CS259D: Data Mining for Cyber Security
Malicious uses of domain names

- Bots: locate C&C
- Spam/Phishing: URLs linking to scam servers
Detecting malicious domains via DNS: EXPOSURE (2011)

- **Goal:** detect malicious domains
- **Build features using traffic from authoritative DNS servers to recursive DNS servers**
  - Queried domain name, query issue time, TTL, list of IP addresses associated with domain
Features

- **F1: Time-based features**
  - Short life
  - Daily similarity
  - Repeating patterns
  - Access ratio

- **F2: DNS answer-based features**
  - # of distinct IP addresses
  - # of distinct countries
  - # of domains IP shared with
  - Reverse DNS query results

- **F3: TTL value-based features**
  - Average TTL
  - Standard deviation of TTL
  - # of distinct TTL values
  - # of TTL changes
  - % usage of specific TTL ranges

- **F4: Domain name-based features**
  - % of numerical characters
  - % of the length of the LMS
Time-based features

- Global scope: Short-lived
- Local scope:
  - Daily similarity
    - an increase or decrease of request count at same intervals everyday
  - Regularly repeating patterns
    - Instance of change point detection (CPD)
  - Access ratio
    - Idle vs popular
Detecting abrupt changes

- Time series for each domain
  - $P(t)$ = Request count at hour $t$, normalized by max count
  - $P^-(t)$ = Average of past 8 time intervals
  - $P^+(t)$ = Average of next 8 time intervals
  - $d(t) = |P^+(t) - P^-(t)|$
  - Apply Cumulative Sum (CUSUM) algorithm to $d(t)$
    - Detect times $t$, when $d(t)$ is large & is a local maximum
    - $CUSUM(t) = \max\{0, CUSUM(t-1) + d(t) - local\_max\}$
    - Report $t$ as change point if: $CUSUM(t) > cusum\_max$
  - Repeating patterns:
    - Number of changes
    - Standard deviation of the durations of detected changes
Detecting similar daily behavior

• Compute distances of all pairs of daily time series
  ◦ Normalized each time series by its mean and stdv
  ◦ Use Euclidian distance

• \( d_{ij} = \text{Euclidian distance between } i^{\text{th}} \& j^{\text{th}} \text{ days} \)

• \( D = \text{Average of all } d_{ij} \text{ values} \)
DNS answer-based features

- # of distinct IPs
  - Resolved for a domain during the experiment
- # of different countries for those IPs
- Reverse DNS query results of those IPs
- # of domains that share those IPs
  - Can be large for web hosting providers as well
  - Reduce false positives by looking for reverse DNS query results on Google top 3 search results
TTL value-based features

- TTL: Length of time to cache a DNS response
  - Recommended between 1-5 days

- Average TTL value
  - High availability systems
    - Low TTL values
    - Round Robin DNS
    - Example: CDNs, Fast Flux botnets

- Standard deviation of TTL
  - Compromised home computers (dynamic IP) assigned much shorter TTL than compromised servers (static IP)

- # of TTL changes, Total # of different TTL values
  - Higher in malicious domains

- % usage of specific TTL ranges
  - Considered ranges: [0,1), [1,10), [10,100), [100,300), [300,900), >900
  - Malicious domains peak at [0, 100) ranges
Domain name-based features

• Easy-to-remember names
  ◦ Important for benign services
    • Main purpose of DNS
  ◦ Unimportant for attackers (e.g., DGA-generated)

• Features:
  ◦ Ratio of numerical characters to name length
  ◦ Ratio of length of the longest meaningful substring (i.e., a dictionary word) to length of domain name
    • Query name on Google & check # of hits vs a threshold

• Features applied to only second-level domains
  ◦ Example: server.com for x.y.server.com

• Other possible feature: entropy of the domain name
  ◦ DGA-generated names more random than human-generated
Training

