> reverse benchmarking|

Hacking the EULA: Reverse Benchmarking Web Application Security Scanners

Overview

Each year thousands of work hours are lost by security practitioners sorting through web application security reports separating out erroneous vulnerability data. Individuals must currently work through this process in a vacuum, as there is little to no publicly available information that is helpful. Compounding this situation, restrictive vendor EULAs (End User License Agreements) prohibit publishing of research about the quality of their signature base. Due to these agreements, a chilling effect has discouraged public research into the common types of false positives that existing commercial web application scanner technologies are prone to exhibit.

About the speakers…

Tom Stracener, Sr. Security Analyst, **Cenzic Inc**.

Marce Luck, Information Security Architect, Fortune 100 company

Introduction

Web Application Security Scanning Technology

The Problem of False Positives

What is Reverse Benchmarking?

Reverse Benchmarking Methodology

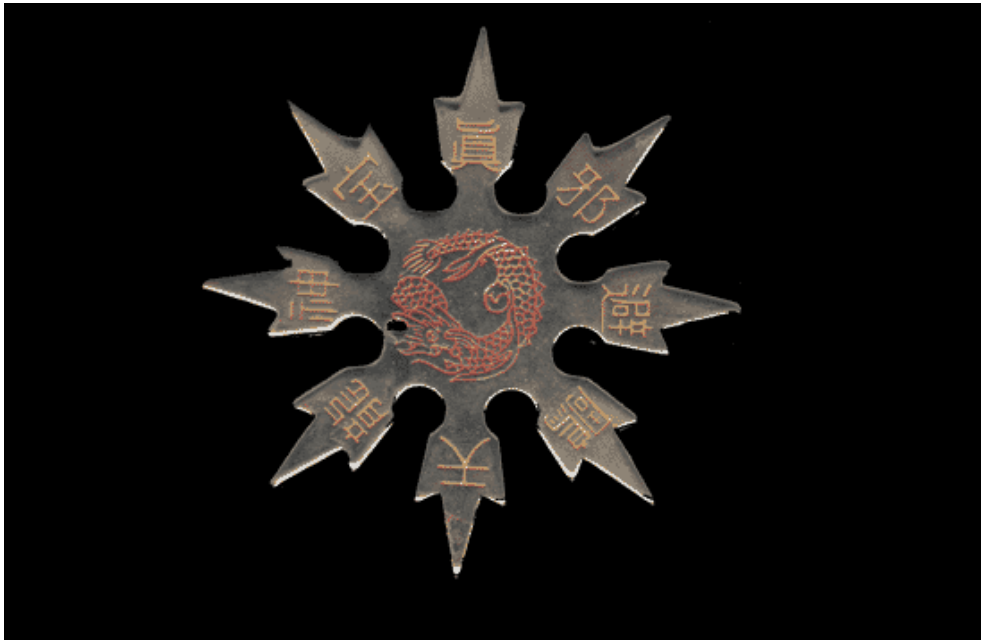
10 Common False Positive Types

Further Research

Business as usual…

B) **You shall not**: (i) use the Software or any portion thereof except as provided in this Agreement; (ii) modify, create derivative works of, translate, **reverse engineer**, decompile, disassemble (except to the extent applicable laws specifically prohibit such restriction) or **attempt to derive the Source Code of the Software** provided to You in machine executable object code form; (iii) market, distribute or otherwise transfer copies of the Software to others; (iv) rent, lease or loan the Software; (v) **distribute externally or to any third party any communication that compares the features, functions or performance characteristics of the Software with any other product of You or any third party**; or (vi) attempt to modify or tamper with the normal function of a license manager that regulates usage of the Software.

What is Reverse Benchmarking?



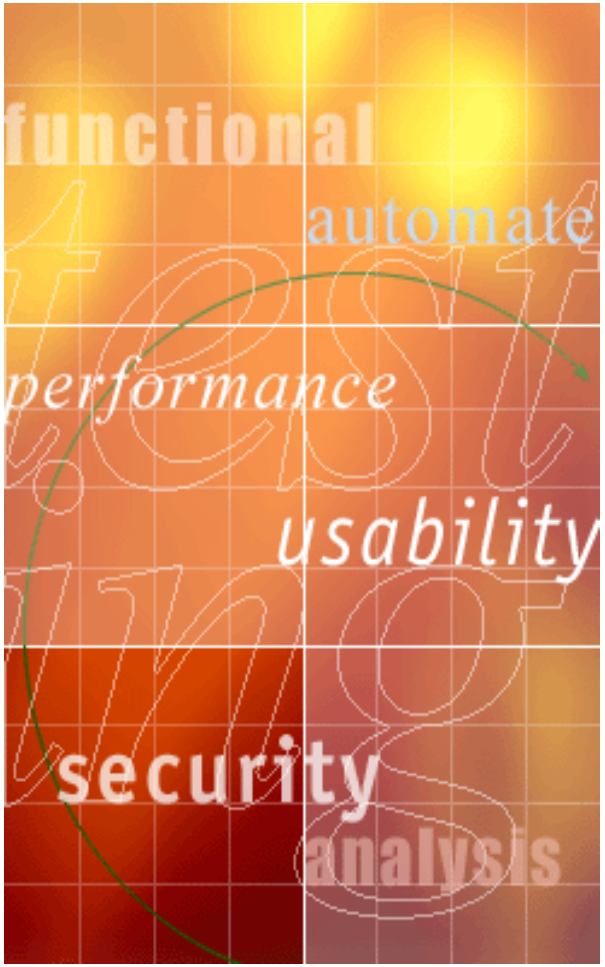
Whoever is first in the field and awaits the coming of the enemy, will be fresh for the fight; whoever is second in the field and has to hasten to battle will arrive exhausted.

