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

William Robertson
<wkr@cs.ucsb.edu>

Reliable Software Group
UC Santa Barbara

13^{th} Network and Distributed System Security Symposium
San Diego, CA
February 2006

Why are web applications important?

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

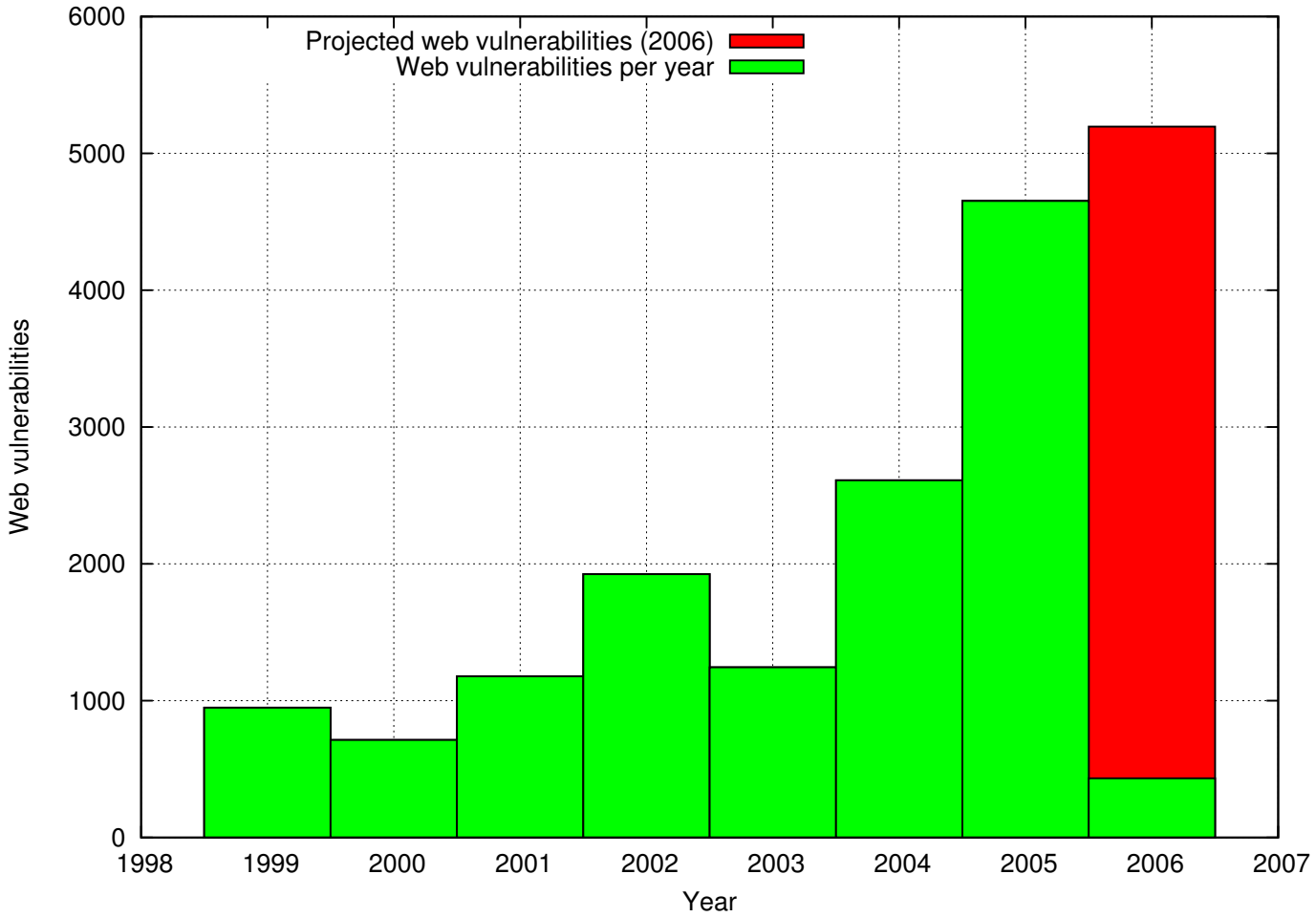
Why are web applications important?