- DNS traffic from the Security Information Exchange (SIE)
  - Response data from authoritative name servers in North America & Europe
  - 2.5 months
  - >100 billion DNS queries (1 million queries/minute on average)
  - 4.8 million distinct domain names
- Filtering
  - Alexa top 1000 (20% reduction)
  - Domains older than 1 year (50% more reduction)
Training

- **Malicious domains**
  - 3500 domains
  - Types:
    - Botnet C&C, drive-by-download sites, phishing/scam pages
  - Example Sources:
    - malwaredomains.com, Zeus Block List

- **Benign domains**
  - 3000 domains
  - Example Source: Alexa top 1000

- **Training period**
  - Initial period of 7 days (for time-based features)
  - Retraining every day
Classifier

- C4.5 decision tree algorithm
- Feature selection
C4.5 Primer

- Check for base cases
- For each attribute
  - Compute attribute’s normalized information gain
- Split over attribute with highest gain
- Recurse

- Normalized information gain = difference in entropy of class values
## Classifier Accuracy

<table>
<thead>
<tr>
<th></th>
<th>AUC</th>
<th>Detection Rate</th>
<th>False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full data</td>
<td>0.999</td>
<td>99.5%</td>
<td>0.3%</td>
</tr>
<tr>
<td>10-folds Cross-Validation</td>
<td>0.987</td>
<td>98.5%</td>
<td>0.9%</td>
</tr>
<tr>
<td>66% Percentage Split</td>
<td>0.987</td>
<td>98.4%</td>
<td>1.1%</td>
</tr>
</tbody>
</table>
Testing

- **False positive rate**
  - Filter out domains with < 20 requests in 2.5 months (300,000 domains remaining)
  - 17,686 detected as malicious (5.9%)
  - Hard to verify manually
  - Verification
    - Google searches
    - Well-known spam lists
    - Norton Safe Web
    - McAfee Site Advisor
  - False positive rate: 7.9%

- **Detection rate**
  - 216 domains reported by malwareurls.com & present in dataset
  - 5 had less than 20 queries
  - 211 detected malicious
  - Detection rate: 98%
Evasion

- Assign uniform TTL values across all compromised machines
  - Reduces attacker’s infrastructure reliability
- Reduce # of DNS lookups of malicious domain
  - Not trivial to implement
  - Reduces attacker’s impact
  - Requires high degree of coordination
Administrativia

- Recommended books on website
- Piazza: https://piazza.com/class/i0php4r6eyb43c
- Reading materials for this lecture on website
- Reading material for next lecture on website by tomorrow
Insider threats: Examples

- **Vodafone Greece**
  - Targeted 100+ high-ranking officials
    - Prime minister of Greece & his wife
    - Ministers: national defense, foreign affairs, justice
    - Greek European Union commissioner
    - Mayor of Athens
  - Started before Aug ’04, continued till March ’05
  - Detected accidentally due to rootkit update misconfig
  - Traced to an insider in Vodafone
  - Vodafone fined $76M

- **Edward Snowden**
Insider threats

• “Despite some variation from year to year, inside jobs occur about as often as outside jobs. The lesson here, though, surely is as simple as this: organizations have to anticipate attacks from all quarters.”

CSI/FBI COMPUTER CRIME AND SECURITY SURVEY 2005
Types of insider attackers

- **Traitors**
  - A legitimate user with proper access credentials gone rogue
  - Full knowledge of systems & security policies

- **Masqueraders**
  - An attacker who has stolen/obtained and uses credentials of a legitimate user
Types of insiders attacks

- Unauthorized extraction, duplication, or exfiltration of data
- Tampering with data (unauthorized changes of data or records)
- Destruction and deletion of critical assets
- Downloading from unauthorized sources or use of pirated software which might contain backdoors or malicious code
- Eavesdropping and packet sniffing
- Spoofing and impersonating other users
- Social engineering attacks
- Misuse of resources for non-business related or unauthorized activities
- Purposefully installing malicious software
Insider threats: defense

