innocuous flow pool = 100,459 flows (0.007% FP)

- **Test dataset**
  - “Normal” Test flow pool = 217,164 (0.0% FP)
  - “Suspicious” Test flow pool = 100 worm variants

- **Attacker’s dataset**
  - Candidate Score Multiplier Strings extracted from 5,000 flows
Experimental results with Bayes signatures

Score Multip. Srings (m=4): “Pragma: no-cache”, “-powerpoint”
Experimental results with Bayes signatures

Score Multipl. Strings (m=4): “Pragma: no-cache”, “-powerpoint”

Under attack it is impossible to find a threshold that produces high Detection Rate and low False Positive Rate
Experimental results with all the 3 types of signatures

- The results are not deterministically predictable (due to tokens that are common just by chance)
  - Simulations: 2 groups of tests
    - For each group of tests we simulated 2 scenarios:
      - 1 faf/worm and 2 faf/worm
      - 1° group of tests: 45 rounds
      - 2° group of tests: 20 rounds

<table>
<thead>
<tr>
<th></th>
<th>1 fake anomalous flow</th>
<th>2 fake anomalous flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunction</td>
<td>73.3%</td>
<td>88.9%</td>
</tr>
<tr>
<td>Token-subsequences</td>
<td>60.0%</td>
<td>73.3%</td>
</tr>
<tr>
<td>Bayes</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>All three signatures</td>
<td>44.4%</td>
<td>62.2%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1 fake anomalous flow</th>
<th>2 fake anomalous flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunction</td>
<td>65%</td>
<td>95%</td>
</tr>
<tr>
<td>Token-subsequences</td>
<td>40%</td>
<td>90%</td>
</tr>
<tr>
<td>Bayes</td>
<td>90%</td>
<td>100%</td>
</tr>
<tr>
<td>All three signatures</td>
<td>20%</td>
<td>85%</td>
</tr>
</tbody>
</table>

Table 1: Percentage of successful attacks (using “Forwarded-For” and “Modified-Since”)
Table 2: Percentage of successful attacks (using “Cache-Control” and “Range: bytes”)

Monday, September 6, 2010
Experimental results with all the 3 types of signatures

- The results are not deterministically predictable (due to tokens that are common just by chance)
  - Simulations: 2 groups of tests
    - For each group of tests we simulated 2 scenarios:
      - 1 faf/worm and 2 faf/worm
      - 1° group of tests: 45 rounds
      - 2° group of tests: 20 rounds

<table>
<thead>
<tr>
<th></th>
<th>1 fake anomalous flow</th>
<th>2 fake anomalous flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunction</td>
<td>73.3%</td>
<td>88.9%</td>
</tr>
<tr>
<td>Token-subsequences</td>
<td>60.0%</td>
<td>73.3%</td>
</tr>
<tr>
<td>Bayes</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>All three signatures</td>
<td>44.4%</td>
<td>62.2%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1 fake anomalous flow</th>
<th>2 fake anomalous flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunction</td>
<td>65%</td>
<td>95%</td>
</tr>
<tr>
<td>Token-subsequences</td>
<td>40%</td>
<td>90%</td>
</tr>
<tr>
<td>Bayes</td>
<td>90%</td>
<td>100%</td>
</tr>
<tr>
<td>All three signatures</td>
<td>20%</td>
<td>85%</td>
</tr>
</tbody>
</table>

Table 1: Percentage of successful attacks (using “Forwarded-For” and “Modified-Since”)

Table 2: Percentage of successful attacks (using “Cache-Control” and “Range: bytes”)
Conclusion

- Can Machine Learning be secure? [2]
- The Noise Injection Attack has a good chance to mislead the signature generation process
- Unless a precise semantic-based (thus expensive) flow classifier + signature generation scheme is used
  - Syntactic-based Worm Signature Generators are vulnerable to the Noise Injection Attack

Can Machine Learning be Secure? (ASIACCS 2006)
Thank you!
Combining our attack with the Red Herring

