Generalization and Characterization Techniques for the Anomaly-based Detection of Web Attacks

2008. 05. 26

Distributed Multimedia Computing Lab.

Minjae Cho

popeye@ece.hanyang.ac.kr

Paper Information

- **Title**
  - **Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks**




- **Authors**
  - William Robertson, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer
  - **Computer Security Group (formerly Reliable Software Group)**
  - **Dept. of Computer Science, University of California, Santa Barbara**

- **Published**
  - **13th Network and Distributed System Security Symposium(NDSS06)**
  - **San Diego, CA. Feb. 2006**
  - **http://www.isoc.org/**

Outline

- Introduction
  - Web Applications
  - Misuse vs. Anomaly Detection
  - System Example
- Architectures
  - Anomaly Detector
  - Anomaly Models
  - Anomaly Generalization
  - Attack Class Inference
- Evaluation
  - False Positives
  - Performances
- Conclusions
  - Conclusions
  - Future Work

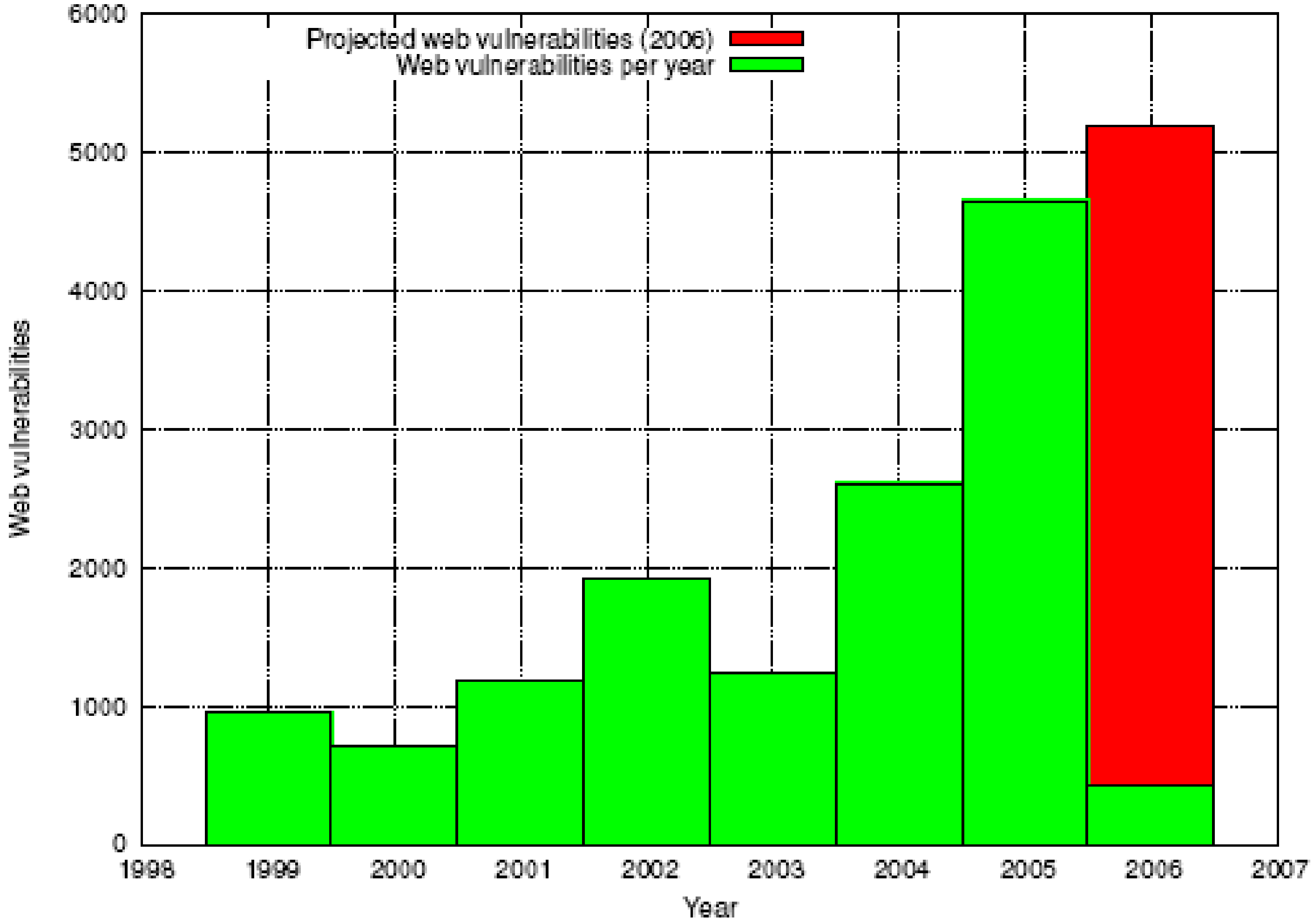
Why are web applications important?

