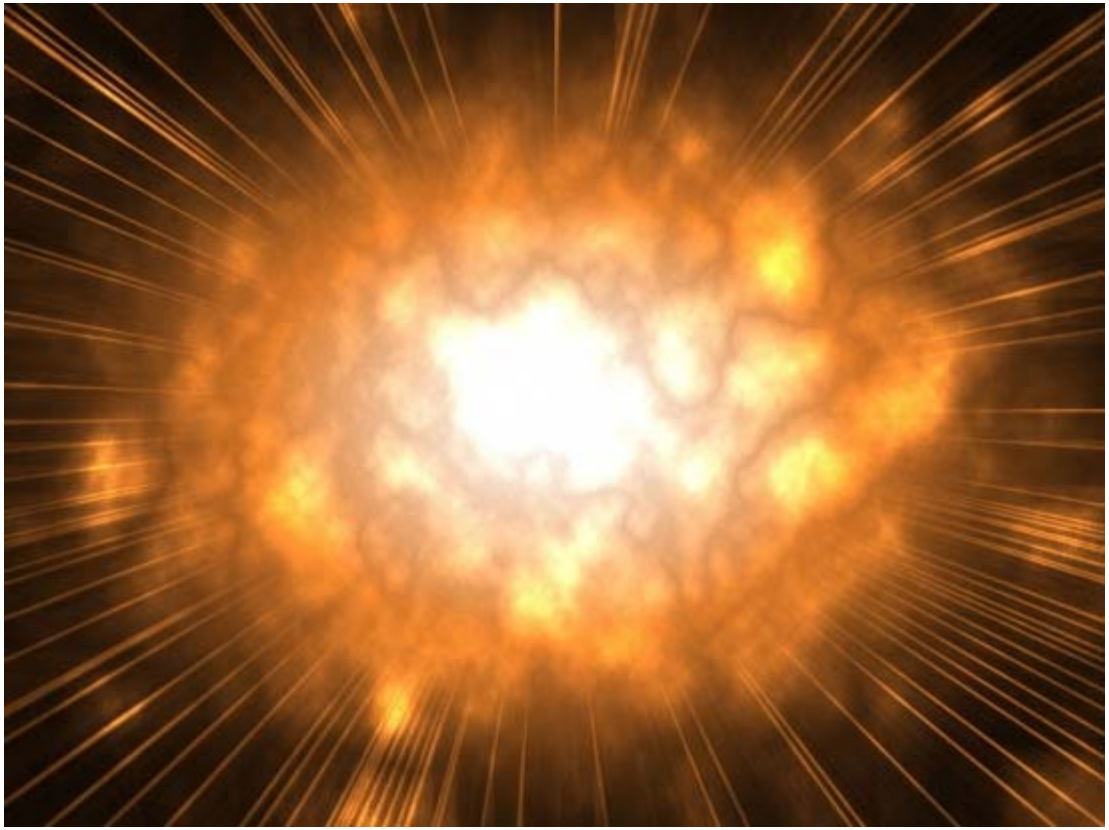
WebBlaze: New Security Technologies for the Web

Dawn Song

Computer Science Dept.

UC Berkeley

Web: Increasing Complexity



Ensuring Security on the Web Is Complex & Tricky

- **Does the browser correctly enforce desired security policy?**
- **Is third-party content such as malicious ads securely sandboxed?**
- **Do browsers & servers have consistent interpretations/views to enforce security properties?**
- **Do web applications have security vulnerabilities?**
- **Do different web protocols interact securely?**

WebBlaze: New Security Technologies for the Web

- **Does the browser correctly enforce desired security policy?**
  - ***Cross-origin capability leaks: attacks & defense [USENIX 09]***
- **Is third-party content such as malicious ads securely sandboxed?**
  - ***Preventing Capability Leaks in Secure JavaScript Subsets [NDSS10]***
- **Do browsers & servers have consistent interpretations/views to enforce security properties?**
  - ***Document Structure Integrity: A Robust Basis for Cross-site Scripting Defense [NDSS09]***
  - ***Content sniffing XSS: attacks & defense [IEEE S&P 09]***
- **Do applications have security vulnerabilities?**
  - ***Symbolic Execution Framework for JavaScript [IEEE S&P10]***
- **Do different web protocols interact securely?**
  - ***Model checking web protocols (Joint with Stanford)***

Outline

- **WebBlaze Overview**
- **Content sniffing XSS attacks & defense**
- **New class of vulnerabilities: Client-side Validation (CSV) Vulnerability**
- **Kudzu: JavaScript Symbolic Execution Framework for in-depth crawling & vulnerability scanning of rich web applications**
- **Conclusions**

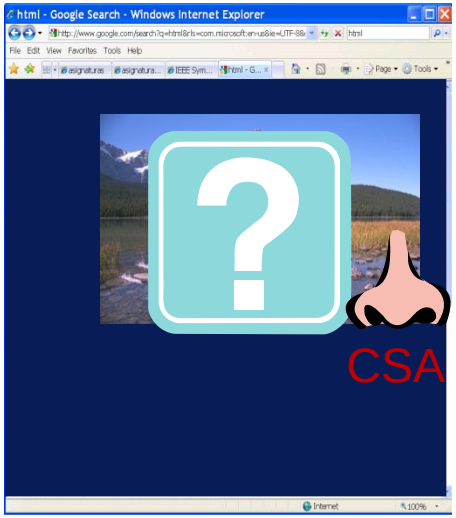
Is this a paper or a web page?

%!PS-Adobe-2.0

%%Creator: <script> ... </script>

What happens if IE decides it is HTML?

