Noisy Defenses: Subverting Malware’s OODA loop

Daniel Bilar

University of New Orleans
Department of Computer Science

May 12, 2008
CSIIRW ’08, Oak Ridge National Labs
## Talk roadmap

### Status Quo

Classic AV byte-pattern matching may have reached its practical and theoretical limits with present modern malware.

### Research 2005-2007: Structural fingerprinting

Can statistical structural ‘fingerprinting’ assist MW identification, classification? Approaches included Win32 API calls, opcode frequency distribution and callgraph properties.

### Moving forward: Noisy Defenses

Enter malware’s decision cycle and adapt responses to influence malware’s next steps. Comprehensively control entropy of targets (passive defense) and environments (active defense) to manipulate malware’s OODA loop.

### A glimpse at the unholy future

RoQ, IC and quantum malware.
**Malware taxonomy: Poly- and Metamorphic Malware**

**Description**

**Polymorphic malware** contain decryption routines which decrypt encrypted constant parts of body.

**Metamorphic malware** generally do not use encryption, but mutates forms in subsequent generations.

**Figure:** Mutated body in subsequent generations
Malware taxonomy: Interactive Malware

**Figure:** Interactive Malware: Open vs closed system. Interacts with OS, network, other programs, humans. May also use dissimulation techniques. Picture by Grégoire Jacob, French Army Signal Academy ESAT (France)
## Metamorphic/Polymorphic Detection

### Empirical AV Results

<table>
<thead>
<tr>
<th>Date</th>
<th>#MW Family</th>
<th>#AV Scanners</th>
<th>Miss (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008/02</td>
<td>8</td>
<td>15</td>
<td>27</td>
</tr>
<tr>
<td>2007/08</td>
<td>10</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>2007/02</td>
<td>10</td>
<td>15</td>
<td>38</td>
</tr>
<tr>
<td>2006/08</td>
<td>10</td>
<td>15</td>
<td>48</td>
</tr>
</tbody>
</table>

*Table: Updated AV scanners against hundreds of well-known, previously submitted, highly polymorphic and metamorphic malware samples (AV-Comparatives.org)*

### Research 2005-2007: Statistical Structural Fingerprinting

<table>
<thead>
<tr>
<th>Structural Perspective</th>
<th>Description</th>
<th>Statistical Fingerprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly instructions</td>
<td>Tabulate instructions</td>
<td>Opcode frequency distribution</td>
</tr>
<tr>
<td>Win32 API calls</td>
<td>Observe API calls made</td>
<td>API call vector</td>
</tr>
<tr>
<td>Callgraph</td>
<td>Explore control flow of functions</td>
<td>Graph structural properties</td>
</tr>
</tbody>
</table>
1st Fingerprint: Win32 API Calls

**Synopsis**
Observe and record Win32 API calls made by malicious code during execution, then compare them to calls made by other malicious code to find similarities.

**Goal**
Classify malware quickly into a family
Set of variants make up a family

**Main Result (2005) [?]**
Simple (tuned) Vector Space Model yields over 80% correct classification
## Win32 API calls: Results

<table>
<thead>
<tr>
<th>Family</th>
<th># of members</th>
<th># correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apost</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Banker</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Nibu</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Tarno</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Beagle</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>Blaster</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Frethem</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Gibe</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Inor</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Klez</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mitgeld</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MyDoom</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>MyLife</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Netsky</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Sasser</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>SDBot</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Moega</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Randex</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Spybot</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Pestlogg</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Welchia</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

### Classification and AV corroboration

Classification by 17 AV scanners yields 21 families. > 80 % correspondence (\(csm\) threshold = 0.8).