- **Web has become a ubiquitous application delivery medium**
- **Easy to develop, deploy, and access web application**

- **Unfortunately, also easy to introduce critical security vulnerabilities**
  - **Lack of awareness of security issues**
  - **Feature-driven development**
  - **Time-to-market constraints**

Result
Web application vulnerabilities are increasing in number and severity

Reported web-related vulnerabilities [CVE Database]



Why anomaly detection?

- Misuse detection performs well in detecting known attacks directed at widely-deployed targets
- Many web applications, however, are custom-developed and possibly deployed at a handful of sites
- Consequently, custom misuse signatures required
  - Signature development costly, error-prone, and almost never done

- Anomaly detection is able (*in theory*) to detect novel attacks against custom code
  - Learns (or provided) a profile of normal behavior, and then detects deviations from established profile

Limitations of existing anomaly detectors

- **prone to producing false positives**
- **no indication as to the *nature* of a detected attack**

Objective
Prosing *generalization* and *characterization* techniques to mitigate these two shortcomings

Generalization and Characterization example

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=256)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=260)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID cdist (dom={0x90})]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID structure (0Aa)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=264)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID structure (0a)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID cdist (dom={0x90,0x41})]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID structure (0Aa)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=256)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID cdist (dom={0x90,0x41})]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID cdist (dom={0x90})]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=260)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=264)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID structure (0a)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=256)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID cdist (dom={0x90})]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=256)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=256)]

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID structure (0a)]

.....

Generalization and Characterization example

Cross-site scripting

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?file attribute length exceeded (len=256)]

...

Buffer overflow

ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID cdist (dom-{0x90})]

...

Cross-site scripting

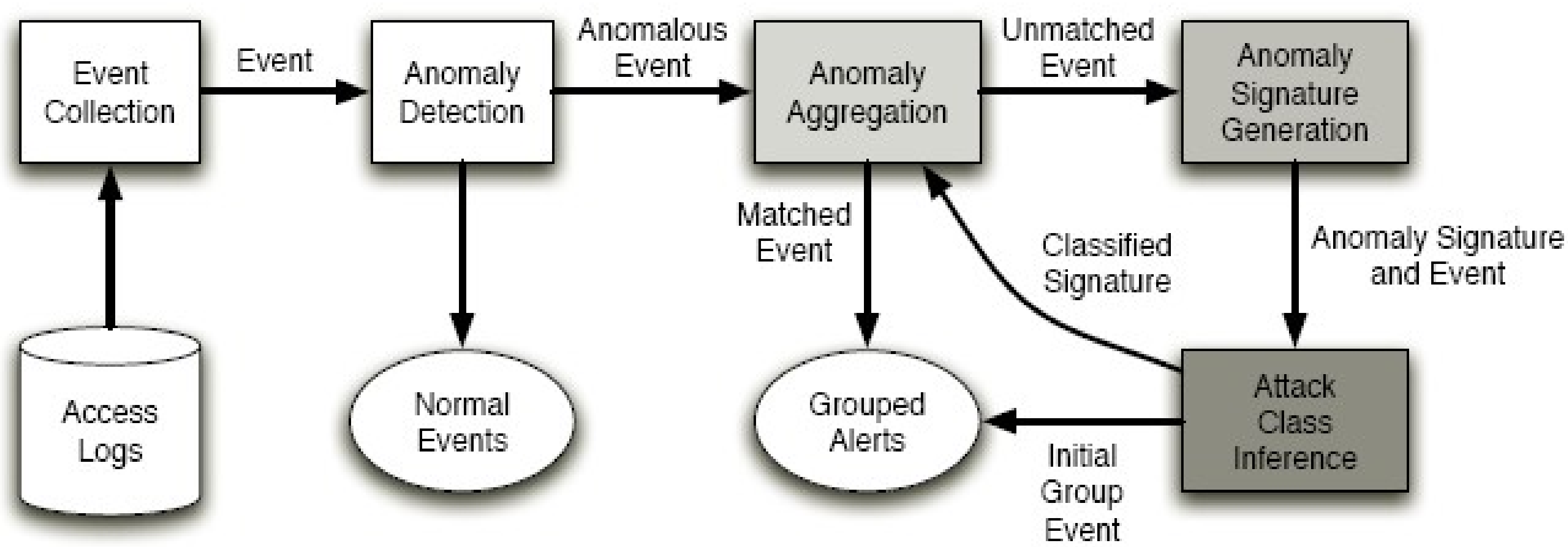
ALERT 10.0.0.3 -> 10.0.0.1 [/cgi-bin/show.cgi?sID structure (0a)]