• Masqueraders
  ◦ Behavioral profiling & anomaly detection
  ◦ Requires extensive logging of systems & users
    • Host-based
      • Pros: Better coverage
      • Cons: Most insider attacks at application level not network level
    • Host-based
      • Cons: hard to deploy

• Traitors
  ◦ Decoys/traps (e.g., honeypots, honeytokens)
Insider attack detection: Types of audit data

- CLI command sequences
- System calls
- Database/file accesses
- Keystroke dynamics
- Mouse dynamics
User behavior modeling using unix shell commands

- **Multi-class classification**
  - Data from each user as samples from one class
  - Self vs non-self
  - Require retraining as users join/leave organization
  - Non-self samples bias model’s view of masquerader

- **Single-class classification**
  - Builds a profile for user only using that user’s data
  - Requires less data
  - Distributed implementation
Schonlau dataset

- Unix shell commands of 70 users
  - Collected using Unix acct
  - 50 random users as intrusion targets
  - 20 masquerade users
- 15,000 commands per user
  - Over days or months
  - First 5,000 commands clean
  - Next 10,000 commands randomly injected with 100-command intrusion blocks
  - Blocks of size 100: clean or dirty
- Goal: detect dirty blocks
- Issues
  - Widely different time periods for different users
  - Different number of login sessions per user
  - Different number (0-24) of intrusion blocks per users
  - User job functions unknown
  - acct logs commands in the order they finished
One-class classification

- One-class Naïve Bayes
- One-class SVM
Naïve Bayes Classifier

- **Bayes rule:**
  - For user u, block d: \( p(u|d) = \frac{p(u)p(d|u)}{p(d)} \)
- Different commands assumed independent
- **Multi-variate Bernoulli model:**
  - Total of N unique commands (N=856 for Unix)
  - Each block as a binary N-dimensional vector
  - Each dimension with Bernoulli model
  - Performs better at small vocabulary sizes
- **Multinomial model**
  - Each block as N-dimensional vector
  - Each feature = # of occurrences of command
  - Performs better at large vocabulary sizes
Multivariate Bernoulli model

- $N(u) = \text{number of training blocks for user } u$
- $N(c_i, u) = \text{number of blocks containing } c_i \text{ for user } u$
- Laplacian prior:
  \[ p(c_i|u) = \frac{1 + N(c_i, u)}{2+N(u)} \]
- $p(d|u)$ computed from $p(c_i|u)$ values and the independence assumption
Multinomial model

• Laplacian prior:

\[ p(c_i | u) = \frac{\sum_{j=1}^{N(u)} n_i(d_j) + \alpha}{\sum_{i=1}^{N} \sum_{j=1}^{N(u)} n_i(d_j) + \alpha N} \]

• \( p(d|u) \) computed from \( p(c_i|u) \) & independence
One class Naïve Bayes

- Compute $p(c_i | u)$ only for user’s self profile
- For masquerader, assume each command has probability $1/N$ (completely random)
  - Makes no assumption about masquerader
- Given a block $d$, compute:
  $$\frac{p(d|\text{self})}{p(d|\text{non-self})}$$
- Threshold controls false positive vs detection rate
One class SVM

- Map data to a high-dimensional feature space
- Maximally separate data points from origin
- Allow some outliers, but probability of lying on the wrong side bounded by a parameter
Multivariate vs Multinomial

![Graph showing hit rate versus false positive with two curves: one-class Multinomial and one-class Bernoulli]
One-class vs two-class
One-class SVM vs other algorithms

- One step Markov, 6.7% FP
- Multinomial, no updating, 4.6% FP
- Multinomial, updating, 1.3% FP
- Hybrid Multi-step Markov, 3.2% FP
- IPAM, 2.7% FP
- Uniqueness, 1.4% FP
- Sequence match, 3.7% FP
- Compression, 5.0% FP

hit rate vs false positive
References

  (https://www.iseclab.org/papers/bilge-ndss11.pdf)

- The Athens Affair
  (http://spectrum.ieee.org/telecom/security/the-athens-affair)


- One-class Training for Masquerade Detection (2003)