Jackstraws: Picking Command and Control Connections from Bot Traffic

Grégoire Jacob\textsuperscript{1}, Ralf Hund\textsuperscript{2}, Christopher Kruegel\textsuperscript{1}, Thorsten Holz\textsuperscript{2}

\textsuperscript{1} University of California, Santa Barbara / \textsuperscript{2} Ruhr-University Bochum

Fri Aug 12 2011
Introduction: the botnet threat

What do botnets do?
- Support large-scale malicious activities and the underground economy
- Coordination of malicious attacks
  e.g., denial of service, spam campaigns, click fraud
- Sensitive information theft
  e.g., credentials, credit card numbers

Why are botnets so convenient for attackers?
- Command & Control (C&C) infrastructure for remote control
- Incoming commands to trigger attacks and updates
- Outgoing responses for status monitoring and information leakage
**Introduction:** fighting against botnets

### Botnet detection and mitigation

- **Host-based techniques**
  - Traditional malware detection and mitigation
  - Signature matching and behavior monitoring
- **Network-based techniques**
  - Blacklisting IPs related to C&C servers
  - Signatures matching C&C protocol and commands
- **Automatic generation of these signatures, IP lists or models**
  - **Clean C&C only logs needed for traffic and system calls**

### Difficulty of identifying C&C traffic

- Potentially encrypted C&C traffic
- Non-C&C or “noise” traffic interleaved
  - Malicious connections to 3\(^{rd}\) party websites (e.g., part of the attacks)
  - Configuration connections (e.g., connectivity tests, time recovery)
  - Fake benign connections (e.g., mimicry of legitimate applications)
Introduction: identifying C&C traffic

Our approach: Jackstraws

- Combination of network traces and host-based activity
  - **Rationale:** C&C traffic results in observable host activity
    e.g. system modifications, critical information accesses
  - Host-based model: system call graphs with data dependency
  - Network-related link: each graph associated to a network connection

- Machine learning to identify and generalize C&C-related host activity
  - **Rationale:** similar commands result in similar core activities
    even for different bots
  - Mining significant activities: graph mining over *known* connections
  - Identifying similar activity types: graph clustering
  - Abstracting activity types: graph merging into templates
  - Detecting C&C activity: template matching over *unknown* connections
System: JACKSTRAWS overview

System architecture

- Bot Samples
  - Dynamic Analysis Graph Generation
    - Activity Graphs
    - Network Traces
  - Malicious vs. Benign Graphs
    - Connections Labeling
      - Unknown Graphs
  - Graph Mining
    - Significant Malicious Subgraphs
      - Graph Clustering
    - Similar Subgraphs Clusters
  - Template Generalization
    - C&C Activity Templates
  - Template Matching
    - C&C or Benign Connections

Data Set Generation

Template Learning

C&C Activity Detection
System: graph collection

Analysis environment
- Logging: system calls and network API calls
- Tainting: data flows in memory and over the file system

Graph generation
- Input: trace of system and network calls
- Output: a call graph for each successful connection
- Algorithm:
  - Graph root: successful connect and associated sends/recvs
  - Nodes extension: recursive backward dependency over system calls
  - Nodes labeling: call parameters, resource names being abstracted
  - Graph collapsing: collapse duplicate nodes
System: graph collection

Graph generation

- Network: recv
- Systemcall: NtWriteFile
- Arg: Buffer=buf
- Systemcall: NtWriteFile
- Arg: Buffer=buf
- Systemcall: NtWriteFile
- Arg: Buffer=buf
- Systemcall: NtCreateFile
  - FileName: isSystemDirectory/isExecutable
  - DesiredAccess: FileReadAttributes
  - Attributes: AttributeNormal
  - CreateDisposition: FileSupersede
- Arg: FileHandle=FileHandle
  - Network: recv
- Systemcall: NtWriteFile
  - Arg: Buffer=buf
- Systemcall: NtCreateFile
  - FileName: isSystemDirectory/isExecutable
  - DesiredAccess: FileReadAttributes
  - Attributes: AttributeNormal
  - CreateDisposition: FileSupersede
- Arg: FileHandle=FileHandle
  - Systemcall: NtWriteFile
- Collapse: isMultiple
  - Systemcall: NtWriteFile
  - Collapse: isMultiple
System: graph mining

Frequent subgraph mining:

- **Input**: call graphs associated to malicious vs. benign connections
- **Output**: significant subgraphs covering only malicious (C&C) activity
- **Algorithm**:
  - Graph mining: frequent subgraphs from malicious connections
  - Maximization: stripping induced subgraphs from the mined set
  - Set difference: stripping subgraphs included in benign connections
System: graph mining