...

Outline

- Introduction
  - Web Applications
  - Misuse vs. Anomaly Detection
  - System Example
- Architectures
  - Anomaly Detector
  - Anomaly Models
  - Anomaly Generalization
  - Attack Class Inference
- Evaluation
  - False Positives
  - Performances
- Conclusions
  - Conclusions
  - Future Work

Architectural overview



Architectural overview

- **Anomaly detector**
  - **Learns profiles of normal application behavior and detect deviations**
- **Anomaly signature generator**
  - **Generates anomaly *signatures* to group "similar" alerts**
- **Anomaly aggregator**
  - **Groups anomalies according to model-specific similarity operations**
- **Attack classifier**
  - **Characterizes types of attacks anomalies may represent**

Anomaly detection component

- **Examines web requests sent from clients to server**
  - **Application to be executed**
  - **Application parameters (attribute names and values)**

Example
GET /cgi-bin/show.cgi?sID=12345&file=images/foo.png

- **Applies statistical models to each attribute of each application in two phases**

  - Learning : Builds profiles of normal behavior for each application parameter

  - Detection : Detects deviations from learned profile

Anomaly detection – Attribute length model

Chebyshev inequality

$$p\left(|x-\mu| > |l-\mu|\right) > p\left(l\right) = \frac{\sigma^2}{(l-\mu)^2}$$

μ : mean
σ : variance

Observation

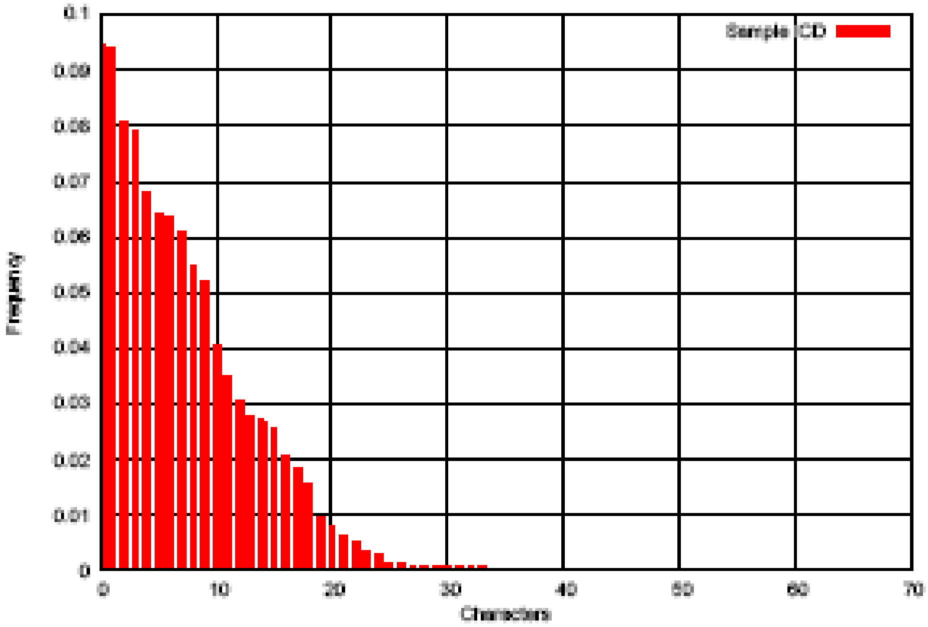
Many attribute values are either fixed in size or vary over a small range

- **Model attempts to approximate actual(unknown) distribution of attribute lengths**
- **Weak bound results in significant tolerance to variations**