-- Sun-Tzu “The Art of War”

> reverse benchmarking|

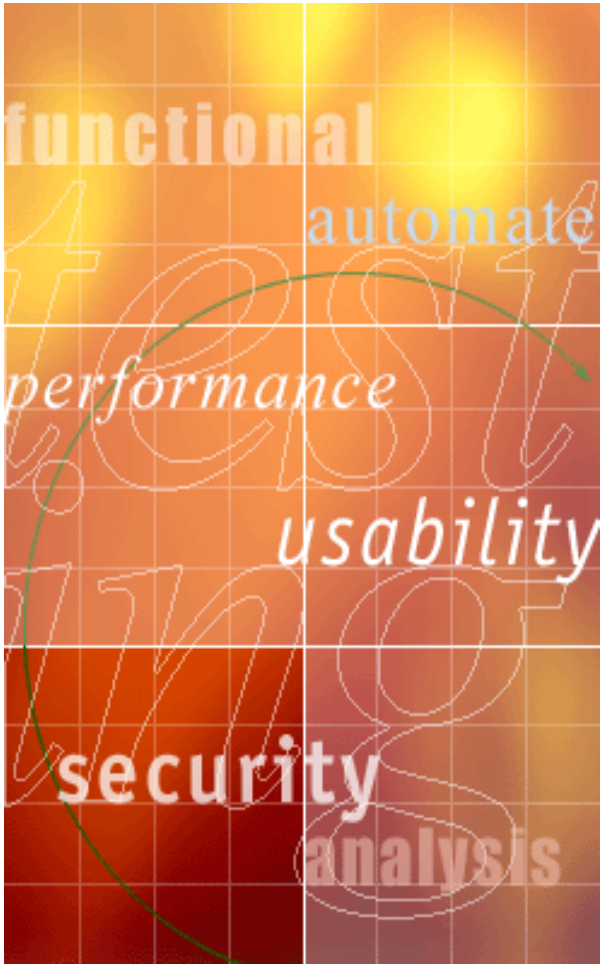
Analyzing Application Security Scanners

- Security Assessment ‘quality’ criteria
  - Functionality (Black vs White Box)
  - Ergonomics & Usability
  - Performance
  - Feature Sets
  - Bling
  - **Accuracy**
  - **False Positive Rates i.e. Signal to Noise**

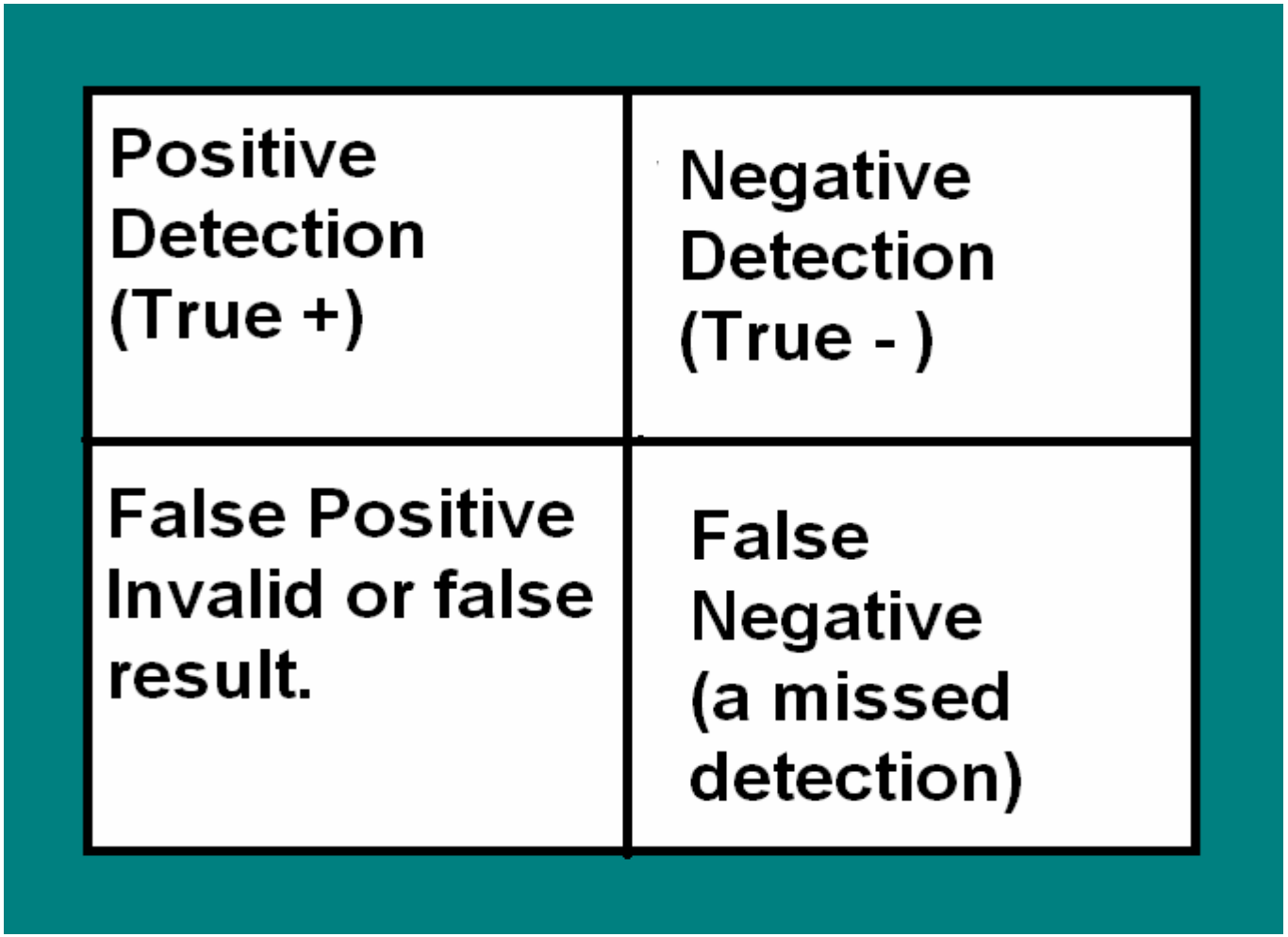


Analyzing Application Security Scanners

- Benchmarking Concepts
  - Benchmarking black box scanners is ultimately a systematic comparison
  - Most common Benchmarking technique is ‘positive’ or ‘comparative’ benchmarking
  - The goal is to see which scanner does the best against a selected application



Positive and Negative Accuracy concepts



+ Benchmarking: Accuracy

Positive Benchmarking is a measure of the number of valid results relative to the total number of vulnerabilities in the application.

- **Example:** Scanner Foodizzle found 8 out of 10 vulnerabilities in the target application, i.e. it was 80% relative to the vulnerability-set.
- **Use:** Measures of ‘accuracy’ are commonly used during positive benchmarking, bake-offs, etc.
- **Challenges:** Accuracy is difficult to measure because its often difficult to know exactly how many vulnerabilities there are in the target application.

