

# Assured Information Sharing for Security Applications: Malicious Code Detection

Prof. Bhavani Thuraisingham  
Prof. Latifur Khan  
Prof. Murat Kantarcioglu  
Prof. Kevin Hamlen

The University of Texas at Dallas

Project Funded by the Air Force Office of Scientific Research  
(AFOSR)

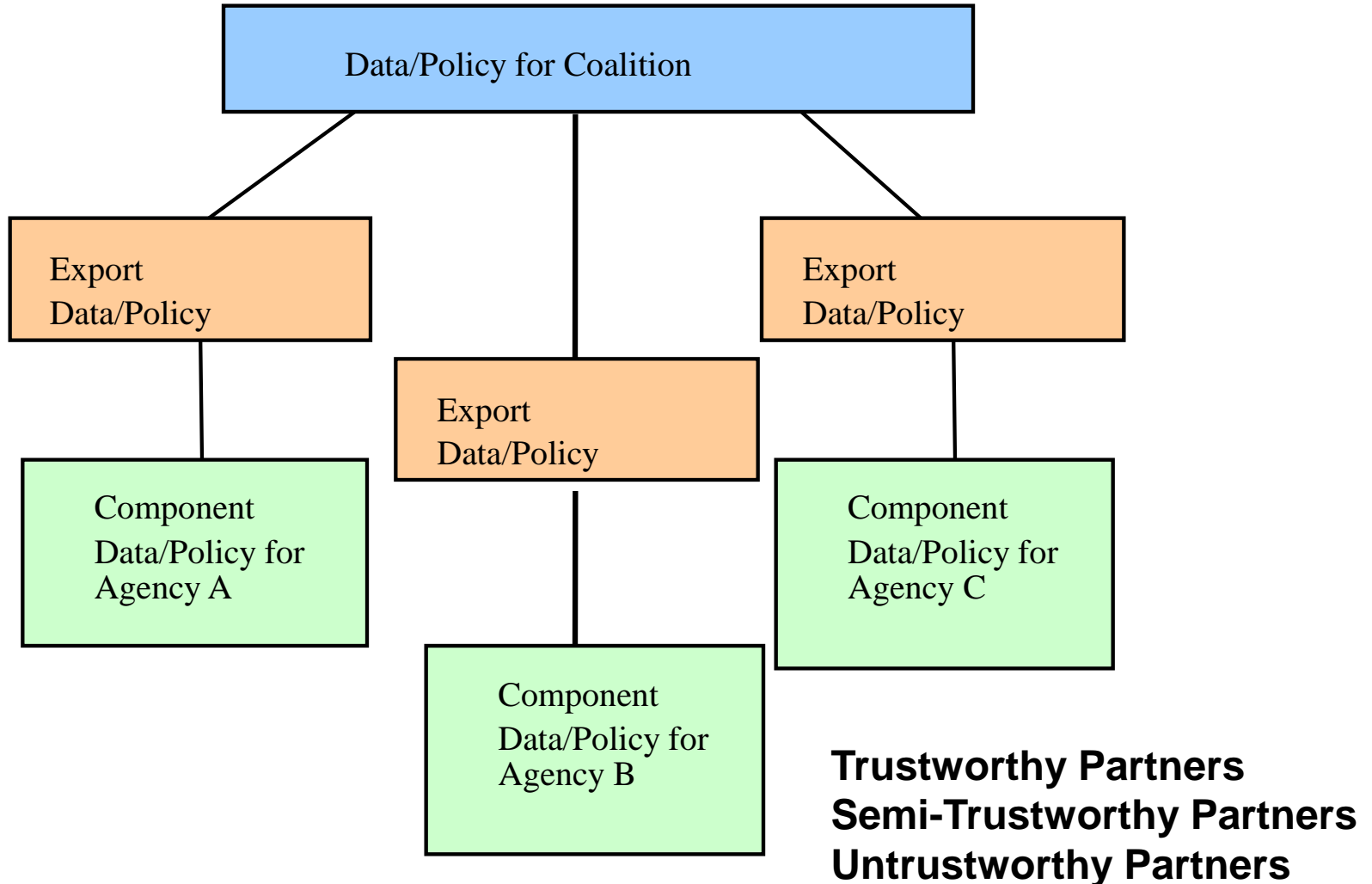
[bhavani.thuraisingham@utdallas.edu](mailto:bhavani.thuraisingham@utdallas.edu)

September 2009

# Assured Information Sharing

- Daniel Wolfe (formerly of the NSA) defined assured information sharing (AIS) as a framework that “provides the ability to dynamically and securely share information at multiple classification levels among U.S., allied and coalition forces.”
- The DoD’s vision for AIS is to “deliver the power of information to ensure mission success through an agile enterprise with freedom of maneuverability across the information environment”
- 9/11 Commission report has stated that we need to migrate from a need-to-know to a need-to-share paradigm
- Our objective is to help achieve this vision by defining an AIS lifecycle and developing a framework to realize it.