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

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

Objective

We propose *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

```
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

```
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

```
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})]

...
```

Buffer overflow

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

...
```

Outline

Architectural overview

Previous work

Anomaly detector Learns profiles of normal application behavior and detect deviations [Kruegel03]

Contributions

Anomaly signature generator Generates *anomaly signatures* to group "similar" alerts

Anomaly aggregator Groups anomalies according to model-specific *similarity operators*

Attack classifier Characterizes types of attacks anomalies may represent

Architectural overview

Previous work

Anomaly detector Learns profiles of normal application behavior and detect deviations [Kruegel03]

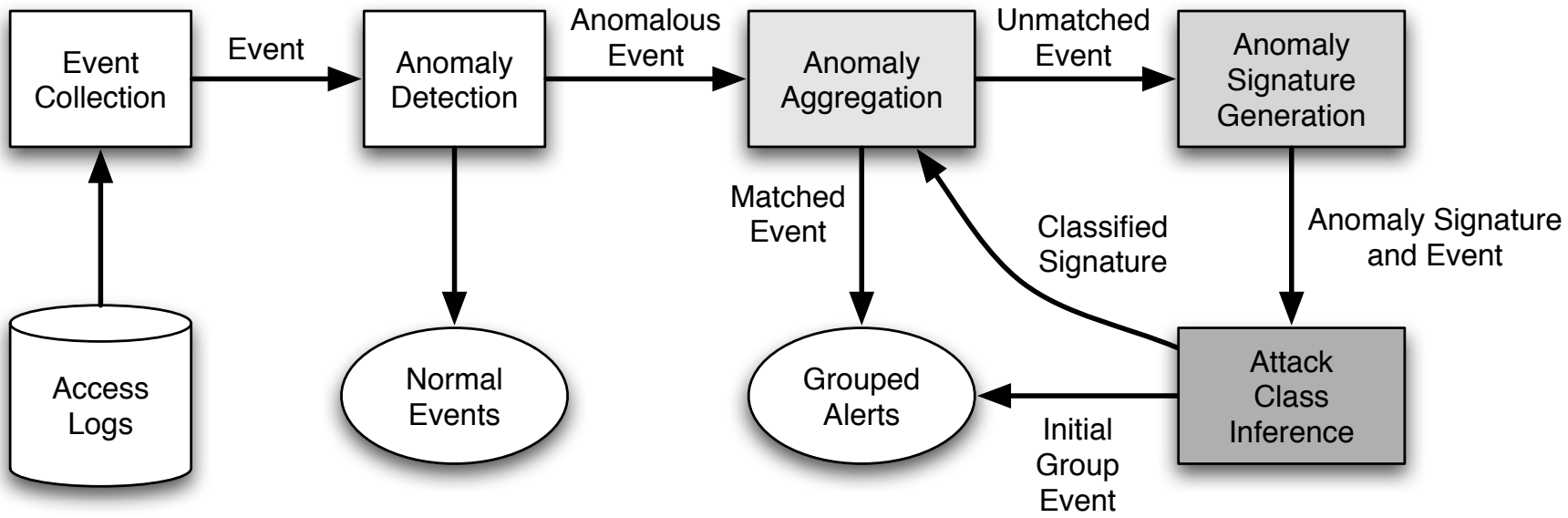
Contributions

Anomaly signature generator Generates *anomaly signatures* to group "similar" alerts

Anomaly aggregator Groups anomalies according to model-specific *similarity operators*

Attack classifier Characterizes types of attacks anomalies may represent

Architectural overview



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 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 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 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

Attribute length model

Chebyshev inequality

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

Observation

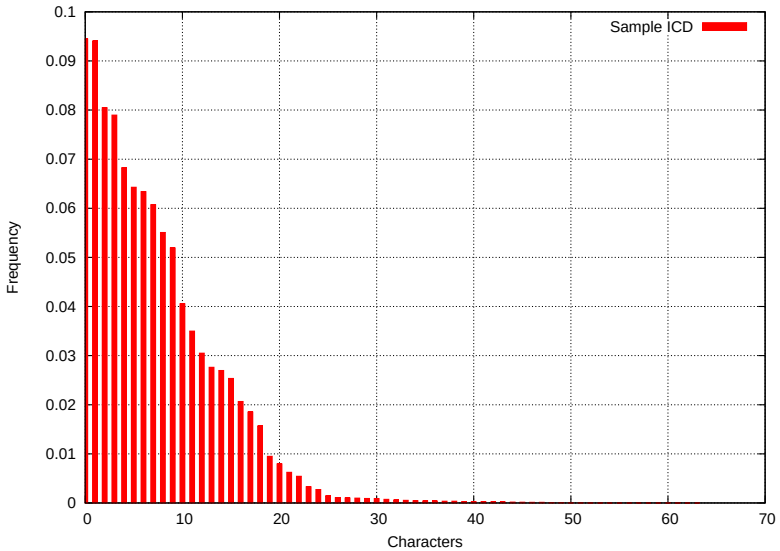
Many attribute values are either fixed in size or vary over a small range

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

Character distribution model

Observation

Many attributes take values that have similar character distributions

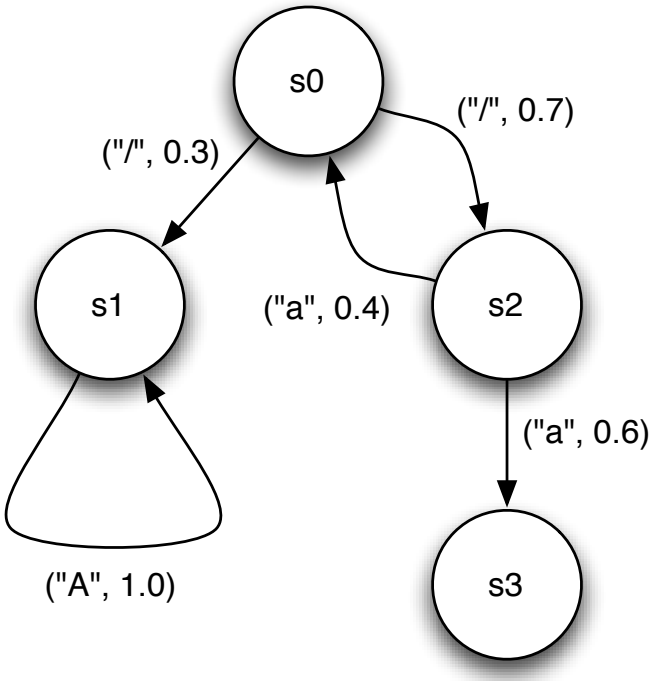


- Model creates *idealized character distribution* (ICD) for attribute
- Anomaly score calculated using variant of Pearson χ^2 test

Structural inference model

Observation

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

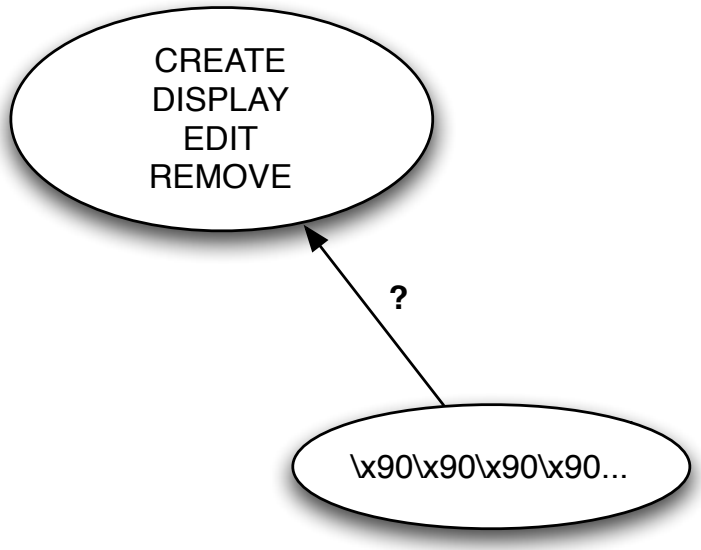


- Model constructs *probabilistic grammar* from learning set