+ Benchmarking Limitations

Positive Benchmarking relies on objective knowledge of vulnerabilities in the target application, and thus breaks down when not performed by experts

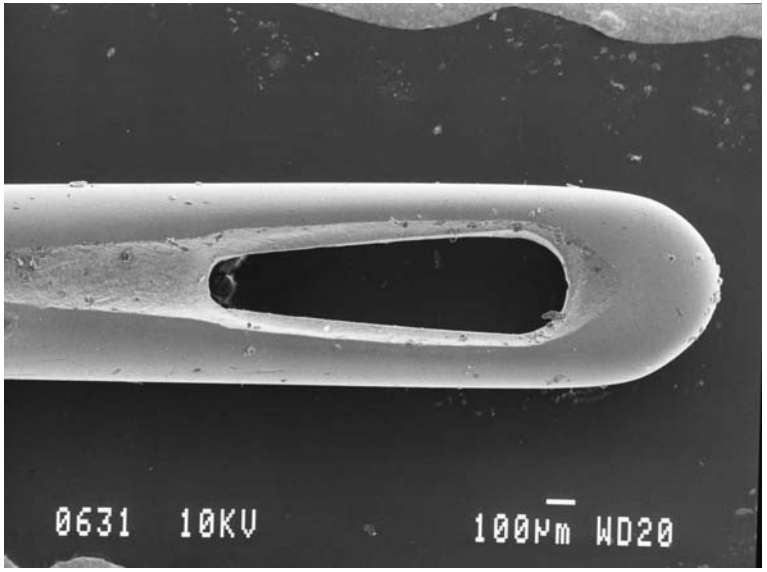
- **Selection Bias:** Scanner Foodizzle found 8 out of 10 vulnerabilities in the target application, i.e. it was 80% relative to the vulnerability-set.
- **Interpreting the Data:** Measures of ‘accuracy’ are commonly used during positive benchmarking, bake-offs, etc.
- **Tuning Against the App:** Accuracy is difficult to measure because its often difficult to know exactly how many vulnerabilities there are in the target application.
- **Analysis Gaps:** typical bake-off or benchmarking methods do not rigorously test important scanner characteristics like the spider, because the spider is only indirectly related to accuracy

+ Benchmarking Limitations

Positive Benchmarking relies on objective knowledge of vulnerabilities in the target application, and thus breaks down when not performed by experts

- **Tuning Against the App:** Its not uncommon for vendors to download the well-known sample applications and ‘tune’ their technology to detect most or all of the security issues.

Reverse Benchmarking enlarges the eye of the EULA needle…



What is Reverse Benchmarking?

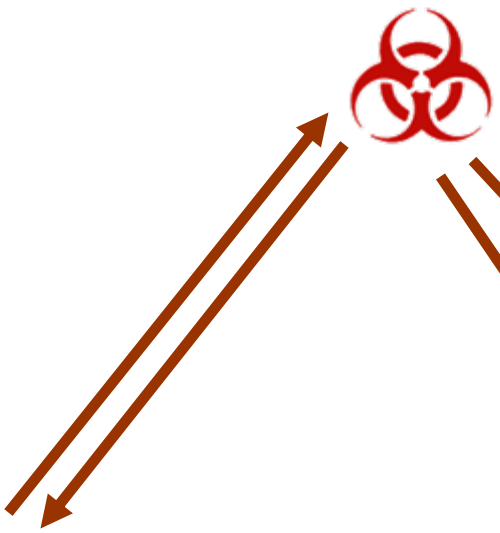
- It's a type of passive Reverse Engineering.
- Its Designed to Kick a Scanner's Ass™
- Causes Massive False Positives
- Facilitates an understanding vulnerability detection methods
- Think of it as Detection Logic Fuzzing

- Exposes poor coding, faulty detection logic
- Reveals Security Testing design flaws
- Confuses Stateless Testing Mechanisms

Rationale for Reverse Benchmarking

- Most of the Common False Positive Types have been around since 1999-2000
- Most testing mechanisms are entirely stateless and have evolved little
- Very little is known about False Positives, as a science
- *There are no taxonomies or Top 10 lists for Common False Positive Types*