# Architecture: 2005-2008

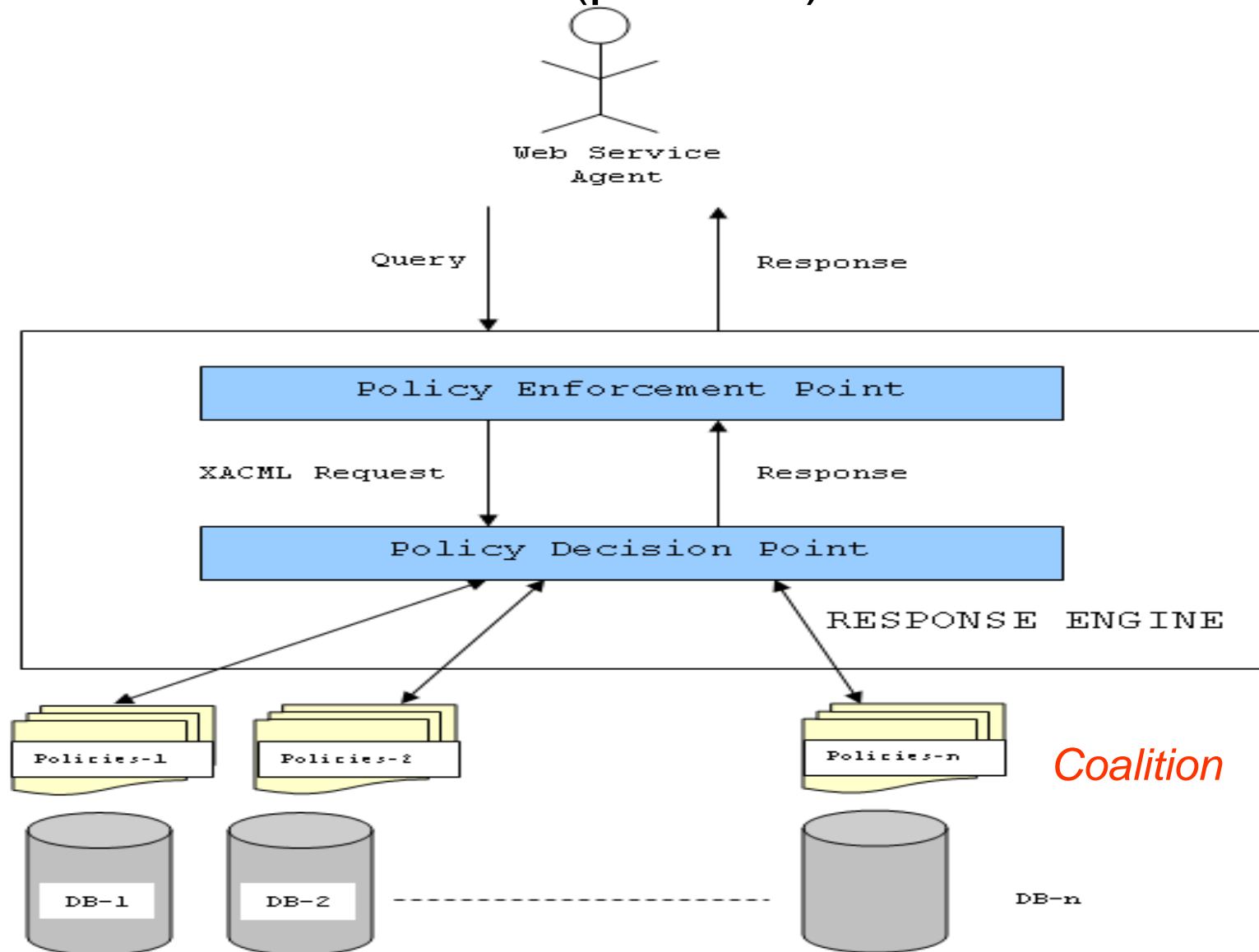


# Our Approach

- Integrate the Medicaid claims data and mine the data; next enforce policies and determine how much information has been lost (Trustworthy partners); Prototype system; Application of Semantic web technologies
- Apply game theory and probing to extract information from semi-trustworthy partners
- Conduct Active Defence and determine the actions of an untrustworthy partner
  - Defend ourselves from our partners using data mining techniques
  - Conduct active defence – find out what our partners are doing by monitoring them so that we can defend our selves from dynamic situations
- Trust for Peer to Peer Networks (Infrastructure security)

# Policy Enforcement Prototype

Dr. Mamoun Awad (postdoc) and students



# Architectural Elements of the Prototype

- ***Policy Enforcement Point (PEP):***

- Enforces policies on requests sent by the Web Service.
- Translates this request into an XACML request; sends it to the PDP.

- ***Policy Decision Point (PDP):***

- Makes decisions regarding the request made by the web service.
- Conveys the XACML request to the PEP.