Anomaly detection – Character distribution

Observation

Many attribute take values that have similar character distribution

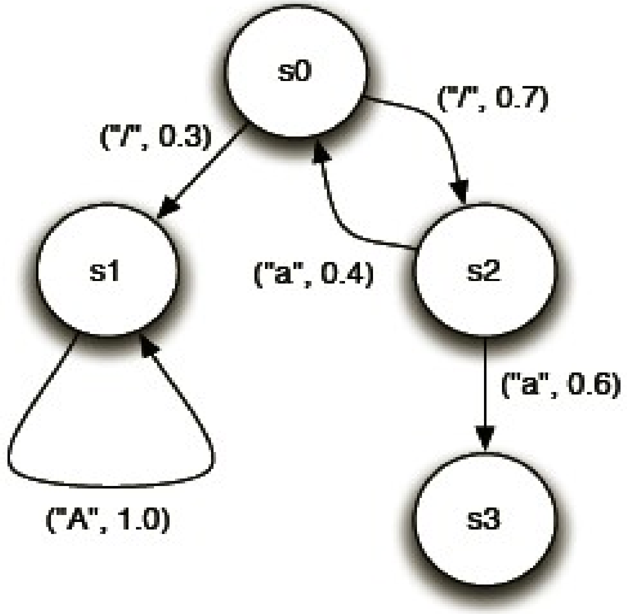


- **Model creates *idealized character distribution* (ICD) for attribute**
- **Anomaly score calculated using variant of Pearson χ^2 test**

Anomaly detection – Structural inference

Observation

Many attribute values can be modeled as strings generated by a regular grammar

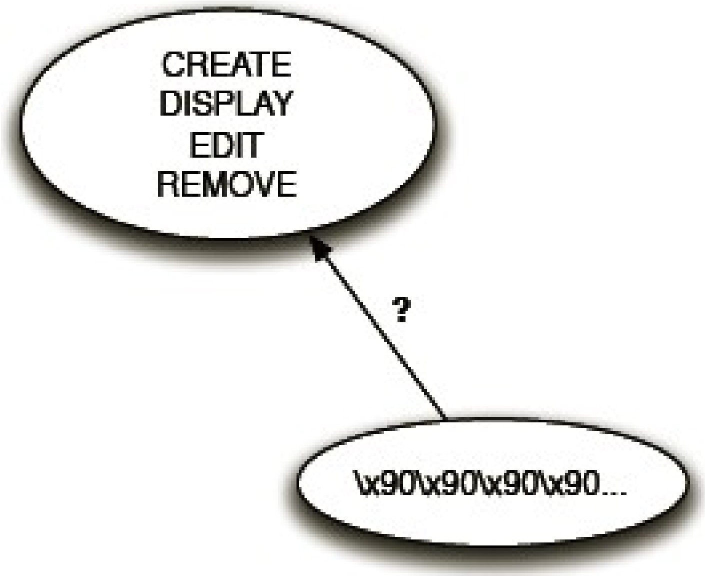


- **Model constructs *probabilistic grammar* from learning set**
- **Anomaly score calculated as product of transition probabilities along path through NFA for a given value**

Anomaly detection – Token finder

Observation

Many attribute takes values that are drawn from a small set of constants



- **Model is applied to an attribute if set of unique values observed during learning phase does not grow proportionally to number of requests**
- **Detection performed by membership test of observed value in learned set of values**

Anomaly generalization component

- **Anomaly generalization**
  - **Construction of an abstract model that matches initial anomaly and similar anomalies**

- **Parameters for each alerting model for a given anomaly are "relaxed" in a model-specific fashion**
- **Resulting set of relaxed models are composed to create an *anomaly signature***
- **Model-specific *similarity operation* used to determine if subsequent anomalies are similar to the initial anomaly**