Content Sniffing Algorithm (CSA)

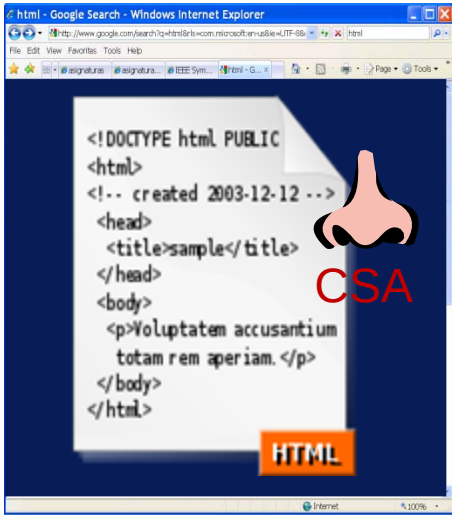


GET /patagonia.gif HTTP/1.1

HTTP/1.1 200 OK
Content-Type: image/gif

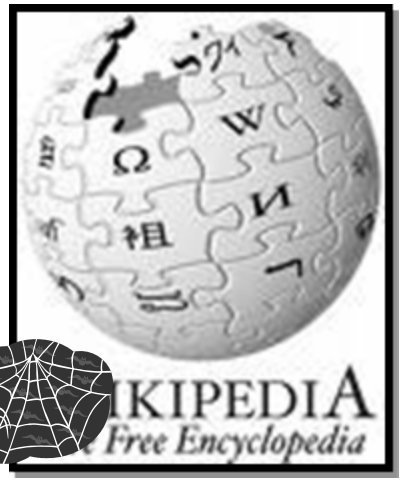
GIF89a38jf9w8nf99uf9…

Content Sniffing XSS Attack

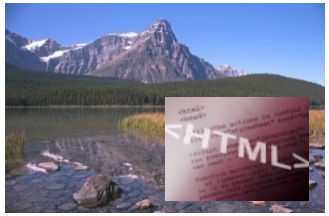


GET /patagonia.gif HTTP/1.1

HTTP/1.1 200 OK
Content-Type: image/gif



patagonia.gif



Automatically Identifying Content Sniffing XSS Attacks

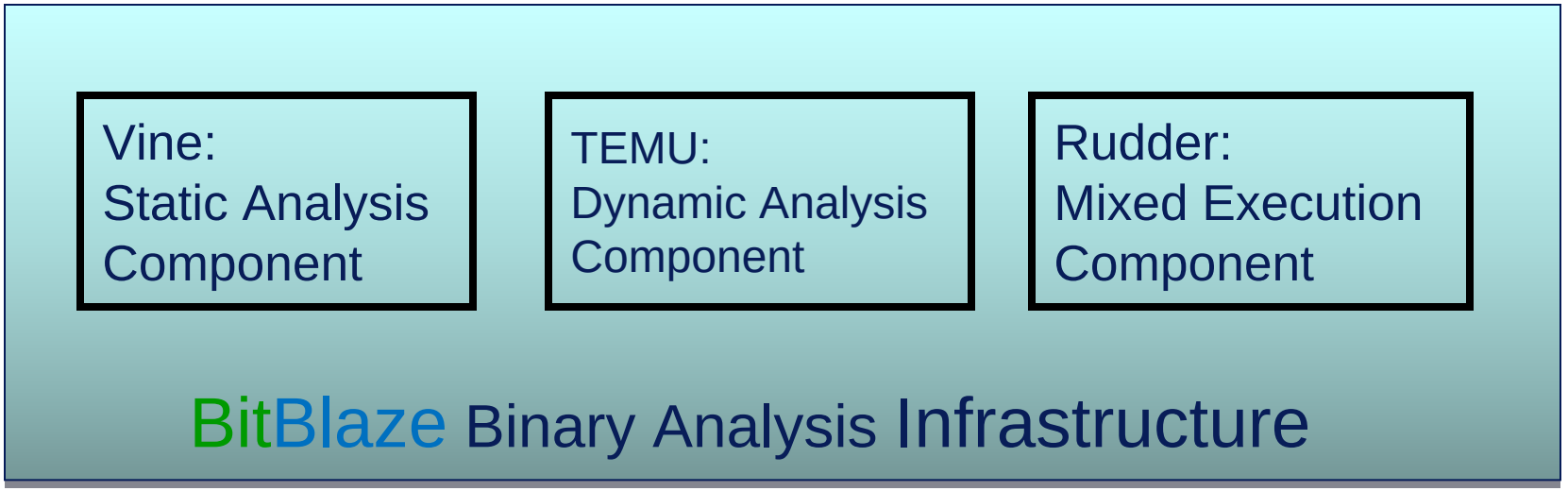
- **Website content filter modeled as Boolean predicate on the input (accepted/rejected)**
- **Browser CSA modeled as multi-class classifier**
  - **One per output MIME type (e.g., text/html or not)**
- **Query a solver for inputs that are:**
  1. **Accepted by the website's content filter**
  2. **Interpreted as HTML by the browser's CSA**

Challenge: Extracting CSA from Close-sourced Browsers

- **IE7, Safari 3.1**
- **Need automatic techniques to extract model from program binaries**

BitBlaze Binary Analysis Infrastructure

- **The first infrastructure:**
  - **Novel fusion of static, dynamic, formal analysis methods**
    - » **Loop extended symbolic execution**
    - » **Grammar-aware symbolic execution**
  - **Identify & cater common needs for security applications**
  - **Whole system analysis (including OS kernel)**
  - **Analyzing packed/encrypted/obfuscated code**



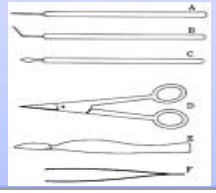
BitBlaze: Security Solutions via Program Binary Analysis

- Unified platform to accurately analyze security properties of binaries
  - ✓ Security evaluation & audit of third-party code
  - ✓ Defense against morphing threats
  - ✓ Faster & deeper analysis of malware

Detecting Vulnerabilities

Generating Filters

Dissecting Malware

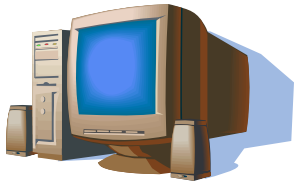


BitBlaze Binary Analysis Infrastructure

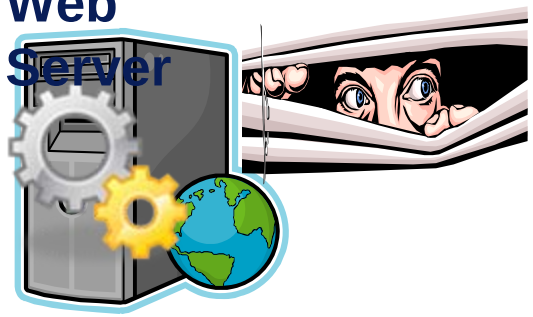
Extracting CSA from Close-sourced Browsers