### ***Policy Files:***

- Policy Files are written in XACML policy language. Policy Files specify rules for “Targets”. Each target is composed of 3 components: Subject, Resource and Action; each target is identified uniquely by its components taken together. The XACML request generated by the PEP contains the target. The PDP’s decision making capability lies in matching the target in the request file with the target in the policy file. These policy files are supplied by the owner of the databases (Entities in the coalition).

### ***Databases:***

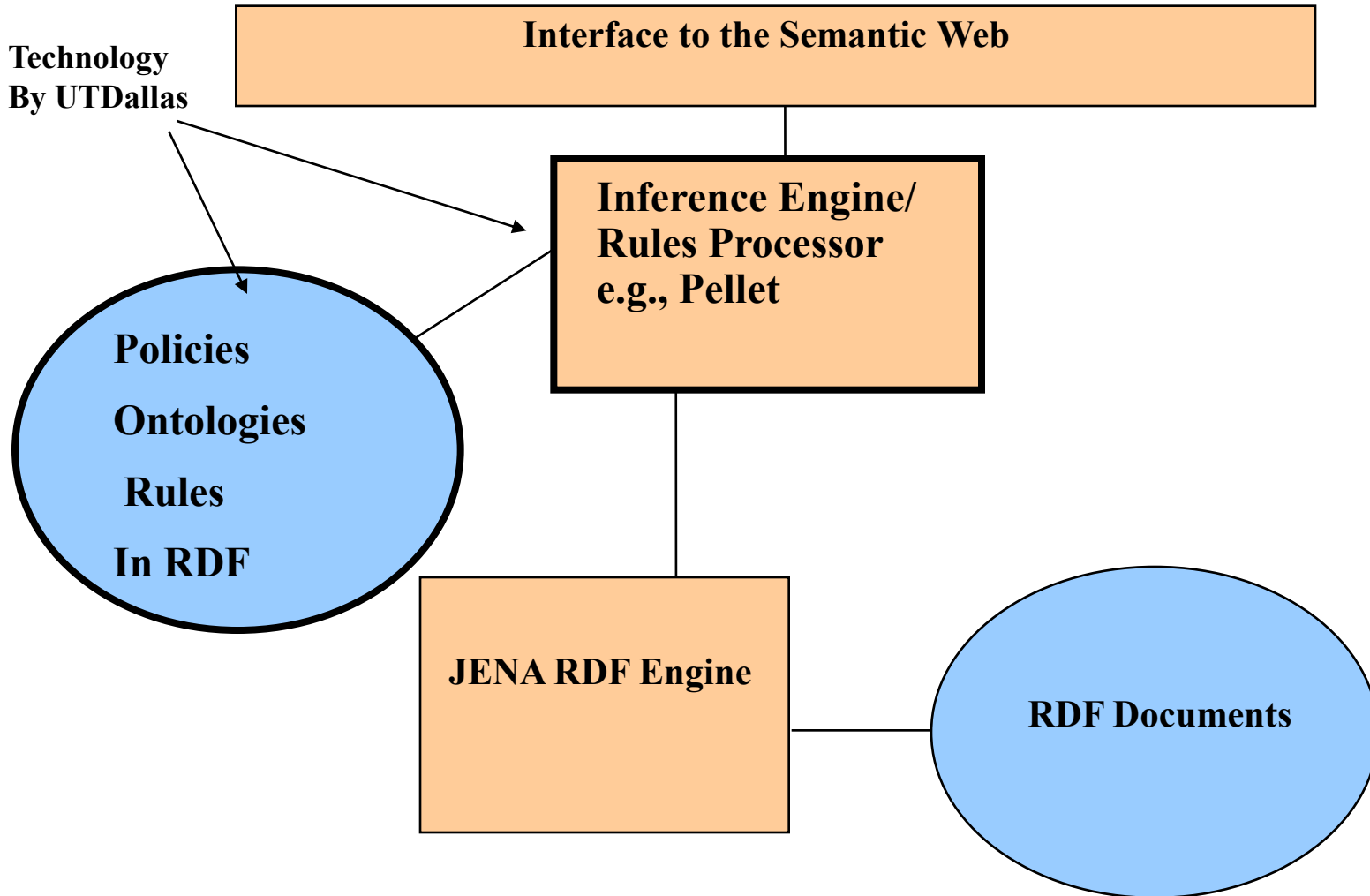
- The entities participating in the coalition provide access to their databases.

# Confidentiality, Privacy and Trust

## CPT

- Trust
  - Trust is established between say a web site and a user based on credentials or reputations.
- Privacy
  - When a user logs into a website to make say a purchase, the web site will specify that its privacy policies are. The user will then determine whether he/she wants to enter personal information.
  - That is, if the web site will give out say the user's address to a third party, then the user can decide whether to enter this information.
  - However before the user enters the information, the user has to decide whether he trusts the web site.
  - This can be based on the credential and reputation.
  - if the user trusts the web site, then the user can enter his private information if he is satisfied with the policies. If not, he can choose not to enter the information.
- Confidentiality
  - Here the user is requesting information from the web site;
  - the web site checks its confidentiality policies and decides what information to release to the user.
  - The web set can also check the trust it has on the user and decide whether to give the information to the user.