- Anomaly score calculated as product of transition probabilities along path through NFA for a given value

Token finder model

Observation

Many attributes take 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 operator* used to determine if subsequent anomalies are similar to the initial anomaly

Attribute length generalization

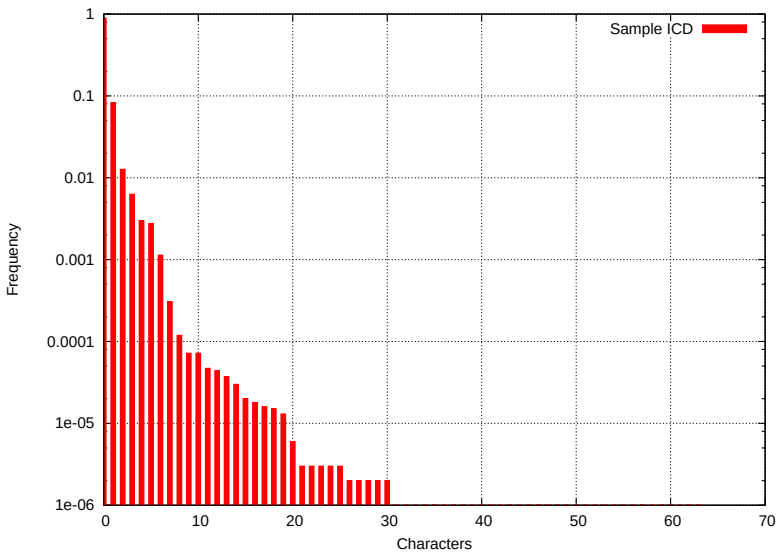
Similarity operator

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

- Parameters to attribute length model (sample mean μ and variance σ^2) are extracted
- Similarity operator $\psi_{attrlen}$ used to determine if subsequent attribute lengths l_{obsv} are within distance $d_{attrlen}$ from the original anomalous length l_{orig}

Character distribution generalization [dominators]

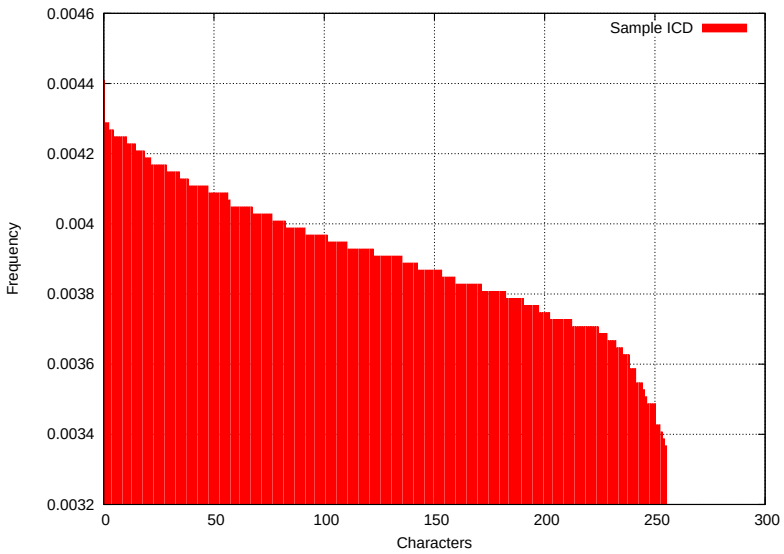
- Anomalous character distribution may exhibit a sharp drop-off, indicating a small set of dominating characters



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

Character distribution generalization [uniform]

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



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

Structural inference generalization

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 (e.g., Hamming, Levenshtein distance)

Token finder generalization

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) tests whether subsequent anomalous values are similar to initial anomaly

Attack class inference

- Intended to address concern that traditional anomaly detectors 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 attacks
  - directory traversal
  - cross-site scripting
  - SQL injection
  - buffer overflows

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 "/" or if structural inference model determines underivable character to be "." or "/"
- Heuristic scans for directory traversal characters

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 "/" or if structural inference model determines underivable character to be "." or "/"
- Heuristic scans for directory traversal characters

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

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

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

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

Buffer overflows

Example