- **IE7, Safari 3.1**
- **String-enhanced symbolic execution on binary programs**
  - **Build on top of BitBlaze**
  - **Model extractions via program execution space exploration**
  - **Model string operations and constraints explicitly**
  - **Solve string constraints**
- **Identify real-world vulnerabilities**

Symbolic Execution: Path Predicate



GET /
HTTP/1.1



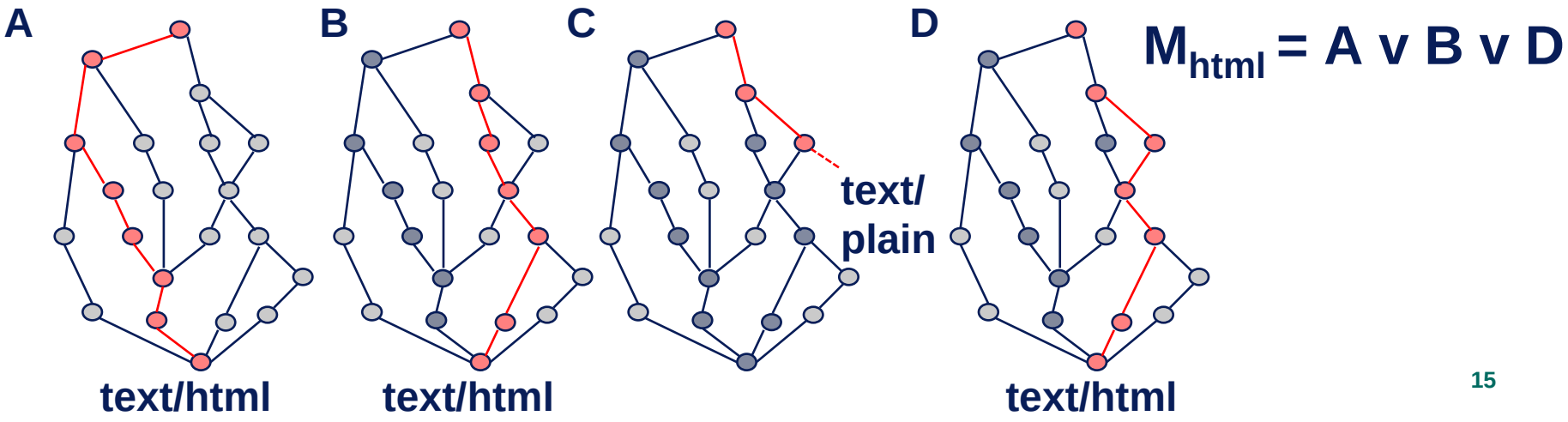
Executed instructions	Intermediate Representation (IR)
`mov(%esi), %al`	`AL = INPUT[0]`
`mov $0x47, %bl`	`BL = 'G'`
`cmp %al, %bl`	`ZF = (AL == BL)`
`jnz FAIL`	`IF(ZF==0)JMP(FAIL)`
`mov 1(%esi), %al`	`AL = INPUT[1]`
`mov $0x45, %bl`	`BL = 'E'`
`cmp %al, %bl`	`ZF = (AL == BL)`
`jnz FAIL`	`IF(ZF==0)JMP(FAIL)`
…	…

Path predicate

(INPUT[0] == 'G')
^
(INPUT[1] == 'E')
^
…

Model Extraction on Binary Programs

- **Symbolic execution for execution space exploration**
  - **Obtain path predicate using symbolic input**
  - **Reverse condition in path predicate**
  - **Generate input that traverses new path**
  - **Iterate**
- **String-enhanced symbolic execution**
- **Model: disjunction of path predicates**



IE7/HotCRP Postscript Attack

- **HotCRP Postscript signature**

  strncasecmp(DATA, "%!PS-", 5) == 0

- **IE 7 signatures**

  application/postscript: strncmp(DATA, "%!", 2) == 0

  text/html: strcasestr(DATA,"<SCRIPT") != 0

- **Attack**

  %!PS-Adobe-2.0

  %%Creator: <script> ... </script>

IE7/Wikipedia GIF Attack

- **Wikipedia GIF signature**

  strncasecmp(DATA,“GIF8”,4) == 0)

- **IE 7 signatures**

  image/gif: (strncasecmp(DATA,“GIF87”,5) == 0) ||
  (strncasecmp(DATA,“GIF89”,5) == 0)

  text/html: strcasestr(DATA,"<SCRIPT") != 0

- **Fast path: check GIF signature first**
- **Attack**

  GIF88<script> … </script>

Results: Models & Attacks

Model	Seeds	Path count	% HTML paths	Avg. # Paths per seed	Avg. Path gen. time	# Inputs generated	Avg. Path depth
Safari 3.1	7	1558	12.4%	222.6	16.8 sec	7166	12.1
IE 7	7	948	8.6%	135.4	26.6 sec	64721	212.1

- **Filter = Unix File tool / PHP**
- **Find inputs**
  - **Accepted by filter**
  - **Interpreted as text/html**
- **Attacks on 7 MIME types**

Model	IE 7	Safari 3.1
application/postscript	√	√
audio/x-aiff	√	√
image/gif	√	√
image/tiff	√	√
image/png	-	√
text/xml	√	-
video/mpeg	√	√

Defenses

1. **Don't sniff**
   - **Breaks ~1% of HTTP responses**
   - **Works in IE + fails in Firefox = Firefox's problem**

2. **Secure sniffing**
   1. **Avoid privilege escalation**
      - » **Prevent Content-Types from obtaining hig privilege**
   2. **Use prefix-disjoint signatures**
      - » **No common prefix with text/html**

Adoption

- **Full adoption by Google Chrome**
  - **Shipped to millions of users in production**
- **Partial adoption by Internet Explorer 8**
  - **Partially avoid privilege escalation**
  - **Doesn’t upgrade image/* to text/html**
- **Standardized**
  - **HTML 5 working group adopts our principles**

