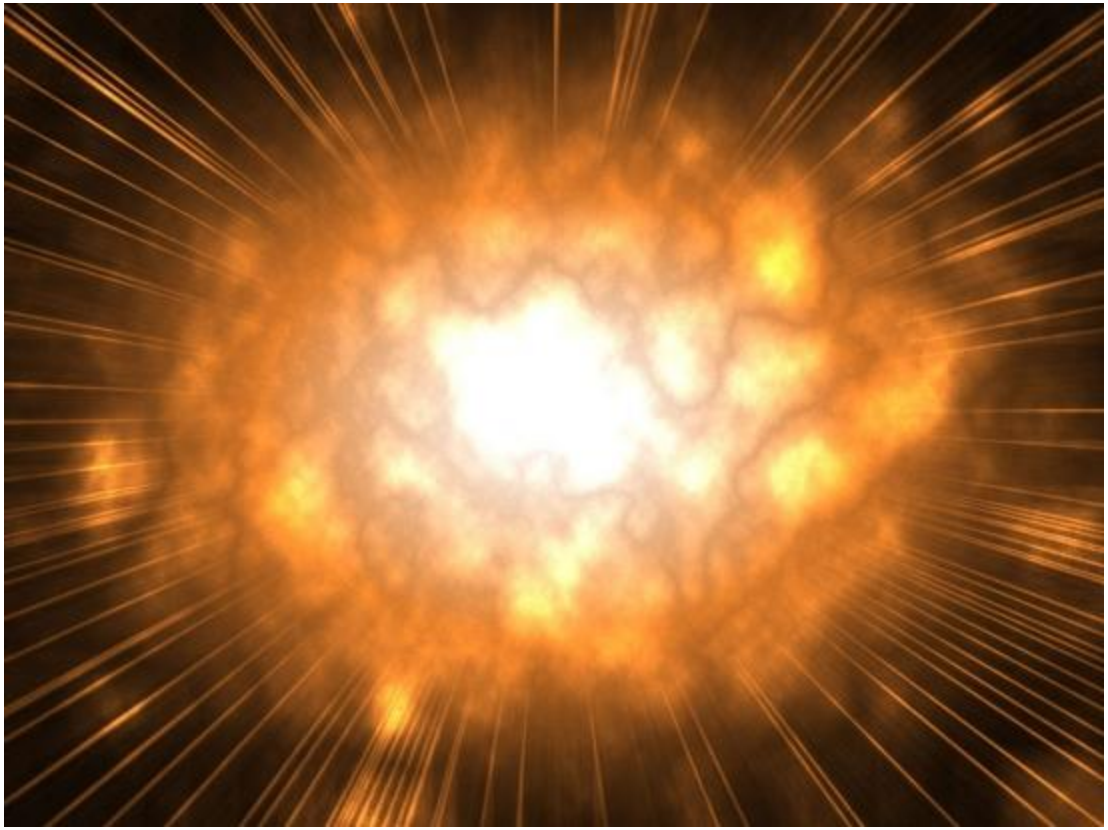
WebBlaze: New Security Technologies for the Web

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 [CSF 10]***

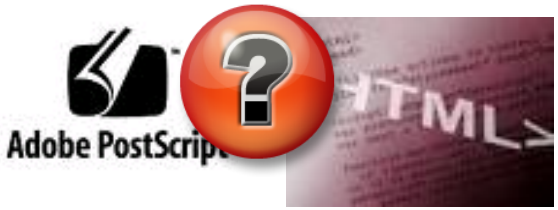
Outline

- **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**
- **Type-based Approach for Context-sensitive Automatic Sanitization in Web Templating Languages**
- **Overview on BitBlaze**
- **Overview on DroidBlaze**
- **Conclusions**

Is this a paper or a web page?

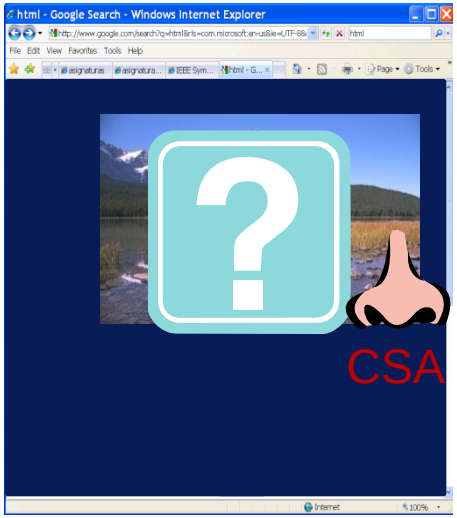
%!PS-Adobe-2.0

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



What happens if IE decides it is HTML?