Anomaly generalization – Attribute length

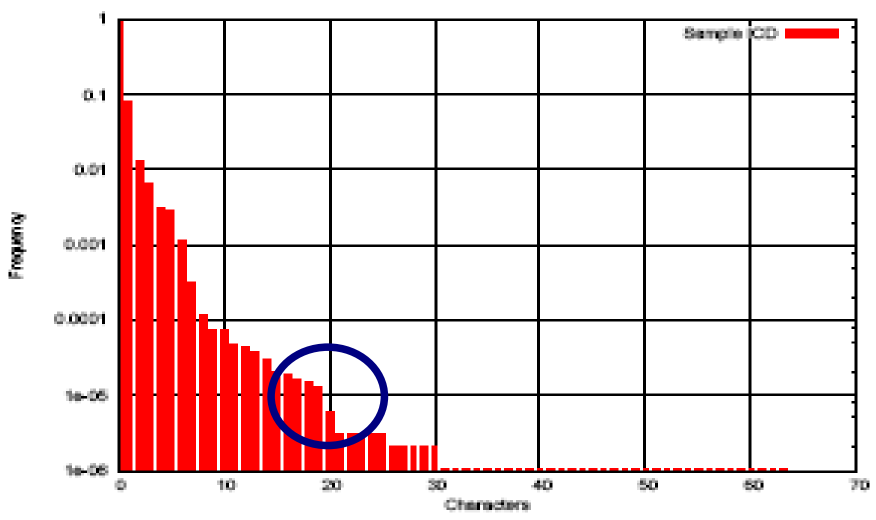
Similarity operator

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

- **Parameters to attribute length model are extracted**
- **Similarity operation $\psi_{attrlen}$ used to determine if subsequent attribute lengths l_{obsv} are within distance $d_{attrlen}$ from the original anomalous length l_{orig}**

Anomaly generalization – Character distribution

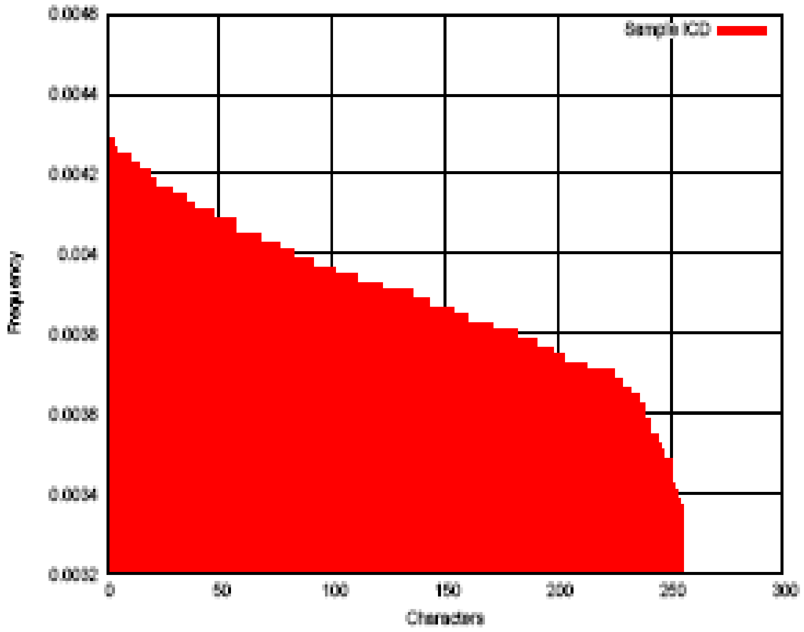
- **Anomalous character distribution may exhibit a sharp drop-off, indicating a small set of dominating characters**



- **Set of dominating character and relative frequency pairs is extracted**
- **Similarity operation ψ_{cdist} tests whether character distributions share at least one dominating character with relative frequencies at most d_{cdist} apart**

Anomaly generalization – Character distribution

- Anomalous character distribution may be loosely approximated by the uniform distribution



- Set of character and relative frequency pairs is extracted from anomalous ICD
- Similarity operation ψ_{cdist} determines whether the maximum distance between any pair of frequency values from initial and observed character distribution is at most d_{cdist}

Anomaly generalization – Structural inference

Similarity Operator

$$\psi_{structure} \equiv lex\left(s_{orig}, s_{obsv}\right) < d_{structure}$$

- **Prefix of anomalous value, including the first occurrence of an anomalous character, is extracted and normalized according to character class**
- **Subsequent anomalous values are normalized in the same manner to s_{obsv} and then compared with s_{orig}**
- **Lexicographical similarity function may be string equality or a string similarity metric**

Anomaly generalization – Token finger

Similarity Operator

$$\psi_{token} \equiv lex\left(l_{orig}, l_{obsv}\right) < d_{token}$$

- **Anomalous value that failed membership test is extracted**
- **Lexicographical similarity function (as in case of structural inference) test whether subsequent anomalous values are similar to initial anomaly**