Reverse Benchmark Target



Enumerates and Categorizes False Positive Types

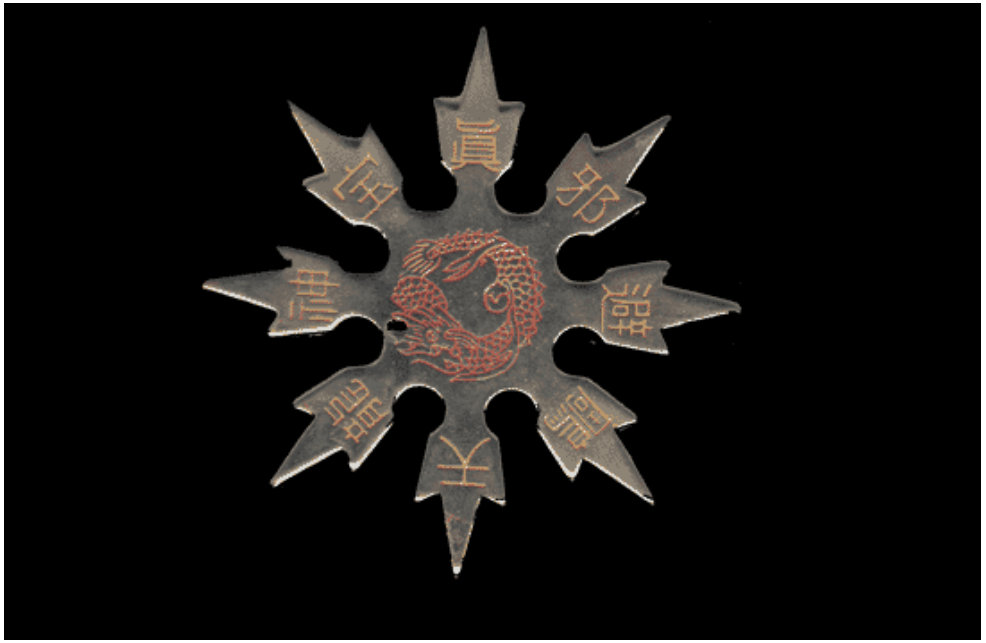
Reveals Vacuous or Meaningless results

Reveals Semantic flaws in vulnerability Categorization

Reveals systemic flaws in application spider technology

Web Application Scanner

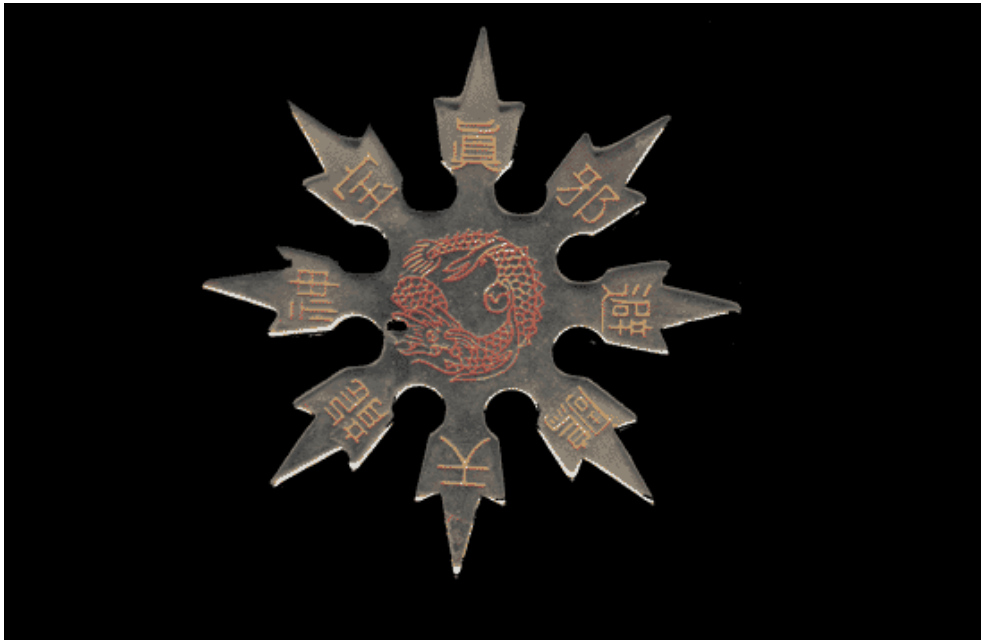
Web Application Security Scanners



Key Trends 2000-2007

- GUI's have gotten prettier but the underlying technology hasn't changed much since 2000.
- Many technologies are still using stateless stimulus-response mechanisms for most security tests (XSS is becoming the exception to this rule).
- False Positives related to the detection of SQL injection and Blind SQL Injection are rampant.
- Mechanics of file scanning is still largely based on Whisker-Nikto, and prone to false positives
- AJAX and Web Services support has increased the numbers of false positives, due to re-use of legacy security testing procedures.
- Signal-to-Noise Ratio is still very bad, with False Positives exceeding useful results usually by 2:1, and this is a conservative figure.
- Most application spiders do poorly against javascript and flash, and some technologies cannot automatically navigate Form-based logins.
- Semantic problems with security tests are widespread, i.e. mislabeled vulnerabilities, ambiguous vulnerabilities, meaningless results.
- Each year the problem gets worse, and acquisitions may further set back innovation.

The Problem of False Positives (A scanner darkly)



- Common False Positive Types are not Easily Studied…

Most EULAs prevent a comparison between technologies

B) **You shall not**: (i) use the Software or any portion thereof except as provided in this Agreement; (ii) modify, create derivative works of, translate, **reverse engineer**, decompile, disassemble (except to the extent applicable laws specifically prohibit such restriction) or **attempt to derive the Source Code of the Software** provided to You in machine executable object code form; (iii) market, distribute or otherwise transfer copies of the Software to others; (iv) rent, lease or loan the Software; (v) **distribute externally or to any third party any communication that compares the features, functions or performance characteristics of the Software with any other product of You or any third party**; or (vi) attempt to modify or tamper with the normal function of a license manager that regulates usage of the Software.

Reverse Benchmarking Methodology

- Active False Positive Solicitation and Reverse Fault Injection via a sample web application.

- A reverse benchmarking target can be used to model a production application, thereby decreasing the semantic gap between triggered false positives and false positives found within the production environment

Reverse Benchmarking Goals

- The goal of Reverse Benchmarking is not to malign vendors, but to aid the security community and help developers avoid the same mistakes with each new generation of technology

- Systematically performed, Reverse Benchmarking can help security practioners learn to quickly distinguish false positives from valid security issues, as they will learn the conditions under which the technology they are using fails.

- Based on the type of trigger that elicits the false positive, a taxonomy of false positive types can be developed. A set of common causes or contributing factors for each type can be outlined.

Common Causes of False Positives

- Partial Match Problems
  - Detection strings may be a subset of existing content and triggered by the presence of unrelated words or elements within the HTML or DOM

```
GET /search.pl~bak
```

```
July 2007
```

```
200 OK
```

Parameter Echoing

- Parameter values may be echoed back in places within a web application, and this can trigger false positives.

```
<TEXTAREA rows=3 ls=100>
•  <?php
•  // get the form data
•  $field1 = $_POST['comments'];
•  // Echo the value of the comments parameter
•  echo "Backacha Biatch: $field1";
•  ?>
•  </TEXTAREA>
```

Mistaken Identity

- Some security tests look for vulnerability conditions so general that the vulnerability reported must be disambiguated in order to be verified.
- Many types of PHP forum software, Calendars, Blogs reuse a common code base and so overlapping URI and application responses

Alibaba Search Overflow

Paul's Search SQL InjXn

YABB Search.pl XSS

`GET /search.pl`

- Semantic Ambiguity
  - Signature-based detection is often relies on signatures that are generic and thus are neither necessary nor sufficient for the vulnerability to be present.