Frequent subgraph mining

Malicious Dependency Graphs → Frequent Subgraphs → Maximal Subgraphs → Mining Process → Malicious Significant Subgraphs

Frequent Subgraph Mining
Subgraph Maximization
Graph Sets Difference

Benign Dependency Graphs
System: graph clustering and template generation

Graph clustering:
- **Input**: significant malicious subgraphs
- **Output**: clusters group graphs that represent similar activity
- **Algorithm**:
  - Graph similarity: common edges in the maximal common subgraph
  - Graph clustering: clustering by repeated bisection

Template generation:
- **Input**: clusters of similar malicious subgraphs
- **Output**: graph template covering the graphs of the cluster
- **Algorithm**:
  - Template construction: minimal common supergraph
  - Template generalization: supergraph weighted by node frequency
    + Frequent nodes constitute the core activity shared by bots
    + Infrequent nodes constitute optional activity specific to different bots
System: graph clustering and template generation
System: template matching

Template matching:

- **Input**: template, unlabeled collected call graphs
- **Output**: match result
- **Algorithm**:
  - Core matching: subgraph isomorphism with core nodes
    + Mandatory nodes must be present
  - Extended match: maximal common supergraph for optional nodes
    + Isomorphism result used to initialize search
System: template matching

Template matching
Evaluation: dataset presentation

Collected botnet traffic
- 37,572 bot samples corresponding to 745 families (e.g. EgroupDial, Palevo, Virut)
- 130,635 network connections and associated behavior graphs (successful connections only)

Labeling connections for ground truth
- Manually-crafted network signatures: 385 C&C, 162 benign
- 10,801 malicious connections
- 12,367 benign connections
- 66,538 unknown connections
- 40,929 incomplete or irrelevant graphs removed
Evaluation: dataset presentation

Training and testing sets

- Successful Benign Connections
  - Training Set: 8,267
  - Testing Set 1: 4,100
  - Testing Set 2: 66,538

- Successful Malicious Connections
  - (6,539)

- Unknown Connections
  - (66,538)
Evaluation: training the system

System configuration
- Mining frequency threshold: 10%
  - Trade-off between maximum coverage and mining runtime
- Bisection threshold: 60% average and 40% minimal similarity
  - Higher thresholds reduce the effect of generalization

System runtime
- Mining: 16h, Clustering: 4.5h, Generalization: 30min
- Reasonable processing time \textit{wrt.} the NP-hardness of algorithms

Templates quality
- 417 templates generated
  - 397 templates semantically meaningful
- Different types of commands covered
  - Information leakage, download and execute, startup, stealth
Evaluation: testing the system

Testing over labeled connections

- Detection rate: 81.6%
- Detection without the generalization: 66.0%
- Detection of new families that were missing in the training set
- False negatives: 18.4% mainly due to incomplete/infrequent activity
- False positives: 0.2% mainly due to weaker templates
Evaluation: testing the system

Testing over unknown connections

- 66,538 unknown connections
- New matches: 9,464 connections
- New detected families: 193 not covered by network signatures
- New detected variants: missed by outdated network signatures
- False negatives: high proportion of benign traffic (manual verification)
- False positives: 27
## Evaluation: system limitations

### Testing over unknown connections

<table>
<thead>
<tr>
<th>Weakness</th>
<th>Consequences</th>
<th>Potential remediation</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic analysis</td>
<td>Incomplete call logs</td>
<td>Enhanced analysis environment: e.g. multi-path execution</td>
<td>x</td>
</tr>
<tr>
<td>Computational time</td>
<td>Non-termination</td>
<td>Algorithm optimizations: e.g. node labeling, graph collapsing</td>
<td>✓</td>
</tr>
<tr>
<td>Interleaved calls</td>
<td>Noise against mining</td>
<td>System calls selection: e.g. calls with data dependency</td>
<td>✓</td>
</tr>
<tr>
<td>Functional polymorphism</td>
<td>No core activity</td>
<td>Normalizing graphs: e.g. duplicate nodes collapsing, Rewriting rules: e.g. equivalent operations</td>
<td>✓</td>
</tr>
</tbody>
</table>

G. Jacob (UCSB)
Conclusion: Jackstraws

Contributions

- Solution to the problem of identifying C&C traffic from noise
- Automated generation of templates representing C&C behaviors
- Gains provided by the template generalization:
  - Protocol-agnostic representation of C&C activity
  - Increased level of understanding for analysts
  - Coverage extended to families unknown during training