Attack class inference

- Intended to address concern that traditional anomaly detector can detect attacks, but cannot easily explain "WHY"
- Observed that well-known classes of attacks deviate from learned profiles in consistent manner
- Component employs *ad hoc heuristics* to infer the type of attack represented by a detected anomaly
- Heuristics search for general features of a type of attack, not specific attack
  - directory traversal
  - cross-site scripting
  - SQL injection
  - buffer overflows

Attack class inference – Directory traversal

Example

GET /cgi-bin/show.cgi?sID=12345&file=../../../../maillog

- Attacker attempts to access unauthorized files by traversing directory tree
- Heuristic activated
  - if ICD dominated by "." or "/"
  - if structural inference model determines underivable character to be "." or "/"
- Heuristic scans for directory traversal characters

Attack class inference – Cross-site scripting

Example

GET /cgi-bin/show.cgi?sID=12345&file=<script>...</script>

- **Attacker attempts to embed scripts in pages served by vulnerable server to be executed by other clients**
- **Heuristic activated**
  - **if any of structural inference, character distribution, or token finder models generates an alert**
- **Heuristic scans for common syntactic elements of JavaScript or HTML**

Attack class inference – SQL injection

Example

GET /cgi-bin/show.cgi?sID=' or 1=1;--&file=access.log

- **Attacker attempts to execute arbitrary SQL queries by exploiting lack of input sanitization**
- **Heuristic activated**
  - **if structural inference model generates an alert**
- **Heuristic scans for SQL language keywords and escape characters**

Attack class inference – Buffer overflows

Example

GET /cgi-bin/show.cgi?sID=12345&file=%90%90%90%90%90...

- **Attacker attempts to influence control flow of application or corrupt data by overflowing a buffer**
- **Heuristic activated**
  - **if character distribution, structural inference, or attribute length model generate an alert**
- **Variety of techniques may be used to detect executable code (e.g., non-ASCII characters, abstract payload execution, speculative disassembly, etc.)**

Outline

- Introduction
  - Web Applications
  - Misuse vs. Anomaly Detection
  - System Example
- Architectures
  - Anomaly Detector
  - Anomaly Models
  - Anomaly Generalization
  - Attack Class Inference
- **Evaluation**
  - **False Positives**
  - **Performances**
- Conclusions
  - Conclusions
  - Future Work

Evaluation

- Off-line processing of web server logs collected from TU Vienna and UCSB
- Known attacks stripped from data set and attack variations
- System evaluation
  - false positive rate
  - ability to group and characterize alerts
  - performance

Evaluation – False positive rate

Data set	Queries	FP	FPR	FP/day	Groups	Grouped FPR
TUV	737,626	14	1.90×10^{-5}	0.26	2	3.00×10^{-6}
UCSB	35,261	513	1.45×10^{-2}	1.89	3	8.50×10^{-5}

- **Low initial false positive rate**
- **Initial false positive rate further reduced through anomaly aggregation**

Evaluation – grouping and characterization

Attack	Mutants	Groups	Alerting models	Characterization
csSearch	10	1	Length, CDist	XSS
htmlscript	10	1	Length, Structure	Directory traversal
imp	10	1	Length, CDist	XSS
phorum	10	1	Length, CDist, Token	Buffer overflow
phpnuke	10	1	Length, Structure	SQL injection
webwho	10	1	Length	None

- **All attack mutations detected**
- **Most attacks accurately characterized**

Evaluation – Performance [time]

Data set	Requests	Request rate	Elapsed time	Analysis rate
TUV	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

- **Deployable in standalone mode for low to medium traffic sites**
- **Cluster configuration possible for higher traffic sites**

Conclusions

- Major limitations of anomaly detectors can be mitigated through use of *generalization* and *characterization* techniques
  - Constructing an abstract description of an anomaly allows system to group similar anomalies, reducing effective false positive rate
  - Inferring the nature of an anomaly assists administrators and developers in analyzing & patching novel vulnerabilities
- Anomaly aggregation component successfully grouped both false positive and mutated attacks in real-world data sets
- Attack inference component successfully characterized most attacks

Future work

- Apply the techniques to additional models
- Investigate alternative techniques for generalizing anomalies
- Improve attack inference technique
  - More sophisticated heuristics
  - Apply Bayesian techniques to infer attacks based on model outputs as evidence nodes
- Extend system to other domains (e.g., syscall arguments)

Q&A

- **Any questions?**