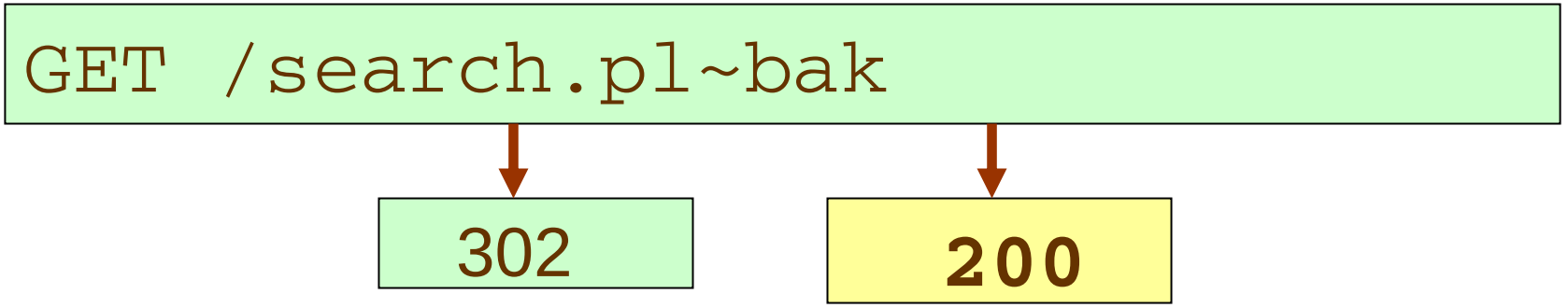
[Microsoft][ODBC SQL Server Driver]

Many false positives arise because the vulnerability is more complex than the vulnerability conditions checked for by the signatures.

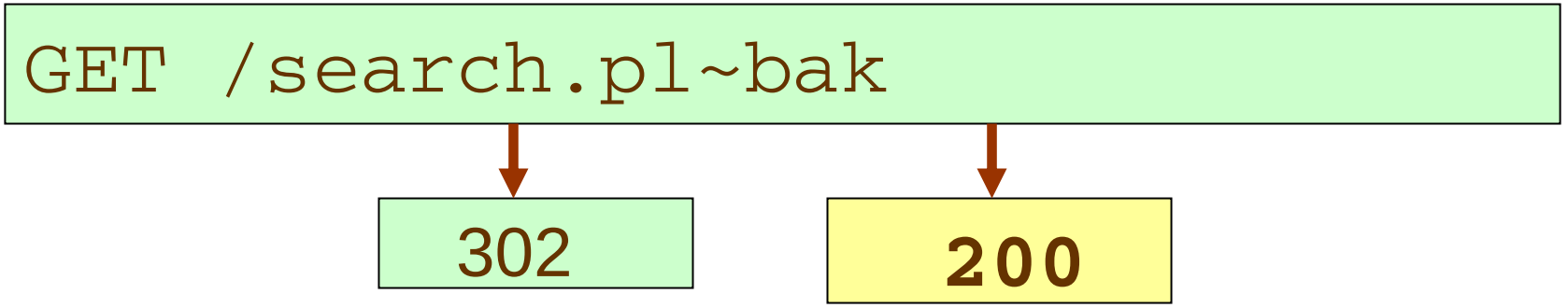
- Response Timing
  - Slow, unresponsive, or delayed server-side processing can trigger security checks that are timing dependent

Some SQL injection tests use a wait_for_delay expression and measure the timing.

- Custom 404 Pages
- Simple file scanning routines and other security tests will trigger erroneously in the presence of custom 404 pages.
- Some signatures are based on 302 Redirects



- Custom 404 Pages
- Simple file scanning routines and other security tests will trigger erroneously in the presence of custom 404 pages.
- Some signatures are based on 302 Redirects



- Creating a Reverse Benchmark target
  - Nature of the target will depend on your goals as a researcher

Reverse Engineering

1. Emphasis on exposing as much of the signature base and rule set as possible without inspecting datafiles or code. Clear generic cases that will likely impact the largest portion of the rule base

2. Focus on generic trigger signatures, including available open source scanners. (i.e. use of Nikto detections strings in response data.

- Creating a Reverse Benchmark target
  - Nature of the target will depend on your goals as a researcher

Bakeoffs/Comparisons

1. Emphasis on exposing false positives or signature flaws of all varieties, including the uncommon or essoteric. Use of non-standard or overly difficult application configuration to stress test the scanner.

2. Focus on unusual or non-standard trigger signatures. i.e. Javascript or Flash road test

- Creating a Reverse Benchmark target
  - Nature of the target will depend on your goals as a researcher