# Policy Engine



# Distributed Information Exchange

(Ryan Layfield, Murat Kantarcioglu,  
Bhavani Thuraisingham)

- Multiple, sovereign parties wish to cooperate
  - Each carries pieces of a larger information puzzle
  - Can only succeed at their tasks when cooperating
  - Have little reason to trust or be honest with each other
  - Cannot agree on single impartial governing agent
  - No one party has significant clout over the rest
  - No party innately has perfect knowledge of opponent actions
    - Verification of information incurs a cost
    - Faking information is a possibility
- Current modern example: Bit Torrent
  - Assumes information is verifiable
  - Enforces punishment however through a centralized server

# Game Theory

- Studies such interactions through mathematical representations of gain
  - Each party is considered a **player**
  - The information they gain from each other is considered a **payoff**
  - Scenario considered a **finite repeated game**
    - Information exchanged in discrete ‘chunks’ each round
    - Situation terminates at a finite yet unforeseeable point in the future
  - **Actions** within the game are to either **lie** or tell the **truth**
- **Our Goal: All players draw conclusion that telling the truth is the best option**

# Experimental Setup

- We created an evolutionary game in which players had the option of selecting a more advantageous behavior
- Available behaviors included:
  - Our punishment method
  - Tit-for-Tat
  - ‘Subtle’ lie
- Every 200 rounds, behaviors are re-evaluated