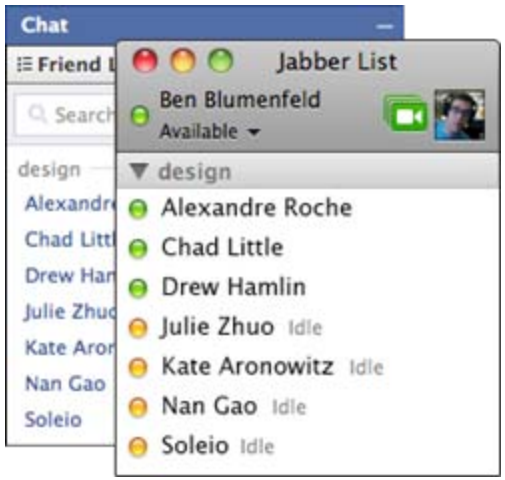
Outline

- **WebBlaze Overview**
- **Content sniffing XSS attacks & defense**
- **New class of vulnerabilities: Client-side Validation (CSV) Vulnerability**
- **Kudzu: JavaScript Symbolic Execution Framework for in-depth crawling & vulnerability scanning of rich web applications**
- **Conclusions**

Rich Web Applications

- **Large, complex Ajax applications**
- **Rich cross-domain interaction**

Gmail by Google Google maps Google docs



Client-side Validation(CSV) Vulnerabilities

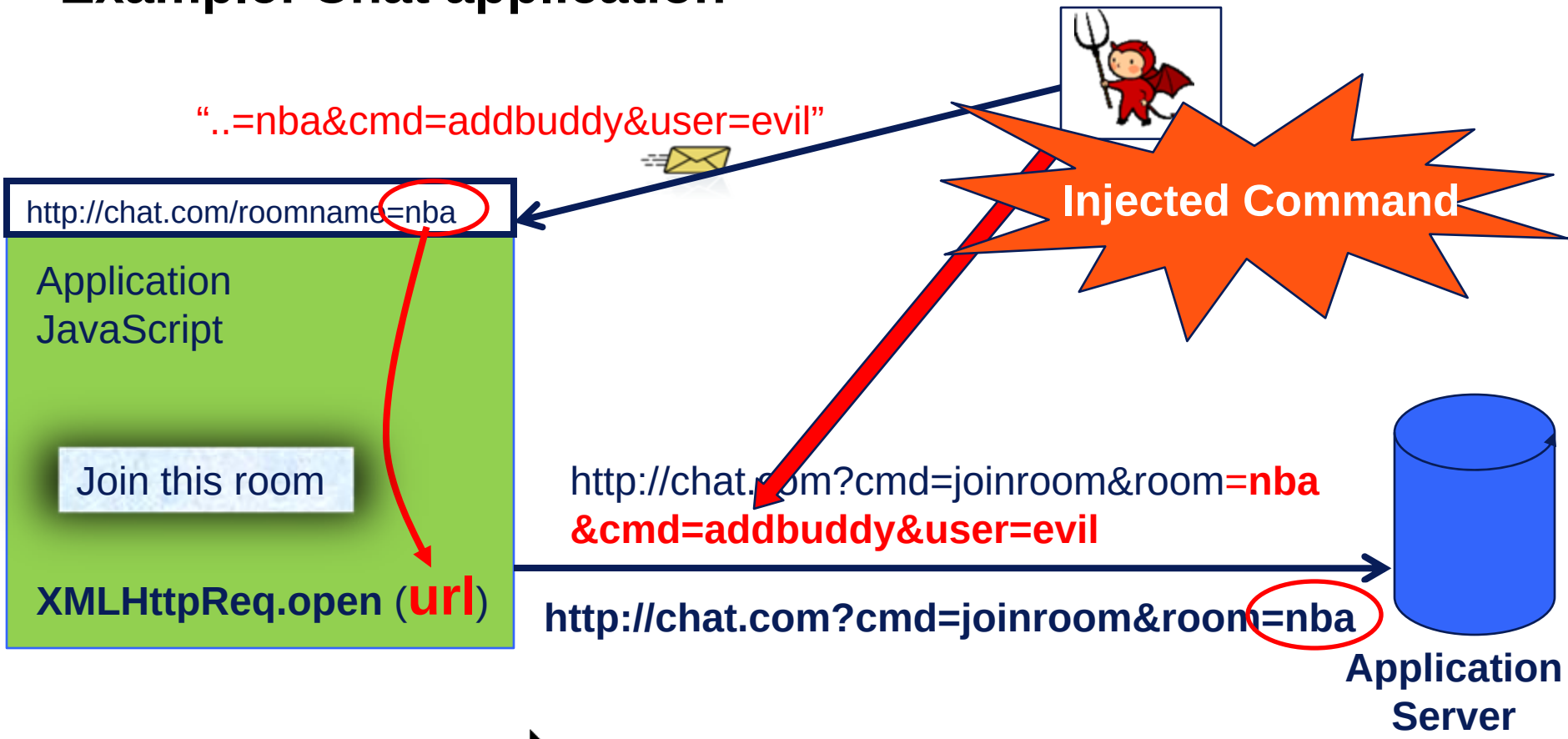
- **Most previous security analysis focuses on server side**
- **A new class of input validation vulnerabilities**
  - **Analogous to server-side bugs**
  - **Unsafe data usage in the client-side JS code**
  - **Different forms of data flow**
    - **Purely client-side, data never sent to server**
    - **Returned from server, then used in client-side code**

Vulnerability Example (I): Code Injection

- **Code/data mixing**
- **Dynamic code evaluation**
  - **eval**
  - **DOM methods**
- **Eval also deserializes objects**
  - **JSON**