### Threshold parameter

<table>
<thead>
<tr>
<th>Threshold</th>
<th>%</th>
<th>#</th>
<th>false fam.</th>
<th>both</th>
<th>miss. fam.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td>0.8</td>
<td>62</td>
<td>5</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>0.75</td>
<td>0.8</td>
<td>62</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>0.8</td>
<td>0.82</td>
<td>63</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>0.85</td>
<td>0.82</td>
<td>63</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>0.9</td>
<td>0.79</td>
<td>61</td>
<td>1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>0.95</td>
<td>0.79</td>
<td>61</td>
<td>2</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>0.99</td>
<td>0.62</td>
<td>48</td>
<td>0</td>
<td>2</td>
<td>27</td>
</tr>
</tbody>
</table>

Figure: Threshold parameter

Figure: 77 malware samples classified
2nd Fingerprint: Opcode Frequency

Synopsis

Statically disassemble the binary, tabulate the opcode frequencies and construct a statistical fingerprint with a subset of said opcodes.

Goal

Compareopcode fingerprint across non-malicious software and malware classes for quick identification purposes.

Main Result (2006) [?]

For differentiation purposes, infrequent opcodes explain more data variation than common ones.
**Opcode frequency: Results**

<table>
<thead>
<tr>
<th>Cramer's V (in %)</th>
<th>Op</th>
<th>Krn</th>
<th>Usr</th>
<th>Tools</th>
<th>Bot</th>
<th>Trojan</th>
<th>Virus</th>
<th>Worm</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>bt</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td>36</td>
<td>fdivp</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td>42</td>
<td>fild</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td>17</td>
<td>fstcw</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td>16</td>
<td>imul</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td>10</td>
<td>int</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td>12</td>
<td>nop</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td></td>
<td>pushf</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td></td>
<td>rdtsc</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td></td>
<td>sbb</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td></td>
<td>xbb</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td></td>
<td>setb</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td></td>
<td>setle</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td></td>
<td>shld</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>blue</td>
<td>green</td>
<td>yellow</td>
<td>red</td>
<td>orange</td>
<td>yellow</td>
<td>brown</td>
</tr>
</tbody>
</table>

**Figure:** Infrequent opcodes explain more variation, hence better predictor for malware differentiation

**INT:** Rooktikts (and tools) make heavy use of software interrupts ➔ tell-tale sign of RK?

**NOP:** Virus makes use ➔ NOP sled, padding?
3rd Fingerprint: Graph properties

Synopsis
Represent executables as callgraph, and construct graph-structural fingerprint for software classes

Goal
Compare ‘graph structure’ fingerprint of unknown binaries across non-malicious software and malware classes

Main Result (2007) [?]
Malware tends to have a lower basic block count, implying a simpler structure: Less interaction, fewer branches, limited functionality
Callgraph: Degree Distribution

Power (Pareto) law

Investigate whether indegree $d_{\text{indeg}}(f)$, outdegree $d_{\text{outdeg}}(f)$ and basic back count $d_{\text{bb}}(f)$ distributions of executable’s functions follows a truncated power law of form

$$P_{d^*}(m) \sim m^{\alpha_{d^*}(f)} e^{-\frac{m}{k_c}}$$

with $\alpha$ a power law exponent, $k_c$ distribution cutoff point, $\hat{\alpha}(n)$ Hill estimator (inset) used for consistency check

Figure: Pareto fitted ECCDF with Hill estimator $\hat{\alpha}(n)$
# Callgraph: Differentiation Results

<table>
<thead>
<tr>
<th>class</th>
<th>Basic Block</th>
<th>Indegree</th>
<th>Outdegree</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>2.57</td>
<td>1.04</td>
<td>-0.47</td>
</tr>
<tr>
<td>Goodware</td>
<td>N(1.634,0.3)</td>
<td>N(2.02, 0.3)</td>
<td>N(1.69,0.307)</td>
</tr>
<tr>
<td>Malware</td>
<td>N(1.7,0.3)</td>
<td>N(2.08,0.45)</td>
<td>N (1.68,0.35)</td>
</tr>
</tbody>
</table>

**Table:** Only one statistically relevant difference found: Basic block distribution metric $\mu_{\text{malware}}(k_{bb}) \neq \mu_{\text{goodware}}(k_{bb})$ via Wilcoxon Rank Sum

**Interpretation**

Malware tends to have a **lower basic block count**, implying a simpler structure: Less interaction, fewer branches, limited functionality.
Quo vadis, AV?

Modern Malware Detection Complexity

**Bad** Classic ‘white-box’ AV has been failing for maybe half a decade. Structural fingerprinting is of no direct help (alas).