- **Increases the probability of success**
  - The worm includes some **temporay invariants**
  - This invariants **expire over time**
  - This means that even if POLYGRAPH generates useful Conjunction and Token-subsequence signatures, after a while they will become useless
  - The second time POLYGRAPH generates the signature, it could be not as “fortunate” as the first time
  - Further, if the temporary invariants are chosen among “high frequency normal tokens”, the combination of the attacks **will not interphere with the attack to Bayes Signatures**
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow

- $P(FP \mid FI) < P(FP \mid TI)$

- $PF = $ Protocol Framework
- $TI = $ True Invariant
- $FI = $ Fake Invariant

Worm

Fake anomalous flow
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = \text{i-th Worm variant}$
- $F_i = \text{i-th Fake anomalous flow}$

- $P(\text{FP} | F_i) < P(\text{FP} | T_i)$

- $PF = \text{Protocol Framework}$
- $T_i = \text{True Invariant}$
- $F_i = \text{Fake Invariant}$

<table>
<thead>
<tr>
<th>W1</th>
<th>F1</th>
<th>W2</th>
<th>F2</th>
<th>W3</th>
<th>F3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fake anomalous flow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Misleading Conjunction and Token-Subsequence Signatures

- $W_i =$ i-th Worm variant
- $F_i =$ i-th Fake anomalous flow

- $P(FP \mid FI) < P(FP \mid TI)$

- $PF =$ Protocol Framework
- $TI =$ True Invariant
- $FI =$ Fake Invariant

\[ \text{Worm} \]
\[ \text{Fake anomalous flow} \]
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = \text{i-th Worm variant}$
- $F_i = \text{i-th Fake anomalous flow}$

- $P(\text{FP} | \text{FI}) < P(\text{FP} | \text{TI})$

- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$

\[
\begin{array}{cccccc}
W_1 & F_1 & W_2 & F_2 & W_3 & F_3 \\
\text{PF + FI-1} & \text{PF + TI}
\end{array}
\]
Misleading Conjunction and Token-Subsequence Signatures

- \( W_i = \) i-th Worm variant
- \( F_i = \) i-th Fake anomalous flow

\[
P(FP | FI) < P(FP | TI)
\]

- \( PF = \) Protocol Framework
- \( TI = \) True Invariant
- \( FI = \) Fake Invariant

\[\begin{array}{cccccc}
W_1 & F_1 & W_2 & F_2 & W_3 & F_3 \\
\text{PF + } FI-1 & PF + TI & PF \\
\end{array}\]
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP \mid FI) < P(FP \mid TI)$

- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$

**Diagram:***

- Worm
- Fake anomalous flow

\[ W_1 \quad F_1 \quad W_2 \quad F_2 \quad W_3 \quad F_3 \]

- $PF + FI_{-1}$
- $PF + TI$
- $PF$
- $PF + TI$
Misleading Conjunction and Token-Subsequence Signatures

- Wi = i-th Worm variant
- Fi = i-th Fake anomalous flow
- P(FP | FI) < P(FP | TI)

- PF = Protocol Framework
- TI = True Invariant
- FI = Fake Invariant

PF + FI-1

PF + TI

PF

PF + TI

PF
Misleading Conjunction and Token-Subsequence Signatures

- \( Wi = i \)-th Worm variant
- \( Fi = i \)-th Fake anomalous flow
- \( P(FP | FI) < P(FP | TI) \)
- \( PF = \) Protocol Framework
- \( TI = \) True Invariant
- \( FI = \) Fake Invariant

Diagram:

- Worm: \( W1, W2, W3 \)
- Fake anomalous flow: \( F1, F2, F3 \)

PF = Protocol Framework
TI = True Invariant
FI = Fake Invariant
Misleading Conjunction and Token-Subsequence Signatures