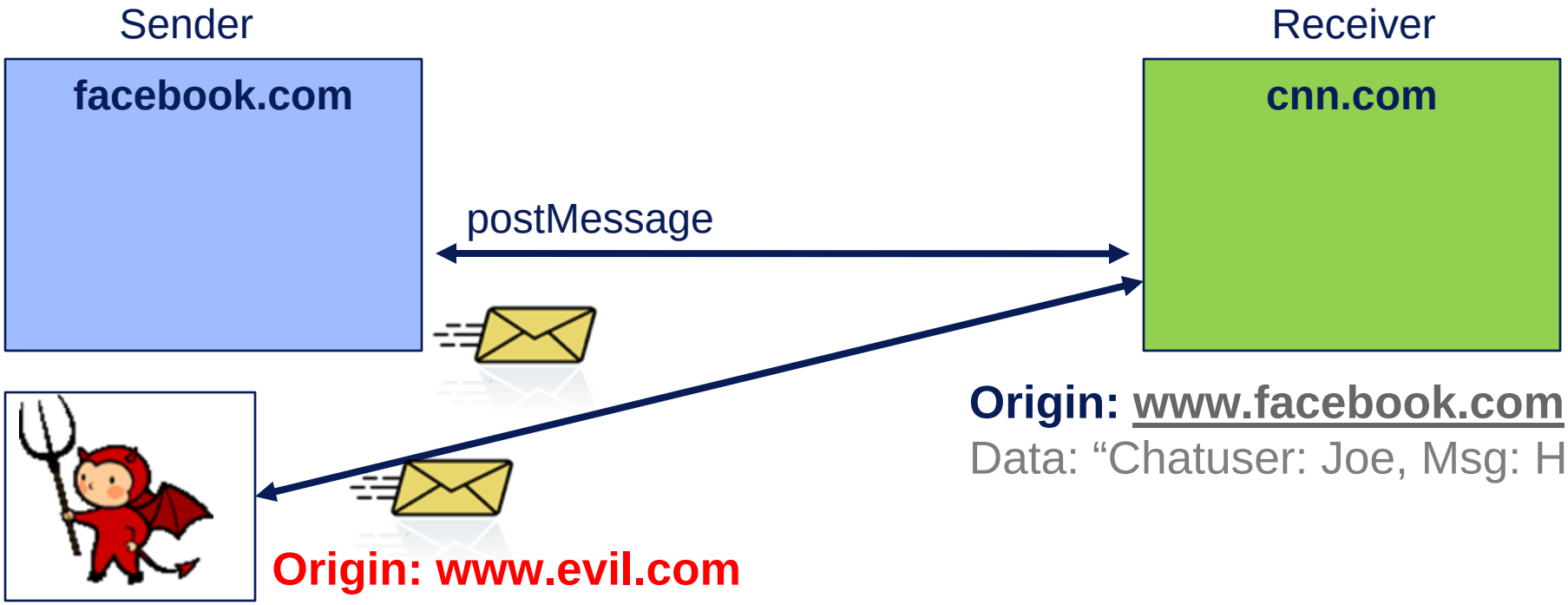
Vulnerability Example (II): Application Command Injection

- **Application-specific commands**
- **Example: Chat application**



Vulnerability Example (III): Origin Misattribution

- **Cross-domain Communication**
  - **Example: HTML 5 postMessage**



Vulnerability Example (IV): Cookie Sink Vulnerabilities

- **Cookies**
  - **Store session ids, user's history and preferences**
  - **Have their own control format, using attributes**
- **Can be read/written in JavaScript**
- **Attacks**
  - **Session fixation**
  - **History and preference data manipulation**
  - **Cookie attribute manipulation, changes**

Outline

- **WebBlaze Overview**
- **Content sniffing XSS attacks & defense**
- **New class of vulnerabilities: Client-side Validation (CSV) Vulnerability**
- **Kudzu: JavaScript Symbolic Execution Framework for in-depth crawling & vulnerability scanning of rich web applications**
- **Conclusions**

Motivation

- **AJAX applications**
  - **Increasingly complex, large execution space**
  - **Lots of bugs, few techniques for systematic discovery**
- **Current web vulnerability scanners cannot handle rich web apps**
- **Need tools for automatic in-depth exploration of rich web apps**
- **Lots of potential applications**
  - **Testing, Vulnerability Diagnosis, Input Validation Sufficiency Checking**