**Worse** Practical AV detection in hitherto tractable *linear time* struggling against *NP-completeness* and *undecidability*. Detection of interactive malware is at least in class $\sum_3$, no longer $\Pi_2$ [?]

Conjecture

Detection of modern malware through techniques based on (strong) Church-Turing thesis (computation-as-functions) a practical/theoretical dead-end

Moving forward

Posit need for shift towards ‘interactive computation’, both in practice and theory
Interactive Computation [?]

Idea

Theoretical bridge between TM (functions) and concurrency (communication) aspects

Back to the Future: Elucidated by Turing

Turing proposed interactive choice machines as another model of computation distinct from automatic TMs - and not reducible to them.

Figure: Open I/O during computation, not just before or after
Information-Gain Adversarial Malware

Entropy

**Byte sequence-matching AV approach**
Input (suspicious data) → function (pattern-based decision) → output (yes/no malicious)

Modern malware systematically thwarts AV information gain through multi-stage, interactive, opaque implementation

How to defend?

**Two can play the information thwarting game** Adapt environment / defenses to control malware’s information gain for benefit of defender

Noisy Defense Idea

Enter malware’s decision cycle and adapt responses to influence malware’s next steps. Comprehensively control entropy of targets (passive defense) and environments (active defense) to enter malware’s OODA loop.
## Passive and Active Defenses

<table>
<thead>
<tr>
<th>Passive Defenses</th>
<th>Active Defenses</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Approach</strong></td>
<td>Model malware’s internal hypothesis structure, then control decisions by manipulating view of the projected world</td>
</tr>
<tr>
<td>Prevent exact identification of targets and environment by introducing irregularities</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Components</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network/Host</strong> Virtualization, honeynets (simulated decoys that detract from ‘real’ ones)</td>
<td><strong>Observation Framework</strong> Infer MW’s internal hypothesis structure via dynamic ‘black-box’ interaction</td>
</tr>
<tr>
<td><strong>Program/OS</strong> Hot-patch binaries, ASLR (random heap, stack, library positioning)</td>
<td><strong>Control Framework</strong> Dynamically choose strategies that control adversarial information gain for benefit of defense</td>
</tr>
<tr>
<td><strong>Implementation</strong> Mitigate side channel attacks (data structure, protocols, observables)</td>
<td></td>
</tr>
</tbody>
</table>

---

**Sources**

Passive and Active Defenses

---

**Passive Defense**

Overview

Malware

Structural Detection

Noisy Defense

Unholy Present/Future

Sources
## Active Defense Framework

### Observe and Choose Model: Process Query System [?]

**PQS** DBMS framework that allows for process description queries against internal models (FSM, Hidden Markov, etc). Designed to solve the Discrete Source Separation Problem.

**Idea**

Detect processes by leveraging correlation between observations and processes’ states.

**Processes detection** Models of MW’s internal control structure.

**State estimation** Estimate the current control flow state the malware is in.

### Control Information Gain: Play Games

Create illusion of win-win situation viz the malware’s goals by weakening useful/accurate and strengthening useless/misleading information gain.

**Idea**

Play repeated, Bayesian, imperfect-information games to probabilistically identify entities, then infer/estimate its control structure, then use OODA-loop controlling strategies.
Active Defense: Illustration with Toy Example

Suppose malware has been (perhaps probabilistically) identified through interaction and observation. Its toy internal hypothesis structure (and strategies) are modeled by Scan; if XP penetrate; if filtered DoS in a PQS internal model. The defense’s strategies are $S$ (no defense), $S_{HP}$ (honeypot), $S_{FLT}$ (filter/block ICMP response). The game matrix shows (hypothetical) payoffs of the defense’s and malware’s strategy combinations.

The malware starts scanning and wants to get to [Pen, $S$] penetrating a real host. The defense wants to engage sequential strategies such that the malware penetrates a fake host [Pen, $S_{HP}$], thereby giving the illusion of a win for the malware while learning more about it. Again, the defense wants to iteratively control, not necessarily minimize malware’s hoped-for entropy reduction. Strategies may not be fixed and dynamically generated as PQS models adapt to the malware responses, as denoted by $S_{...}$ (new defense strategy) and $<UO>$ (unknown observation).