- \( W_i = i\)-th Worm variant
- \( F_i = i\)-th Fake anomalous flow
- \( P(FP \mid FI) < P(FP \mid TI) \)
- \( PF = \) Protocol Framework
- \( TI = \) True Invariant
- \( FI = \) Fake Invariant

PF + FI-1
PF + TI
PF
PF + TI
PF
PF
PF
PF + FI-2

Worm
Fake anomalous flow
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow

- $P(F_P | F_I) < P(F_P | T_I)$

- $PF = $ Protocol Framework
- $T_I = $ True Invariant
- $F_I = $ Fake Invariant

Worm

Fake anomalous flow
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow

$P(FP \mid FI) < P(FP \mid TI)$

- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$

\[\begin{array}{ccccccc}
W1 & F1 & W2 & F2 & W3 & F3 \\
& PF + FI-1 & & PF + TI & & \\
& PF & & PF + TI & & \\
& PF & & PF & & \\
& PF & & PF & & \\
& PF & & PF & & \\
& PF + FI-2 & & PF + TI & & \\
\end{array}\]
Misleading Conjunction and Token-Subsequence Signatures

- Wi = i-th Worm variant
- Fi = i-th Fake anomalous flow
- P(FP | FI) < P(FP | TI)
- PF = Protocol Framework
- TI = True Invariant
- FI = Fake Invariant

Worm

Fake anomalous flow
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP \mid FI) < P(FP \mid TI)$

- $PF = $ Protocol Framework
- $TI = $ True Invariant
- $FI = $ Fake Invariant

\[
\begin{array}{ccccccc}
W1 & F1 & W2 & F2 & W3 & F3 \\
\mid \quad PF + FI-1, \\
\mid \quad PF + TI \\
\mid \quad PF \\
\mid \quad PF + TI \\
\mid \quad PF \\
\mid \quad PF + FI-2 \\
\mid \quad PF + TI \\
\mid \quad PF \\
\mid \quad PF \\
\mid \quad PF + FI-3 \\
\end{array}
\]
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = \text{i-th Worm variant}$
- $F_i = \text{i-th Fake anomalous flow}$

- $P(F_P | F_I) < P(F_P | T_I)$

- $PF = \text{Protocol Framework}$
- $T_I = \text{True Invariant}$
- $F_I = \text{Fake Invariant}$

Worm

Fake anomalous flow
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP | F_i) < P(FP | T_I)$

- $PF = $ Protocol Framework
- $T_I = $ True Invariant
- $F_I = $ Fake Invariant
Misleading Conjunction and Token-Subsequence Signatures

- \( Wi = i\)-th Worm variant
- \( Fi = i\)-th Fake anomalous flow
- \( P(FP | FI) < P(FP | TI) \)
- \( PF = \) Protocol Framework
- \( TI = \) True Invariant
- \( FI = \) Fake Invariant

Worm

Fake anomalous flow

Monday, September 6, 2010
Misleading Conjunction and Token-Subsequence Signatures

- $W_i =$ i-th Worm variant
- $F_i =$ i-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$

- $PF =$ Protocol Framework
- $TI =$ True Invariant
- $FI =$ Fake Invariant

Worm

Fake anomalous flow

Monday, September 6, 2010
Misleading Conjunction and Token-Subsequence Signatures

- \( W_i = \) i-th Worm variant
- \( F_i = \) i-th Fake anomalous flow

- \( P(\text{FP} | \text{FI}) < P(\text{FP} | \text{TI}) \)

- \( \text{PF} = \) Protocol Framework
- \( \text{TI} = \) True Invariant
- \( \text{FI} = \) Fake Invariant
Misleading Conjunction and Token-Subsequence Signatures

- $W_i$ = $i$-th Worm variant
- $F_i$ = $i$-th Fake anomalous flow

- $P(FP \mid FI) < P(FP \mid TI)$

- PF = Protocol Framework
- TI = True Invariant
- FI = Fake Invariant
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = \text{i-th Worm variant}$
- $F_i = \text{i-th Fake anomalous flow}$
- $P(FP | FI) < P(FP | TI)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
Misleading Conjunction and Token-Subsequence Signatures