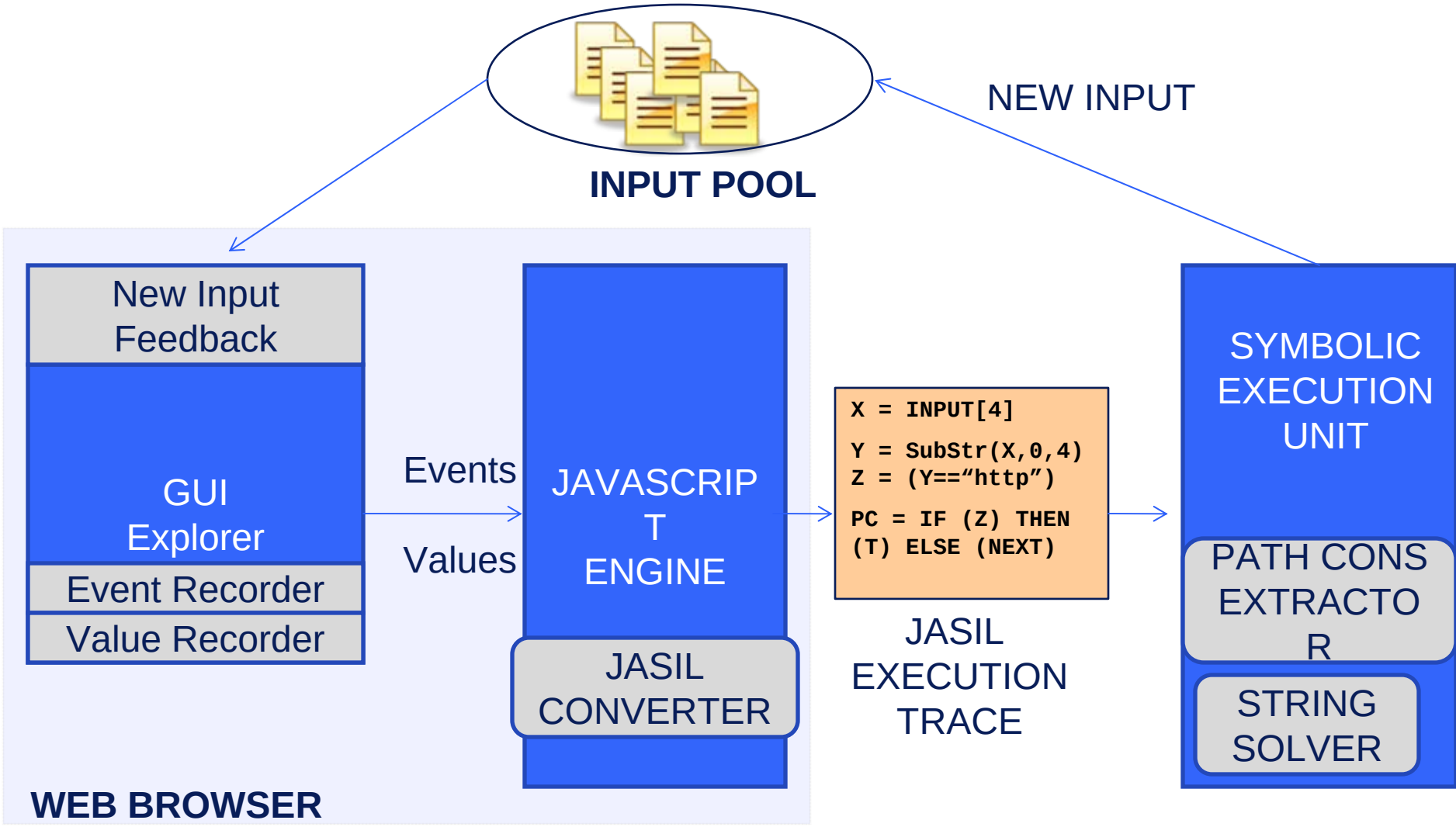
The Approach

- **JavaScript Execution Space Exploration**
- **Challenges**
  - **Large input space (*User, HTTP, Cross-window input*)**
  - **String-heavy**
    - » **Custom Parsing and validation checks, inter-mixed**
    - » **Contrast to PHP code, say, which has pre-parsed input**
  - **GUI exploration**
- **Application: Finding DOM-based XSS**
  - **DOM XSS: Untrusted data evaluated as code(eval, doc.write,..)**
  - **Challenge #1: Explore execution space**
  - **Challenge #2: Determine if data sufficiently sanitized/validated**

Kudzu: Overview

- **Program input space (web apps) has 2 parts**
  - **Event Space**
  - **Value Space**
- **GUI exploration for *event space***
- **Dynamic symbolic execution of JavaScript for *value space***
  - **Mark inputs symbolic, symbolically execute JS**
  - **Extract path constraints, as a formula F**
  - **Revert certain branch constraints in F**
  - **Solve Constraints**
  - **Feed the new input back**