Reverse Engineering

1. Emphasis on exposing as much of the signature base and rule set as possible without inspecting datafiles or code.

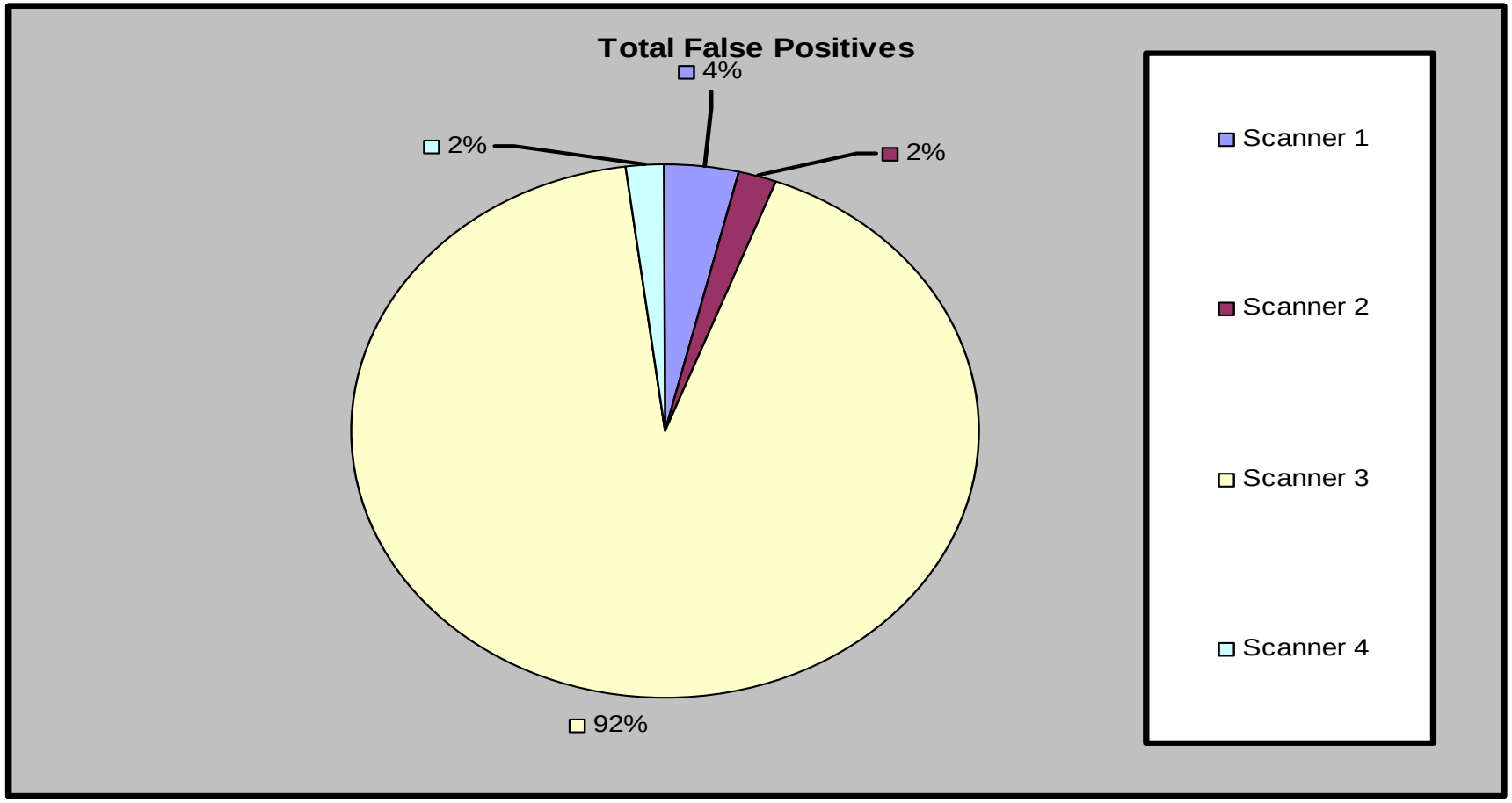
2. Focus on generic trigger signatures

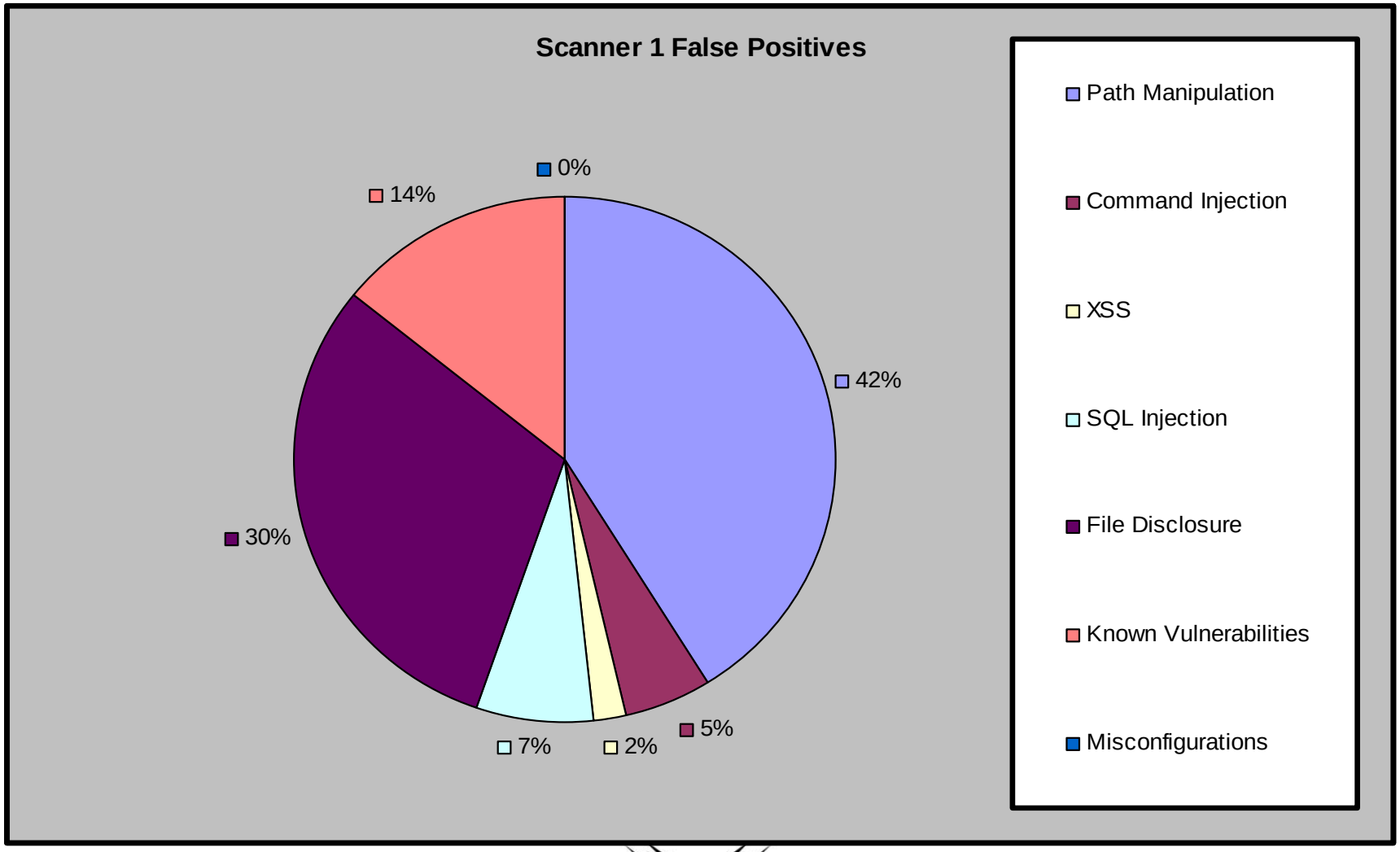
- Open Reverse Benchmarking Project
  - Nature of the target will depend on your goals as a researcher