- $W_i =$ i-th Worm variant
- $F_i =$ i-th Fake anomalous flow
- $P(FP \mid FI) < P(FP \mid TI)$
- $PF =$ Protocol Framework
- $TI =$ True Invariant
- $FI =$ Fake Invariant

\[ [W_1, F_1] \quad W_2 \quad F_2 \quad W_3 \quad F_3 \]

\[
\begin{array}{c}
\text{PF} \\
\text{PF} \\
\text{PF} \\
\text{PF} \\
\text{PF + FI-2}
\end{array}
\]
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$

\[ [W_1, F_1] \quad W_2 \quad F_2 \quad W_3 \quad F_3 \]

- $PF$  
- $PF + FI-2$  
- $PF + TI$

Worm

Fake anomalous flow
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow

- $P(FP \mid FI) < P(FP \mid TI)$

- $PF = $ Protocol Framework
- $TI = $ True Invariant
- $FI = $ Fake Invariant
Misleading Conjunction and Token-Subsequence Signatures

- \( W_i \) = i-th Worm variant
- \( F_i \) = i-th Fake anomalous flow
- \( P(FP \mid FI) < P(FP \mid TI) \)
- \( PF \) = Protocol Framework
- \( TI \) = True Invariant
- \( FI \) = Fake Invariant
Misleading Conjunction and Token-Subsequence Signatures

- $W_i$ = i-th Worm variant
- $F_i$ = i-th Fake anomalous flow
- $P(FP \mid FI) < P(FP \mid TI)$
- PF = Protocol Framework
- TI = True Invariant
- FI = Fake Invariant

\[ [W_1, F_1] \quad W_2 \quad F_2 \quad W_3 \quad F_3 \]

\[ \begin{array}{c}
\text{Worm} \\
\text{Fake anomalous flow}
\end{array} \]
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(F_P | F_I) < P(F_P | T_I)$
- $P(F_P) = $ Protocol Framework
- $T_I = $ True Invariant
- $F_I = $ Fake Invariant

$[W_1, F_1]$  $W_2$  $F_2$  $W_3$  $F_3$

- $PF$
- $PF$
- $PF$
- $PF$
- $PF$
- $PF$
- $PF$
- $PF$
- $PF$
- $PF$

$W_1$
$W_2$
$W_3$

$F_1$
$F_2$
$F_3$

Worm
Fake anomalous flow

Monday, September 6, 2010
Misleading Conjunction and Token-Subsequence Signatures

- \( W_i = \text{i-th Worm variant} \)
- \( F_i = \text{i-th Fake anomalous flow} \)
- \( P(FP | FI) < P(FP | TI) \)
- \( PF = \text{Protocol Framework} \)
- \( TI = \text{True Invariant} \)
- \( FI = \text{Fake Invariant} \)

\[
\begin{array}{cccccc}
W_1 & F_1 & W_2 & F_2 & W_3 & F_3 \\
\text{PF} & \text{PF} & \text{PF} & \text{PF} & \text{PF} & \text{PF} \\
\text{PF} & \text{PF} & \text{PF} & \text{PF} & \text{PF} & \text{PF} \\
\text{PF} + \text{FI-2} & \text{PF} + \text{TI} & \text{PF} & \text{PF} & \text{PF} + \text{FI-3} \\
\end{array}
\]
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$

$W_i = \text{Worm variant}$
$F_i = \text{Fake anomalous flow}$

$P(FP | FI) < P(FP | TI)$

$PF = \text{Protocol Framework}$
$TI = \text{True Invariant}$
$FI = \text{Fake Invariant}$
Misleading Conjunction and Token-Subsequence Signatures

- $W_i$ = i-th Worm variant
- $F_i$ = i-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$
- $PF$ = Protocol Framework
- $TI$ = True Invariant
- $FI$ = Fake Invariant