Content Sniffing Algorithm (CSA)

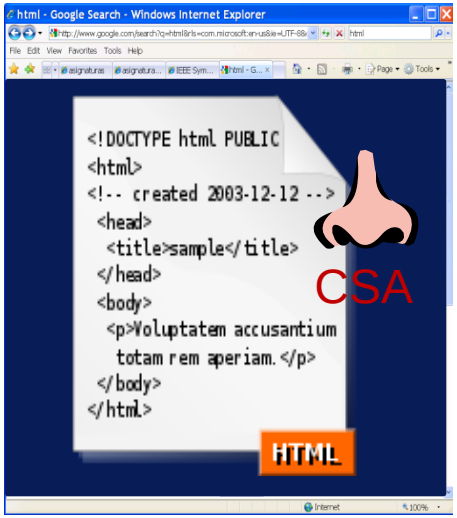


GET /patagonia.gif HTTP/1.1

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

GIF89a38jf9w8nf99uf9…

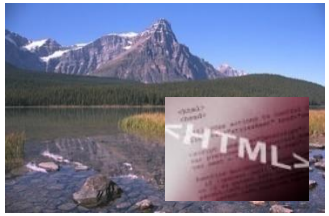
Content Sniffing XSS Attack



GET /patagonia.gif HTTP/1.1

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

patagonia.gif



Automatically Identifying Content Sniffing XSS Attacks

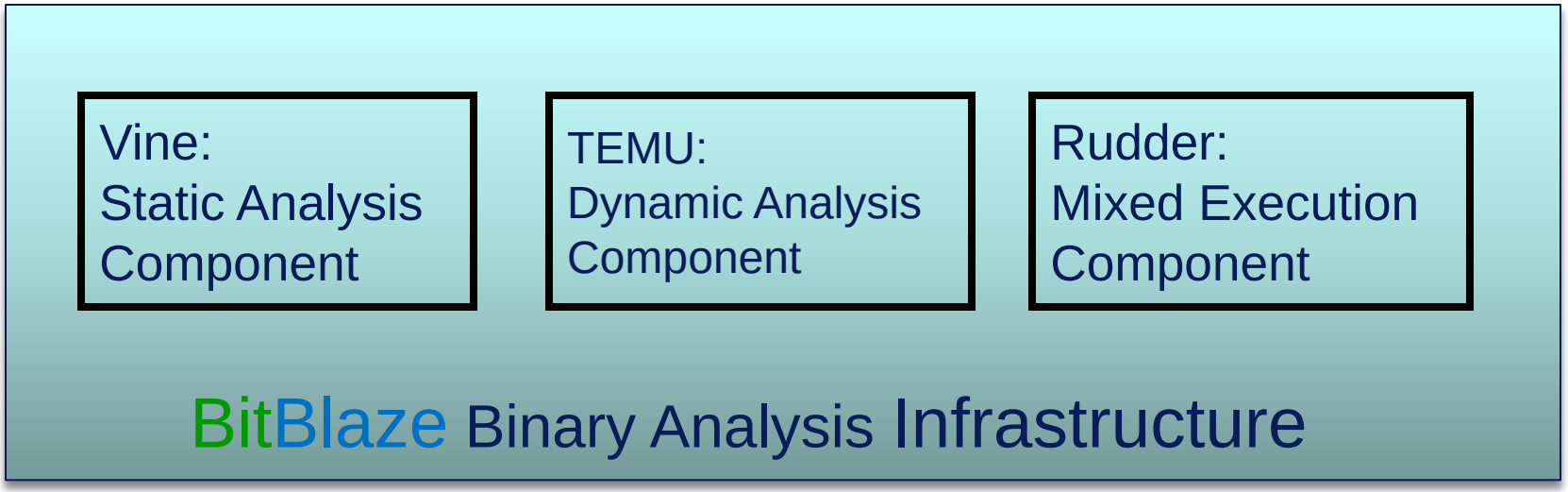
- **Website content filter modeled as Boolean predicate on the input (accepted/rejected)**
- **Browser CSA modeled as multi-class classifier**
  - **One per output MIME type (e.g., text/html or not)**
- **Query a solver for inputs that are:**
  1. **Accepted by the website's content filter**
  2. **Interpreted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