$$p^{select}(a_i) = \frac{f(a_i)}{\sum_{i=0}^n f(a_i)}$$

- **If everyone agrees on a truth-telling behavior, our goal is achieved**

# Results

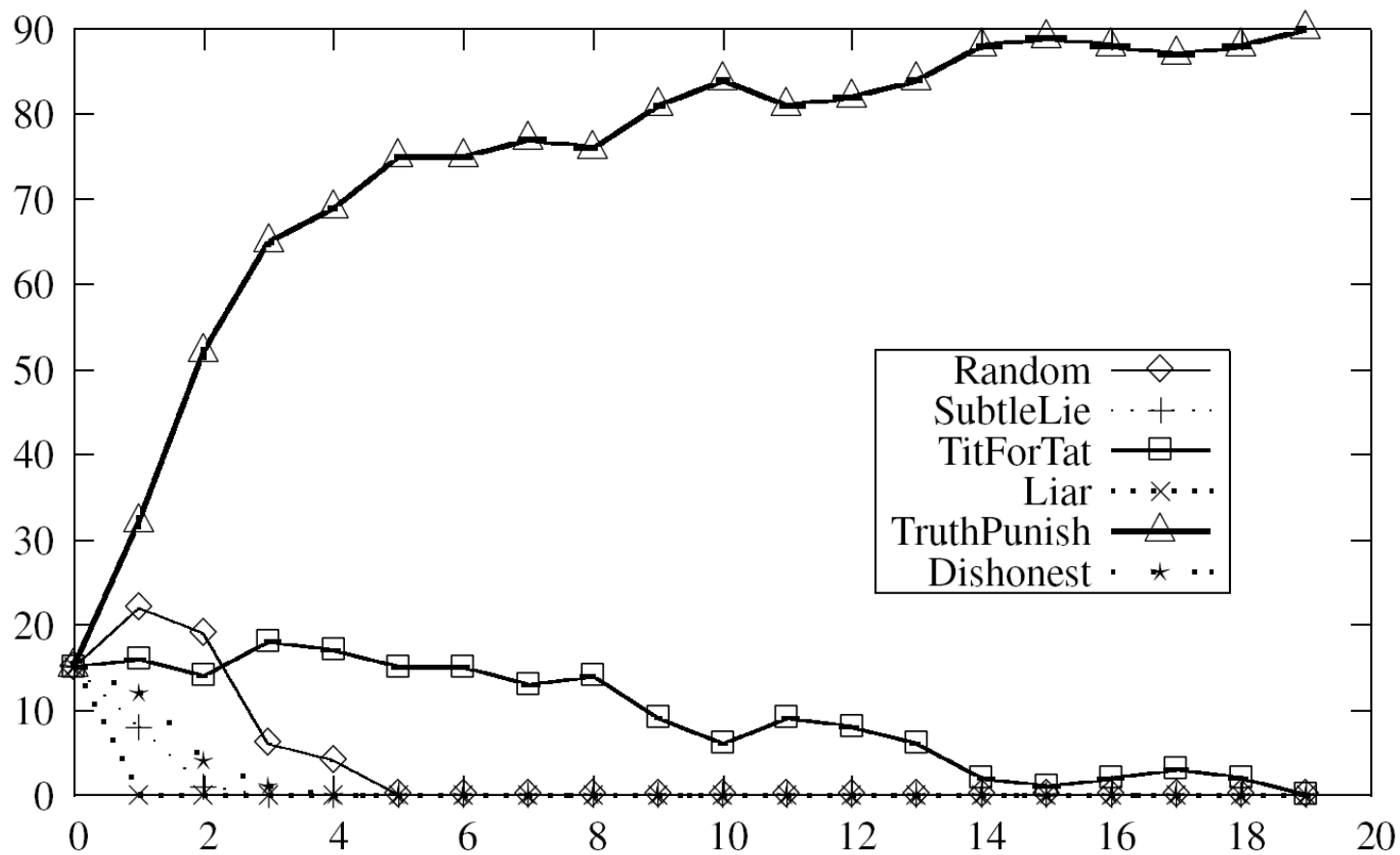


Figure 3: Behaviors by population per generation

# Conclusions:

## Semi-trustworthy partners

- Experiments confirm our behavior success
  - Equilibrium of behavior yielded both a homogenous choice of *TruthPunish* and truth told by all agents
  - Rigorous despite wide fluctuations in payoff  $\Delta$
- Notable Observations
  - Truth-telling cliques (of mixed behaviors) rapidly converged to *TruthPunish*
  - Cliques, however, only succeeded when the ratio of like-minded helpful agents outweighed benefits of lying periodically
    - Enough agents must use punishment ideology
  - Tit-for-Tat was the leading competitor

# Defensive Operations: Detecting Malicious Executables using Data Mining

- **What are malicious executables?**
  - Harm computer systems
  - ***Virus, Exploit, Denial of Service (DoS), Flooder, Sniffer, Spoofer, Trojan etc.***
  - Exploits software vulnerability on a victim
  - May remotely infect other victims
  - Incurs great loss. Example: **Code Red** epidemic cost \$2.6 Billion
- **Malicious code detection: Traditional approach**
  - Signature based
  - Requires signatures to be generated by human experts
  - So, not effective against “zero day” attacks

# Automated Detection

## O State of the Art

- O Automated detection approaches:
  - **Behavioural**: analyse behaviours like source, destination address, attachment type, statistical anomaly etc.
  - **Content-based**: analyse the content of the malicious executable
    - Autograph (H. Ah-Kim – CMU): Based on automated signature generation process
    - N-gram analysis (Maloof, M.A. et .al.): Based on mining features and using machine learning.

## X Our New Ideas

- X **Content -based approaches consider only machine-codes (byte-codes).**
- X **Is it possible to consider higher-level source codes for malicious code detection?**
- X **Yes: Disassemble the binary executable and retrieve the assembly program**
- X **Extract important features from the assembly program**
- X **Combine with machine-code features**

# Feature Extraction

## X Binary n-gram features

- Sequence of n consecutive bytes of binary executable

## X Assembly n-gram features

- Sequence of n consecutive assembly instructions

## X System API call features

- DLL function call information

## •Hybrid Approach

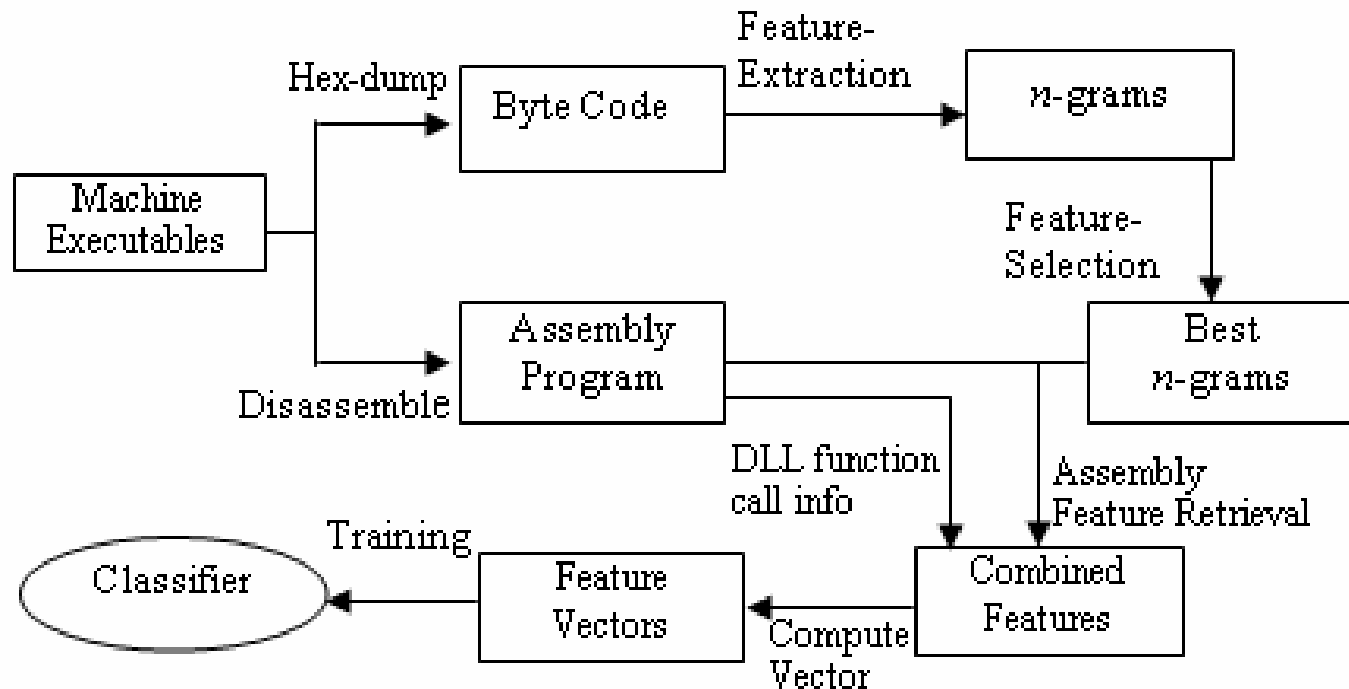
- Collect training samples of normal and malicious executables.