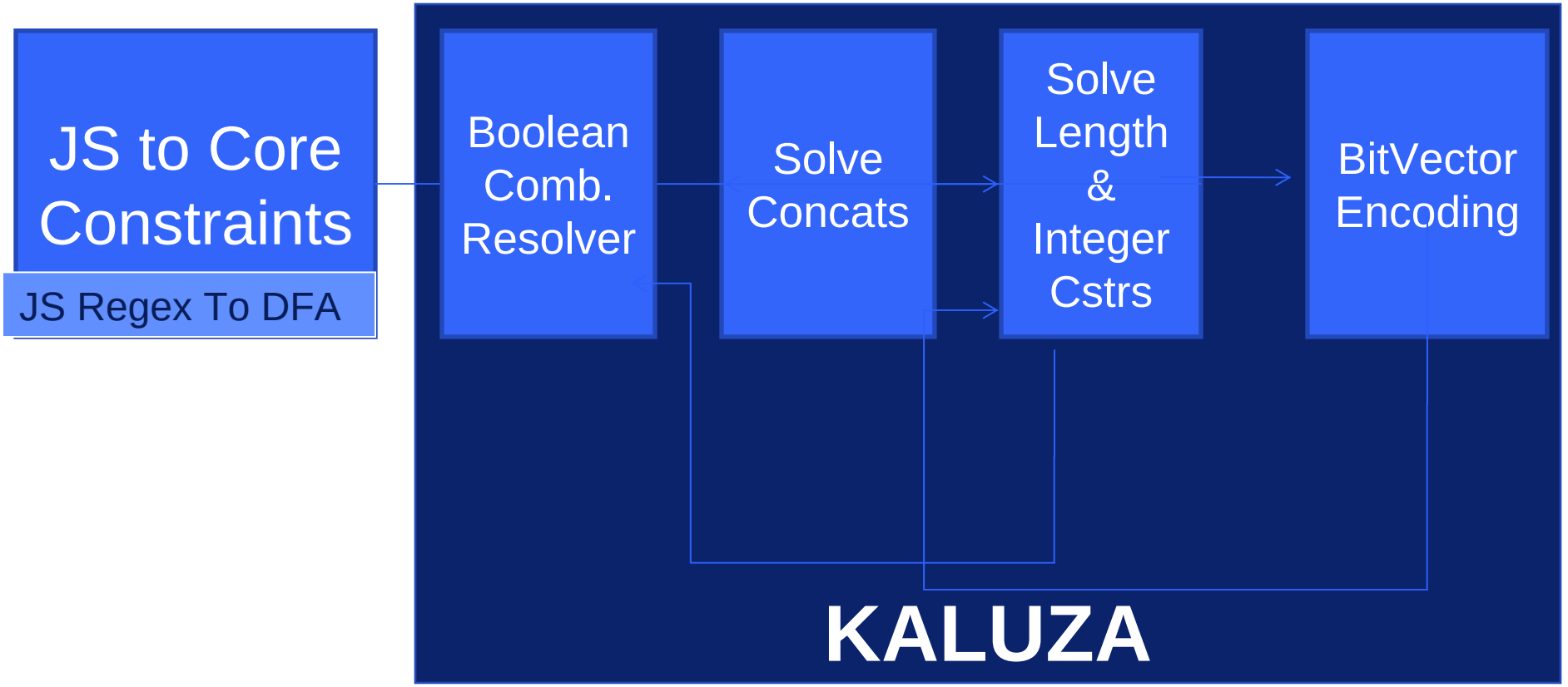
Kudzu: Path Exploration System



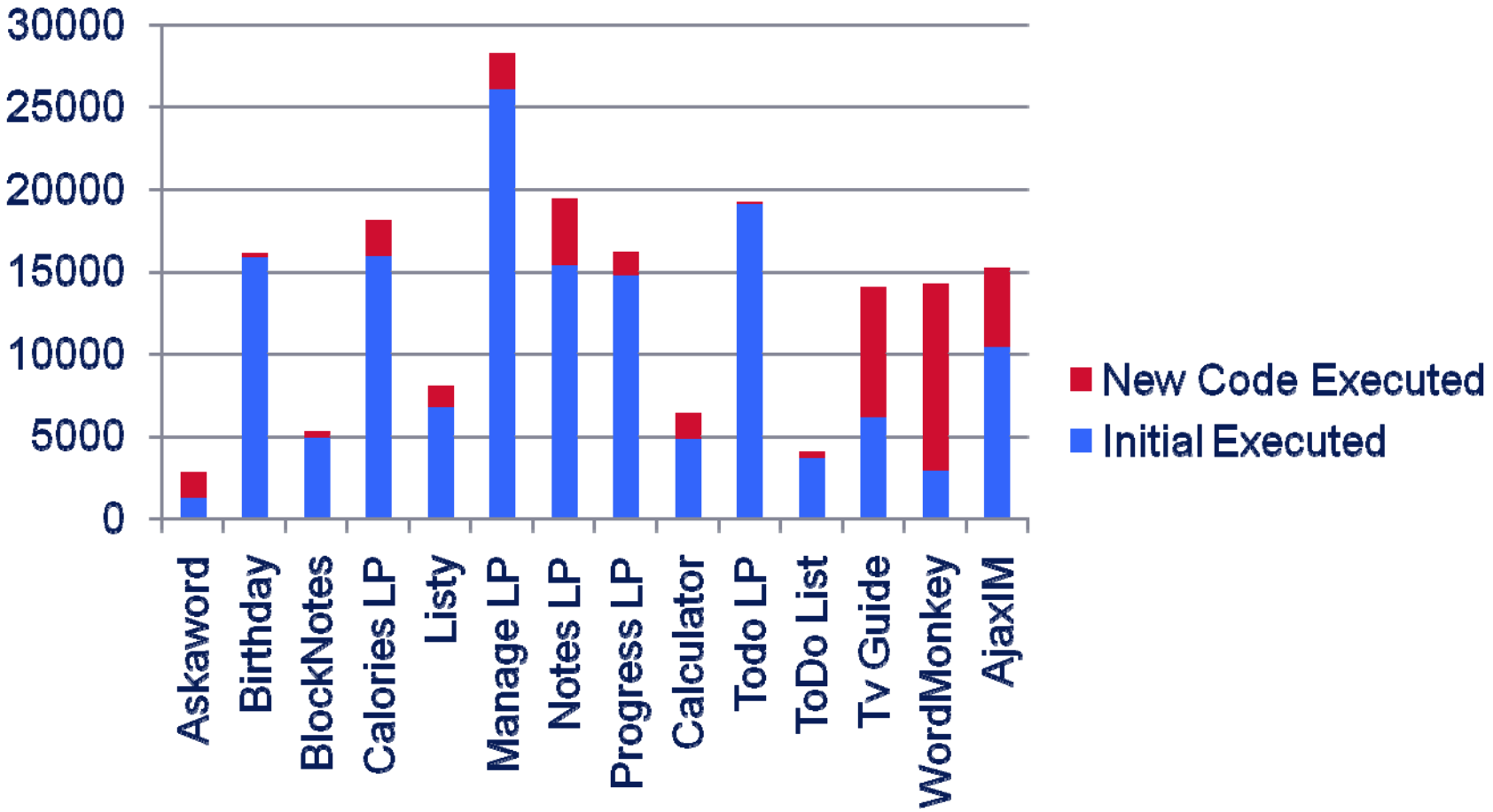
Kaluza: New String Constraint Solver

charAt	charCodeAt	concat	indexOf	lastIndexOf	match	replace	split
substr	toString	test	length	Enc/decodeURI	escape	parseInt	search