$[W_1, F_1]$ $W_2$ $F_2$ $W_3$ $F_3$

- $PF$
- $PF$
- $PF$
- $PF$
- $PF$
- $PF$
- $PF$
- $PF$
- $PF$

- $W_i$ = i-th Worm variant
- $F_i$ = i-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$
- $PF$ = Protocol Framework
- $TI$ = True Invariant
- $FI$ = Fake Invariant
Misleading Conjunction and Token-Subsequence Signatures

- $W_i =$ i-th Worm variant
- $F_i =$ i-th Fake anomalous flow
- $P(F_P | F_I) < P(F_P | T_I)$
- $P(F_P) = \text{Protocol Framework}$
- $T_I = \text{True Invariant}$
- $F_I = \text{Fake Invariant}$

Wi = i-th Worm variant
Fi = i-th Fake anomalous flow

PF = Protocol Framework
TI = True Invariant
FI = Fake Invariant
Misleading Conjunction and Token-Subsequence Signatures

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
Misleading Conjunction and Token-Subsequence Signatures

- $W_i$ = i-th Worm variant
- $F_i$ = i-th Fake anomalous flow
- $P(F_P | F_I) < P(F_P | T_I)$
- $P(F_P | F_I)$ = Protocol Framework
- $T_I$ = True Invariant
- $F_I$ = Fake Invariant

Min num of flows = 3 → NO SIGNATURE!

[ W1, F1 ] [ W2, F2 ] [ W3, F3 ]
The results are not Deterministically Predictable

- $W_i = \text{i-th Worm variant}$
- $F_i = \text{i-th Fake anomalous flow}$

- $P(FP | F_i) < P(FP | T_i)$

- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
- $C = \text{Common Token by chance!}$

Worm

Fake anomalous flow
The results are not Deterministically Predictable

- $W_i =$ i-th Worm variant
- $F_i =$ i-th Fake anomalous flow
- $P(FP \mid FI) < P(FP \mid TI)$
- $PF =$ Protocol Framework
- $TI =$ True Invariant
- $FI =$ Fake Invariant
- $C =$ Common Token by chance!

Worm
Fake anomalous flow
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow

- $P(F_P | F_I) < P(F_P | T_I)$

- $PF =$ Protocol Framework
- $T_I =$ True Invariant
- $F_I =$ Fake Invariant
- $C =$ Common Token by chance!

---

Monday, September 6, 2010
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP \mid FI) < P(FP \mid TI)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
- $C = \text{Common Token by chance!}$

$W_1 \quad F_1 \quad W_2 \quad F_2 \quad W_3 \quad F_3$

$PF + F_{I-1}$

$PF + TI + C$
The results are not Deterministically Predictable

- $W_i =$ i-th Worm variant
- $F_i =$ i-th Fake anomalous flow

- $P(FP | FI) < P(FP | TI)$

- $PF =$ Protocol Framework
- $TI =$ True Invariant
- $FI =$ Fake Invariant

$C =$ Common Token by chance!

Monday, September 6, 2010
The results are not deterministically predictable.

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
- $C = \text{Common Token by chance!}$

**Diagram:**

- $W_1$, $F_1$, $W_2$, $F_2$, $W_3$, $F_3$
- $PF + FI_1$, $PF + TI$, $C$
- $PF$

**Legend:**

- **Worm**
- **Fake anomalous flow**
The results are not Deterministically Predictable

- $W_i$ = i-th Worm variant
- $F_i$ = i-th Fake anomalous flow
- $P(FP | F_i) < P(FP | T_I)$
- $PF$ = Protocol Framework
- $T_I$ = True Invariant
- $F_I$ = Fake Invariant
- $C$ = Common Token by chance!