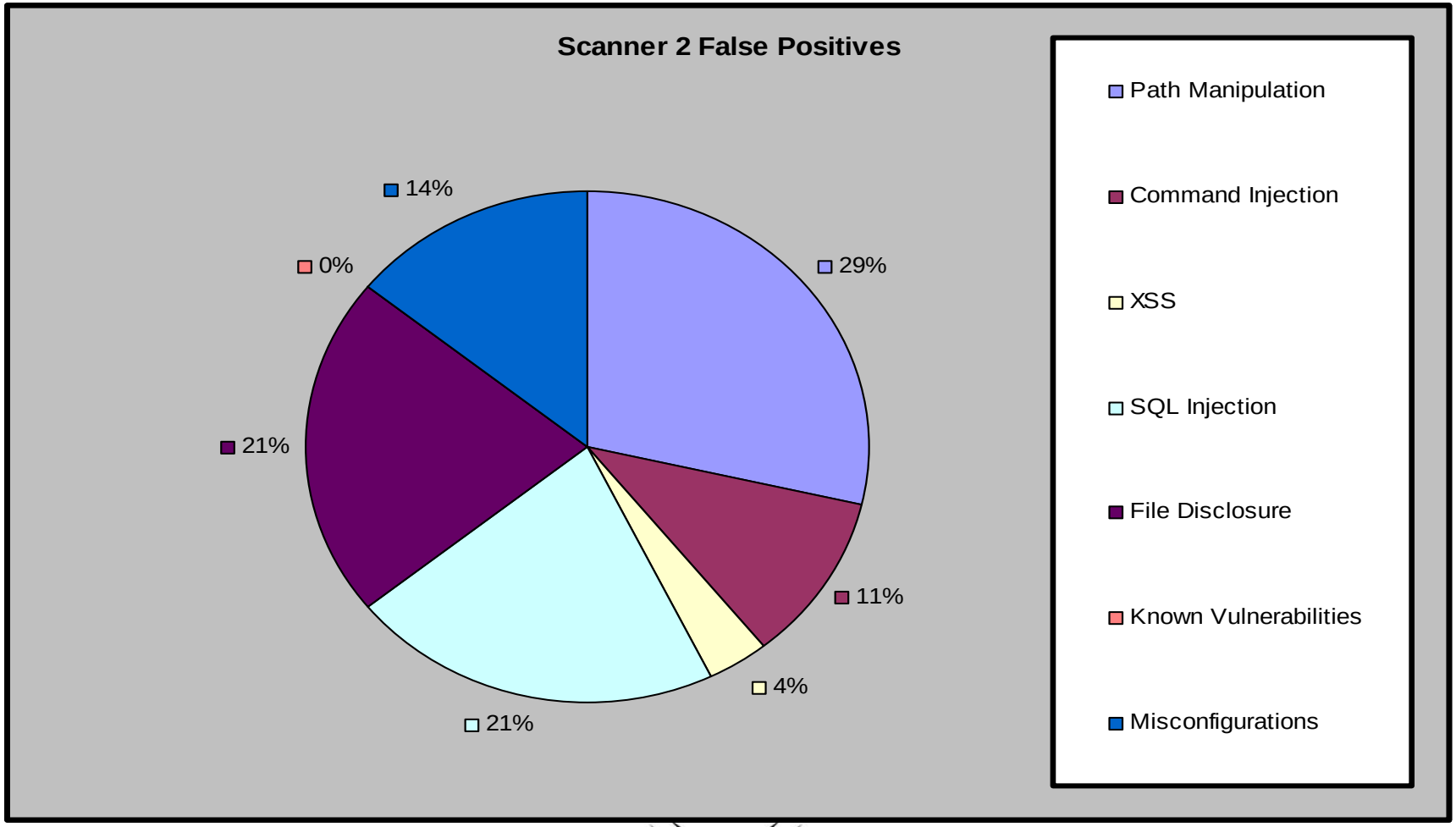
1. Emphasis on exposing as much of the signature base and rule set as possible without inspecting datafiles or code.