```
GET /cgi-bin/show.cgi?sID=12345&file=%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 models generate an alert
- Variety of techniques may be used to detect executable code (e.g., non-ASCII characters, abstract payload execution, speculative disassembly, etc.)

Buffer overflows

Example

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

Experimental setup

- Offline processing of web server logs collected from TU Vienna and UCSB
- Known attacks stripped from data sets, and attack variations injected using Sploit framework
- Detection performed on Pentium IV 1.8 GHz machine with 1GB of RAM
- System evaluated in terms of false positive rate, ability to group and characterize alerts, and performance

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

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

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

Attack 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

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

Related work

Learning-based anomaly detection

- Building profiles of user login times and actions [Denning87]
- Clustering techniques applied to unlabeled network traces [Portnoy01]

Application-level intrusion detection

- Syscall sequences [Forrest96]
- Learning of network application behavior [Mahoney02]
- Service-specific network application anomaly detection [Kruegel02]

Related work

Detection of attacks against web servers

- Misuse-based analysis of web server logs [Almgren00]
- Web server-integrated misuse-based attack detection [Almgren01]
- Misuse-based analysis of multiple event streams [Vigna03]
- Serial combination of misuse and anomaly-based detectors [Tombini04]

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 and patching novel vulnerabilities
- Anomaly aggregation component successfully grouped both false positives and mutated attacks in real-world data sets
- Attack inference component successfully characterized most attacks

Future work

- Apply generalization, characterization 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)