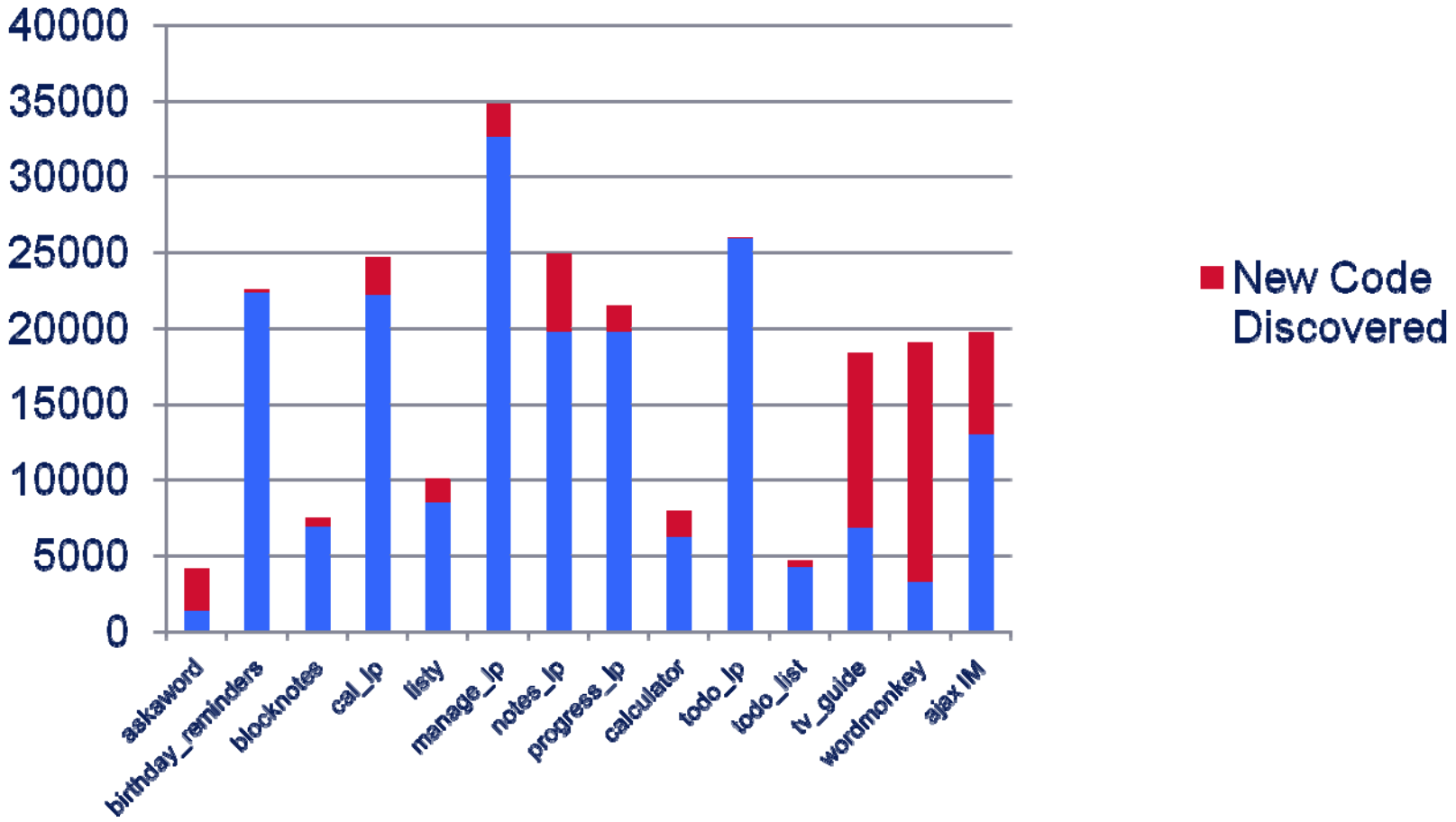
JAVASCRIPT STRING FUNCTIONS



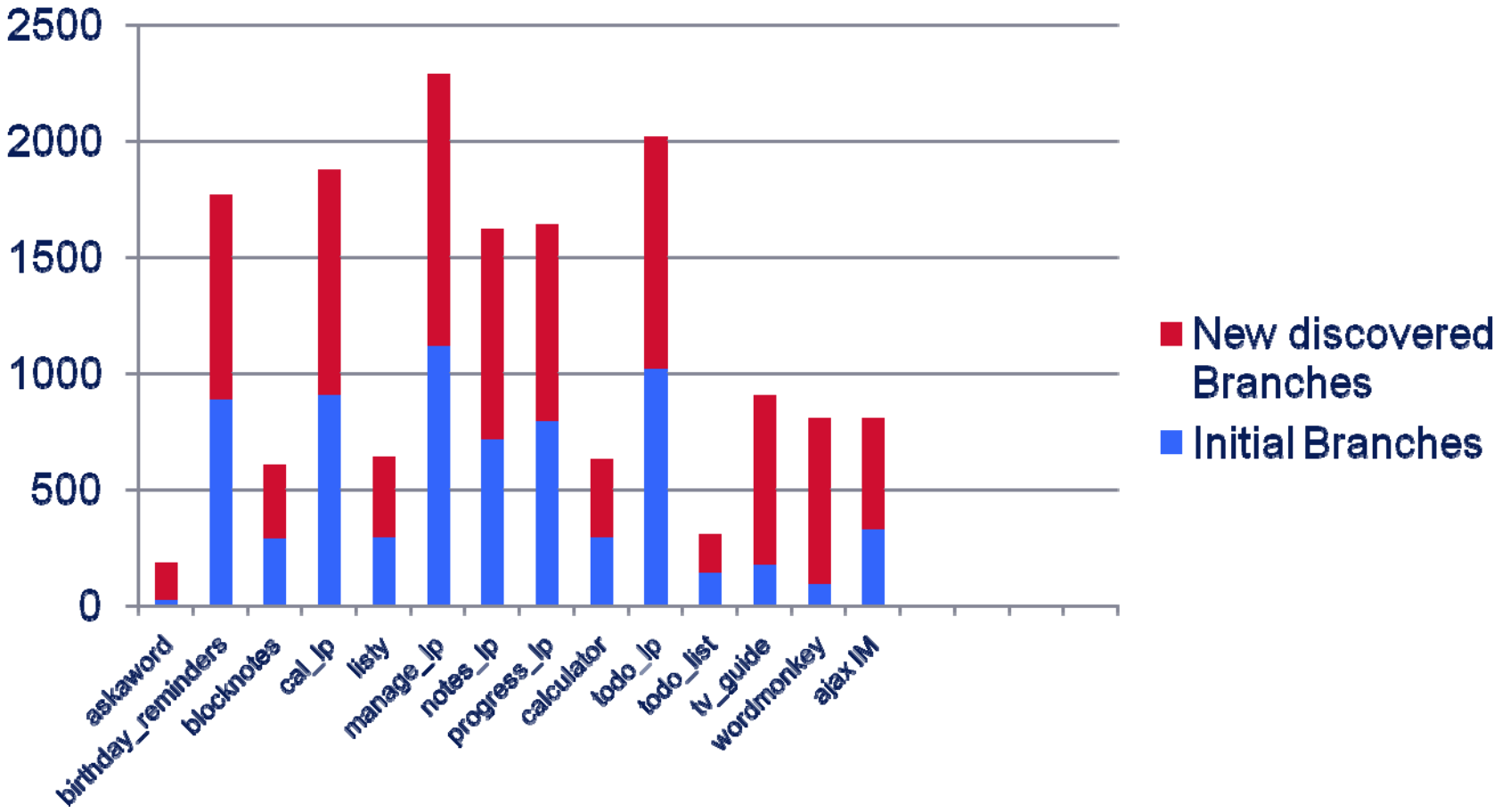
Symbolic Execution + GUI Exploration: New Code Executed



Symbolic Execution + GUI Exploration: New Code Compiled/Discovered



Symbolic Execution + GUI Exploration
New Discovered Branches



11 Vulnerabilities found out of 18 apps

Academia	1
AJAXim	1
Facebook	0
Plaxo	1
ParseURI	1
AskAWord	1
BlockNotes	1
Birthday Reminder	0
Calorie Watcher	0
Expenses Manager	0
Listy	1
NotesLP	0
SimpleCalculator	1
Progress Bar	0
ToDo	1
TVGuide	1
WordMonkey	1
ZipCodeGas	0

Conclusion

- **WebBlaze: new technologies for web security**
  - **Does the browser correctly enforce desired security policy?**
  - **Is third-party content such as malicious ads securely sandboxed?**
  - **Do browsers & servers have consistent interpretations/views to enforce security properties?**
  - **Do applications have security vulnerabilities?**
  - **Do different web protocols interact securely?**





bitblaze.cs.berkeley.edu

webblaze.cs.berkeley.edu

dawnsong@cs.berkeley.edu