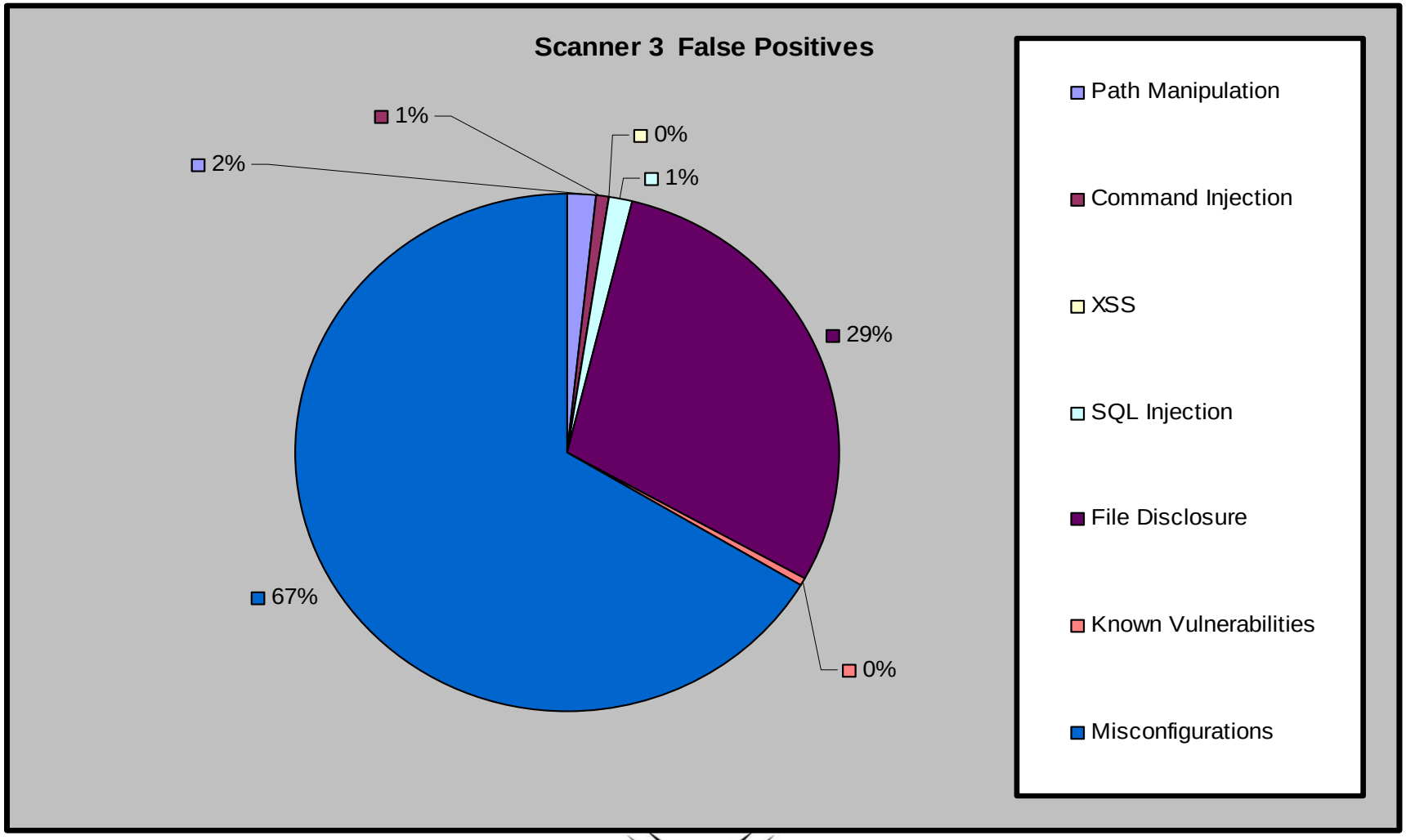
2. Focus on generic trigger signatures

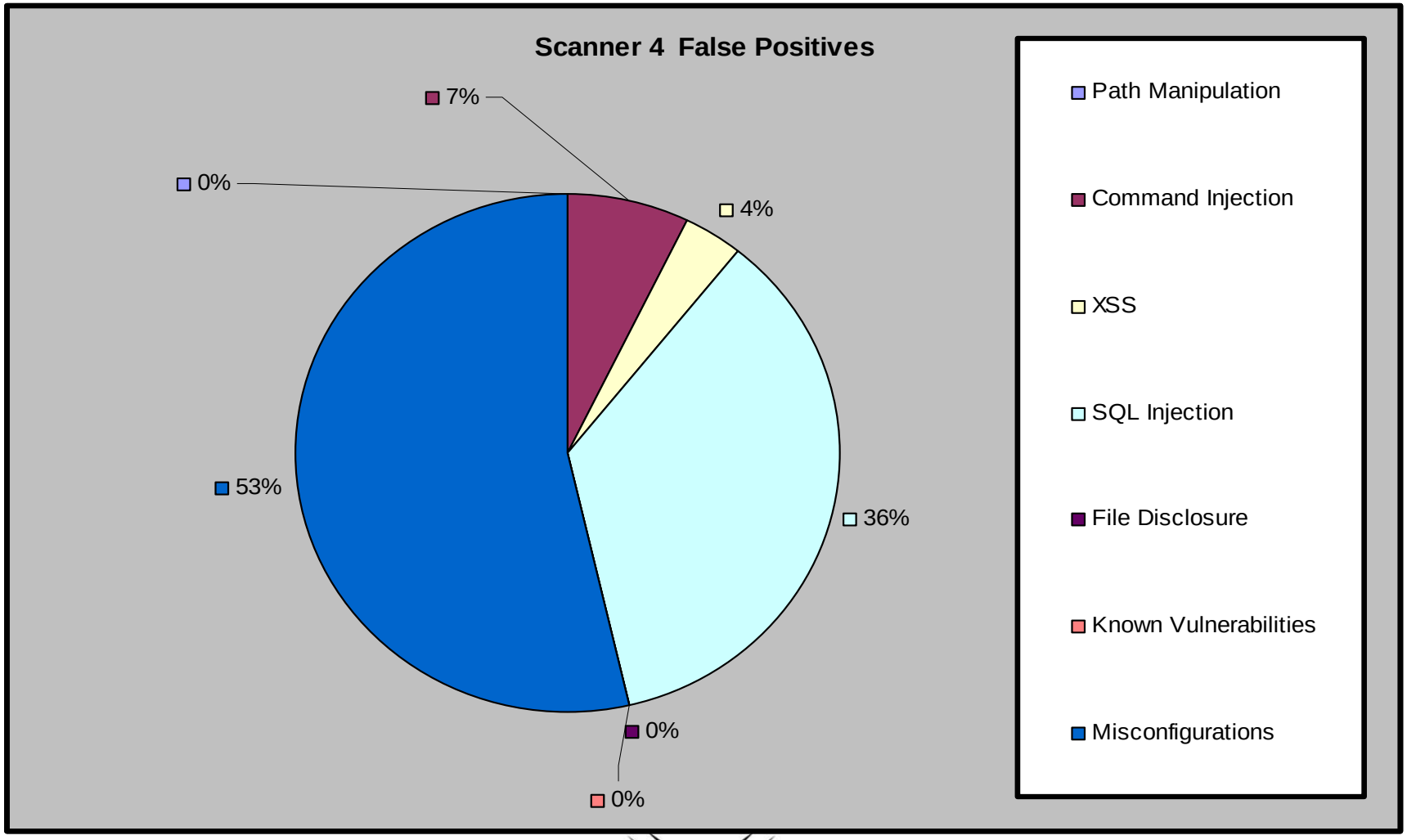
- Backatcha Roadtest Results
- Took 4 popular blackbox web application security scanners
- Ran their default policies against the target reverse benchmarking application
- Put the results into high level buckets
- Generated a few graphs with the results


Total False Positives
4%
2%
2%
92%
Scanner 1
Scanner 2
Scanner 3
Scanner 4


Scanner 1 False Positives
0%
14%
42%
30%
7%
2%
5%
Path Manipulation
Command Injection
XSS
SQL Injection
File Disclosure
Known Vulnerabilities
Misconfigurations


Scanner 2 False Positives
14%
29%
0%
21%
11%
4%
21%
Path Manipulation
Command Injection
XSS
SQL Injection
File Disclosure
Known Vulnerabilities
Misconfigurations


Scanner 3 False Positives
1%
0%
2%
1%
29%
67%
0%
Path Manipulation
Command Injection
XSS
SQL Injection
File Disclosure
Known Vulnerabilities
Misconfigurations


Scanner 4 False Positives
7%
0%
4%
53%
36%
0%
0%
Path Manipulation
Command Injection
XSS
SQL Injection
File Disclosure
Known Vulnerabilities
Misconfigurations

- Conclusions
  - All Scanners had simple problems
  - One scanner did really badly
  - Further Research is needed
  - Community support is needed
  - Examples of false positives

- Further Research
  - Improve reverse benchmarking target
  - Add more tests
  - Improve testing methodology
  - Test with more scanners
- Partner with OWASP
  - Help develop Reverse Benchmarking Module for SiteGenerator