Extract features

- Train a Classifier and build a model
- Test the model against test samples

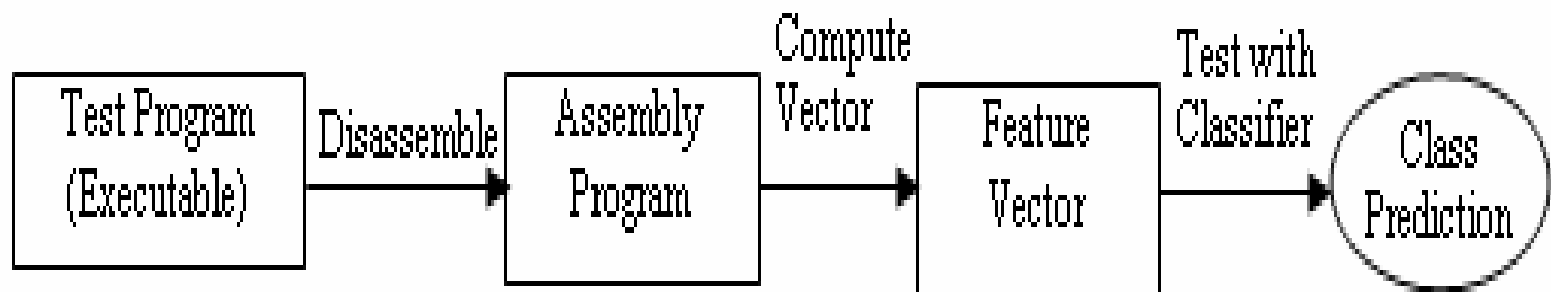
# Hybrid Feature Retrieval (HFR)

- Training



# Hybrid Feature Retrieval (HFR)

- Testing



# Feature Extraction

## Binary n-gram features

- Features are extracted from the byte codes in the form of  $n$ -grams, where  $n = 2, 4, 6, 8, 10$  and so on.

## Example:

Given a 11-byte sequence: 0123456789abcdef012345,

The 2-grams (2-byte sequences) are: 0123, 2345, 4567, 6789, 89ab, abcd, cdef, ef01, 0123, 2345

The 4-grams (4-byte sequences) are: 01234567, 23456789, 456789ab, ..., ef012345 and so on....

## Problem:

- Large dataset. Too many features (millions!).

## Solution:

- Use secondary memory, efficient data structures
- Apply feature selection

# Feature Extraction

Assembly n-gram features

- Features are extracted from the assembly programs in the form of  $n$ -grams, where  $n = 2, 4, 6, 8, 10$  and so on.

Example:

three instructions

*“push eax”*; *“mov eax, dword[0f34]”*; *“add ecx, eax”*;

2-grams

(1) *“push eax”*; *“mov eax, dword[0f34]”*;

(2) *“mov eax, dword[0f34]”*; *“add ecx, eax”*;

Problem: Same problem as binary

Solution: Select best features

- Select Best K features
- Selection Criteria: Information Gain
- *Gain of an attribute A on a collection of examples S is given by*

$$\textit{Gain}(S, A) \equiv \textit{Entropy}(S) - \sum_{V \in \textit{Values}(A)} \frac{|S_V|}{|S|} \textit{Entropy}(S_V)$$

# Experiments

- Dataset
  - Dataset1: 838 Malicious and 597 Benign executables
  - Dataset2: 1082 Malicious and 1370 Benign executables
  - Collected Malicious code from VX Heavens (<http://vx.netlux.org>)
- Disassembly
  - Pedisassem ( <http://www.geocities.com/~sangcho/index.html> )
- Training, Testing
  - Support Vector Machine (SVM)
  - C-Support Vector Classifiers with an RBF kernel