$\begin{array}{ccccccc}
W_1 & F_1 & W_2 & F_2 & W_3 & F_3 \\
\text{PF + } & \text{FI-1} & \text{PF + } & \text{T}_I & \text{C} & \text{PF} \\
\text{PF} & \text{PF + } & \text{T}_I & \text{PF} & \text{PF + } & \text{T}_I\
\end{array}$

Worm

Fake anomalous flow
The results are not Deterministically Predictable

- $W_i =$ i-th Worm variant
- $F_i =$ i-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$
- $PF =$ Protocol Framework
- $TI =$ True Invariant
- $FI =$ Fake Invariant
- $C =$ Common Token by chance!

$W_1 \quad F_1 \quad W_2 \quad F_2 \quad W_3 \quad F_3$

$\overset{PF + FI-1,}{PF + TI \quad C}$

$\overset{PF}{PF + TI}$

$\overset{PF}{PF}$

Monday, September 6, 2010
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP | F_i) < P(FP | T_I)$
- $PF = \text{Protocol Framework}$
- $T_I = \text{True Invariant}$
- $F_I = \text{Fake Invariant}$
- $C = \text{Common Token by chance!}$

## Diagram

```
   \[ W_1 \quad F_1 \quad W_2 \quad F_2 \quad W_3 \quad F_3 \]
   \[ \begin{array}{c}
   \text{PF} + F_{i-1} \\
   \text{PF} + T_I \\
   \text{PF} + T_I \\
   \text{PF} + T_I \\
   \text{PF} \\
   \end{array} \]
```

Worm

Fake anomalous flow

Monday, September 6, 2010
The results are not Deterministically Predictable

- Wi = i-th Worm variant
- Fi = i-th Fake anomalous flow
- \( P(FP | FI) < P(FP | TI) \)
- PF = Protocol Framework
- TI = True Invariant
- FI = Fake Invariant
- C = Common Token by chance!

W1    F1    W2    F2    W3    F3

\[ PF + FI-1, \]
\[ PF + TI \]
\[ C \]
\[ PF \]
\[ PF + TI \]
\[ PF \]
\[ PF \]
\[ PF \]
\[ PF \]
\[ PF + FI-2 \]
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
- $C = \text{Common Token by chance!}$

\begin{align*}
W_1 & \quad F_1 & \quad W_2 & \quad F_2 & \quad W_3 & \quad F_3 \\
\text{PF} + FI_1, & \quad \text{PF} + TI & \quad C & \quad \text{PF} \\
\text{PF} + TI & \quad \text{PF} + TI & \quad \text{PF} \\
\text{PF} & \quad \text{PF} & \quad \text{PF} & \quad \text{PF} \\
\text{PF} + FI_2 & \quad \text{PF} + TI & \quad \text{PF}
\end{align*}

Worm

Fake anomalous flow
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP | F_i) < P(FP | TI)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
- $C = \text{Common Token by chance!}$

Monday, September 6, 2010
The results are not Deterministically Predictable

- **Wi** = i-th Worm variant
- **Fi** = i-th Fake anomalous flow
- **P(FP | FI) < P(FP | TI)**
- **PF** = Protocol Framework
- **TI** = True Invariant
- **FI** = Fake Invariant
- **C** = Common Token by chance!

<table>
<thead>
<tr>
<th></th>
<th>W1</th>
<th>F1</th>
<th>W2</th>
<th>F2</th>
<th>W3</th>
<th>F3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PF + FI-1,</td>
<td></td>
<td>PF + TI</td>
<td></td>
<td>PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PF + TI</td>
<td></td>
<td>PF + TI</td>
<td></td>
<td>PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PF</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Monday, September 6, 2010
The results are not Deterministically Predictable

- $W_i =$ i-th Worm variant
- $F_i =$ i-th Fake anomalous flow
- $P(F_P | F_I) < P(F_P | T_I)$
- $P(F_P | T_I)$
- $F_I =$ Fake Invariant
- $C =$ Common Token by chance!
- $F_P =$ Protocol Framework
- $T_I =$ True Invariant

Monday, September 6, 2010
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP | FI) < P(FP | TI)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
- $C = \text{Common Token by chance!}$