**Figure:** Game loop, PQS modeling example malware’s control structure
Scan; if XP penetrate; if filtered DoS
Thoughts

Open Questions

PQS models
Dynamic inference to model malware’s hypothesis structure is open area. Formalization of interactive malware started just very recently.

Suitable System State Description and Entropy Measures
Defense goal is not to maximally confuse malware, but to manipulate malware’s decision tree by controlling its cross-entropy calculus $D^x$ of perceived target/environent.

Requires *appropriate state representation of targets/environments*, since this directly determines cross-entropy measure $D^x$.

Ex: If system’s governing distribution (probability of given realization) $P = P(n_i | q_i, N, s, I)$ s.t. prior probabilities $q_i$, number of entities $N$, number of states $s$ with $\sum_{i=1}^{s} n_i = N$ and background information $I$ is *multinomial* with $P = N! \prod_{i=1}^{s} \frac{q_i^{n_i}}{n_i!}$, then cross-entropy to manipulate is

\[
D_{KL}^x = \sum_{i=1}^{s} \left( p_i N^{-1} \ln N! + p_i \ln q_i - N^{-1} \ln((p_i N)!) \right).
\]

However, if system is not governed by multinomial $P$ (e.g. Bose-Einstein system’s $P_{BE}$ is multivariate negative hypergeometric), $D_{BE}^x$ is KL only in special cases [?].

Default Distrust?
When to engage active entropic defense? All interactions (benign and malicious) are suspect by default. Interactions as validation mechanism [?]. Feasible, workable?
## The Bagle Worm Example

### Example: Bagle/Beagle

Email-born worm, first appeared in 2004 [?]

### Strategy

Server-side metamorphic [?]
High, at times bursty, intensity release
Few instances (10s-100s) per variant

### Prevalence

Distinct variants (01/09-02/25/2007): **30,000** [?]
Average distinct variants per day: **625**
Not the only one: Feebs, Warezov
Bagle’s RoQ Strategy Illustrated

Figure: Bursty spread

Figure: Small batches per spread

Reduction-of-Quality attack

Bagle’s metamorphic process can be seen as a multi-level RoQ attack [?] against generalized AV systems.
IC Malware

Figure: IC Manufacturing process. Picture from [?]

DARPA BAA07-024
Determine whether IC manufactured in untrusted environment can be trusted to perform just operations specified **and no more**
Malicious IC: Write enable on Trigger

Figure: Picture from [?]. 09/06/2007: Israeli strike against Syrian nuclear reactor. Was a hardware kill switch used to disable air defense and radar systems? Precedent: Exocet missiles in 1982 UK-Argentine Falkland war
Generalized Side Channel Attacks

Principle
Attacker infers process’ state from inadvertently leaked low entropy ancillary observables like time, power, EM radiation, sound, protocols etc. Easiest example: Wrong login/password error message.

Some Attack Channel Examples

**Timing Analysis** Database indexing and data structure operations duration not uniform. Can retrieve privileged data [?]

**Differential Power Analysis** Current used in switching reveals activity that can be mapped to processes. Popular against cryptographic hardware [?]

**CPU Sounds** TEMPEST redux. CPU operations exhibit characteristic acoustic spectral signature [?]

Entropic Defenses
Known defenses overwhelmingly rely on *multi-level high entropy implementations* to ‘mask’ leaks.
FIPS 140-3 will require consideration of certain side channel attacks.
Quantum Cryptography

Figure: QC scheme solves (1-time pad) key distribution problem
Concept: Quantum Malware

Quantum Cryptography

**Used today** Swiss national elections, Oct 21st 2007
BBN Technologies involved, as well as MagiQ, ID Quantique

Malware to come?
Quantum Malware designed to decohere qubits’ phase and thus randomize data through phase gates, distort operations of quantum networks by malicious interference [?]

**Figure:** Proposed Quantum AV scheme. Picture from http://tinyurl.com/3ctx4y
References I


References II