# Results - I

- HFS = Hybrid Feature Set
- BFS = Binary Feature Set
- AFS = Assembly Feature Set

TABLE -I

n	Dataset1			Dataset2		
	HFS	BFS	AFS	HFS	BFS	AFS
1	93.4	63.0	88.4	92.1	59.4	88.6
2	96.8	94.1	88.1	96.3	92.1	87.9
4	96.3	95.6	90.9	97.4	92.8	89.4
6	97.4	95.5	87.2	96.9	93.0	86.7
8	96.9	95.1	87.7	97.2	93.4	85.1
10	97.0	95.7	73.7	97.3	92.8	75.8
<b>Avg</b>	<b>96.30</b>	<b>89.83</b>	<b>86.00</b>	<b>96.15</b>	<b>87.52</b>	<b>85.58</b>

5.  
th  
2.  
m  
:k  
id  
re  
re



# Results - II

- HFS = Hybrid Feature Set
- BFS = Binary Feature Set
- AFS = Assembly Feature Set

TABLE -III  
FALSE POSITIVE AND FALSE NEGATIVE RATES ON DIFFERENT FEATURE SETS

n	Dataset1			Dataset2		
	HFS	BFS	AFS	HFS	BFS	AFS
1	80/5.6	77.7/7.9	12.4/11.1	7.5/8.3	65.0/9.8	12.8/9.6
2	53/1.7	60/5.7	22.8/4.2	3.4/4.1	5.6/10.6	15.1/8.3
4	49/2.9	6.4/3.0	16.4/3.8	2.5/2.2	7.4/6.9	12.6/8.1
6	35/2.0	5.7/3.7	24.5/4.5	3.2/2.9	6.1/8.1	17.8/7.6
8	49/1.9	6.0/4.1	26.3/2.3	3.1/2.3	6.0/7.5	19.9/8.6
10	55/1.2	5.2/3.6	43.9/1.7	3.4/1.9	6.3/8.4	30.4/16.4
<b>Avg</b>	<b>5.4/2.6</b>	<b>17.8/4.7</b>	<b>24.4/3.3</b>	<b>3.9/3.6</b>	<b>16.1/8.9</b>	<b>18.1/9.8</b>

# Peer to Peer Botnet Detection

Masud, M. M. <sup>1</sup>, Gao, J.<sup>2</sup>, Khan, L. <sup>1</sup>, Han, J.<sup>2</sup>,  
Thuraisingham, B<sup>1</sup>

<sup>1</sup>University of Texas at Dallas

<sup>2</sup>University of Illinois at Urbana Champaign

- Data Mining Approach
- Monitor Stream Data

# Background

- Botnet
  - Network of compromised machines
  - Under the control of a botmaster
- Taxonomy:
  - C&C : Centralized, Distributed etc.
  - Protocol: IRC, HTTP, P2P etc.
  - Rallying mechanism: Hard-coded IP, Dynamic DNS etc.
- Network traffic monitoring

# What To Monitor?

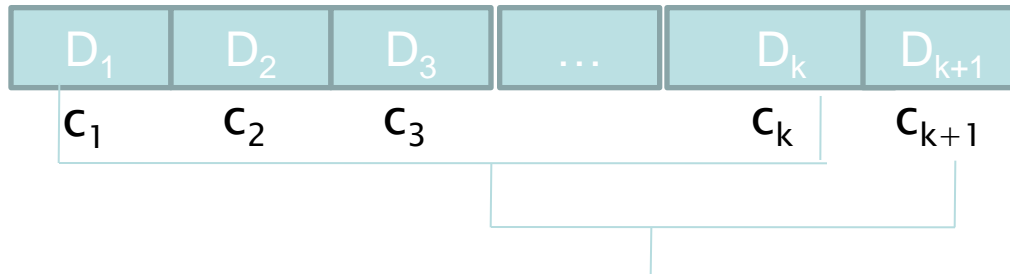
- Monitor Payload / Header?
- Problems with payload monitoring
  - Privacy
  - Unavailability
  - Encryption/Obfuscation
- Information extracted from Header (features)
  - New connection rate
  - Packet size
  - Upload/Download bandwidth
  - Arp request & ICMP echo reply rate

# Mapping to Stream Data Mining

- Stream data : Stream data refers to any continuous flow of data.
  - For example: network traffic / sensor data.
- Properties of stream data : Stream data has two important properties: *infinite length & concept drift*
- Stream data classification: Cannot be done with conventional classification algorithms
- We propose a multi-chunk multi-level ensemble approach to solve these problems,
  - which significantly reduces error over the single-chunk single-level ensemble approaches.