$W_1 F_1 W_2 F_2 W_3 F_3$

- $PF + F_{I-1}$
- $PF + TI$
- $PF$
- $PF + TI$
- $PF$
- $PF + F_{I-2}$
- $PF + TI$
- $PF$
- $PF$
- $PF + F_{I-3}$

Monday, September 6, 2010
The results are not Deterministically Predictable

- $W_i = \text{i-th Worm variant}$
- $F_i = \text{i-th Fake anomalous flow}$

- $P(FP | FI) < P(FP | TI)$

- PF = Protocol Framework
- TI = True Invariant
- FI = Fake Invariant
- C = Common Token by chance!

Monday, September 6, 2010
The results are not Deterministically Predictable

- $\text{Wi} = \text{i-th Worm variant}$
- $\text{Fi} = \text{i-th Fake anomalous flow}$

- $P(FP | FI) \geq P(FP | TI, C)$
- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
- $C = \text{Common Token by chance!}$

\[ Wi = i-th \text{ Worm variant} \]
\[ Fi = i-th \text{ Fake anomalous flow} \]

\[ P(FP | FI) \geq P(FP | TI, C) \]
\[ PF = \text{ Protocol Framework} \]
\[ TI = \text{ True Invariant} \]
\[ FI = \text{ Fake Invariant} \]
\[ C = \text{ Common Token by chance!} \]
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow

$P(F_P | F_I) \geq P(F_P | T_I, C)$

- $P_F = \text{Protocol Framework}$
- $T_I = \text{True Invariant}$
- $F_I = \text{Fake Invariant}$

$C = \text{Common Token by chance!}$

- Worm
- Fake anomalous flow

Monday, September 6, 2010
The results are not Deterministically Predictable

- Wi = i-th Worm variant
- Fi = i-th Fake anomalous flow

\[ P(FP | FI) \geq P(FP | TI, C) \]

- PF = Protocol Framework
- TI = True Invariant
- FI = Fake Invariant
- C = Common Token by chance!

Monday, September 6, 2010
The results are not Deterministically Predictable

- **Wi** = i-th Worm variant
- **Fi** = i-th Fake anomalous flow

- \[ P(F_P | F_I) \geq P(F_P | T_I, C) \]
- **PF** = Protocol Framework
- **TI** = True Invariant
- **FI** = Fake Invariant
- **C** = Common Token by chance!

Monday, September 6, 2010
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow

- $P(F_P | F_I) \geq P(F_P | T_I, C)$
- $PF = $ Protocol Framework
- $T_I = $ True Invariant
- $F_I = $ Fake Invariant
- $C = $ Common Token by chance!

$W = [W_1, W_2] F_1 F_2 W_3 F_3$
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow
- $P(FP \mid FI) \geq P(FP \mid TI,C)$
- $PF =$ Protocol Framework
- $TI =$ True Invariant
- $FI =$ Fake Invariant
- $C =$ Common Token by chance!

$Wi = i$-th Worm variant
$Fi = i$-th Fake anomalous flow

$P(FP \mid FI) \geq P(FP \mid TI,C)$

$PF =$ Protocol Framework
$TI =$ True Invariant
$FI =$ Fake Invariant
$C =$ Common Token by chance!
The results are not Deterministically Predictable

- $W_i = i$-th Worm variant
- $F_i = i$-th Fake anomalous flow

$P(FP | FI) \geq P(FP | TI, C)$

- $PF = \text{Protocol Framework}$
- $TI = \text{True Invariant}$
- $FI = \text{Fake Invariant}$
- $C = \text{Common Token \textbf{by chance!}}$

Good signature!

Monday, September 6, 2010
The results are not Deterministically Predictable

- \( W_i = \text{i-th Worm variant} \)
- \( F_i = \text{i-th Fake anomalous flow} \)

A useful signature could be produced, by chance

However, our experiments showed that the Noise Injection Attach has a high probability of success

Good signature!
Possible countermeasures

- White list
- "Coloring" technique