# The Single-Chunk Single-Level Ensemble (SCE) Approach

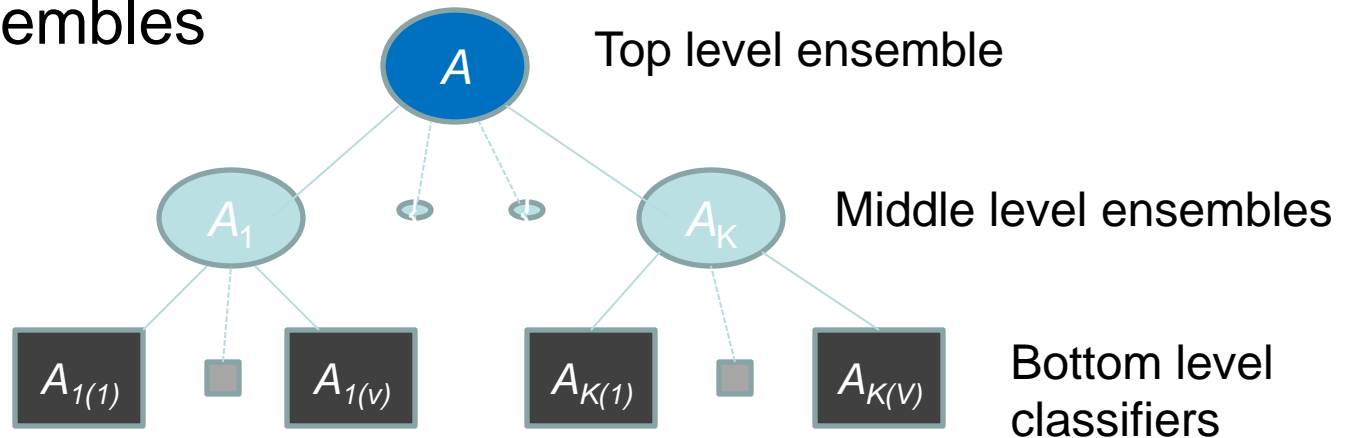
- Divide the data stream into equal sized chunks
  - Train a classifier from each data chunk
  - Keep the best  $K$  such classifier-ensemble



- Select best  $K$  classifiers from  $\{c_1, \dots, c_k\} \cup \{c_{k+1}\}$

# Our Approach: Multi-Chunk Multi-Level Ensemble (MCE)

- Train  $v$  classifiers from  $r$  consecutive data chunks, and create an ensemble, and Keep the best  $K$  such ensembles



- Two-level ensemble hierarchy:
  - Top level ( $A$ ): ensemble of  $K$  middle level ensembles  $A_i$
  - Middle level ( $A_i$ ): ensemble of  $v$  bottom level classifiers  $A_{i(j)}$

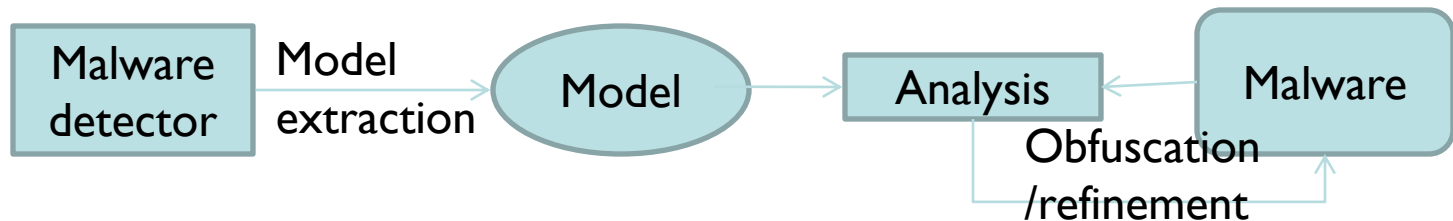
# Offensive Operation: Overview

Kevin Hamlen, Mehedy Masud, Latifur Khan,  
Bhavani Thuraisingham

- Goal
  - To hack/attack other person's computer and steal sensitive information
  - Without having been detected
- Idea
  - Propagate malware (worm/spyware etc.) through network
  - Apply obfuscation so that malware detectors fail to detect the malware
- Assumption
  - The attacker has the malware detector (valid assumption because anti-virus software are public)

# Strategy

- Steps:
  - Extract the model from the malware detector
  - Obfuscate the malware to evade the model
  - Dynamic approach



- There have been some works on automatic model extraction from malware detector, such as:

Christodorescu and Jha. Testing Malware Detectors. In Proc. 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2004).

# **Research Transitioned into AIS MURI – AFOSR UMBC-Purdue-UTD-UIUC-UTSA-UofMI 2008-2013**

- (1) Develop a Assured Information Sharing Lifecycle (AISL)
- (2) a framework based on a secure semantic event-based service oriented architecture to realize the life cycle
- (3) novel policy languages, reasoning engines, negotiation strategies, and security infrastructures
- (4) techniques to exploit social networks to enhance AISL
- (5) techniques for federated information integration, discovery and quality validation
- (6) techniques for incentivized assured information sharing.
- Unfunded Partners: Kings College Univ of London and Univ of Insurbria (Steve Barker, Barbara Carminati, Elena Ferrari)