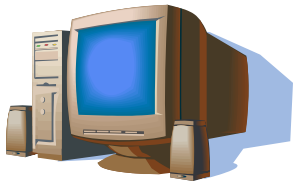
Dissecting Malware

BitBlaze Binary Analysis Infrastructure

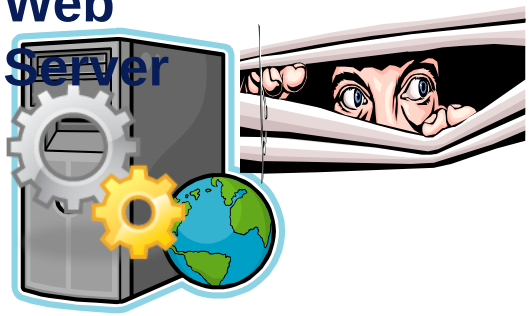
Extracting CSA from Close-sourced Browsers

- **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



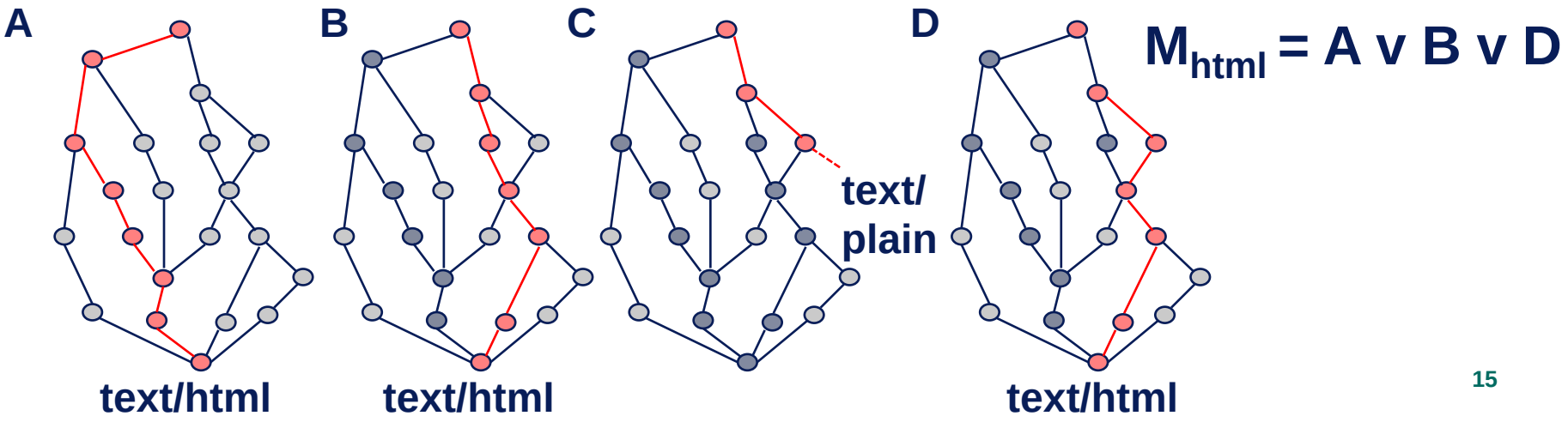
Executed instructions	Intermediate Representation (IR)
`mov(%esi), %al`	`AL = INPUT[0]`
`mov $0x47, %bl`	`BL = 'G'`
`cmp %al, %bl`	`ZF = (AL == BL)`
`jnz FAIL`	`IF(ZF==0)JMP(FAIL)`
`mov 1(%esi), %al`	`AL = INPUT[1]`
`mov $0x45, %bl`	`BL = 'E'`
`cmp %al, %bl`	`ZF = (AL == BL)`
`jnz FAIL`	`IF(ZF==0)JMP(FAIL)`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

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 Postcript 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

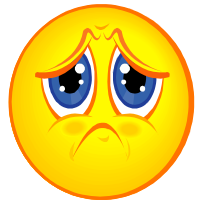
Model	Seeds	Path count	% HTML paths	Avg. # Paths per seed	Avg. Path gen. time	# Inputs generated	Avg. Path depth
Safari 3.1	7	1558	12.4%	222.6	16.8 sec	7166	12.1
IE 7	7	948	8.6%	135.4	26.6 sec	64721	212.1

- **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**

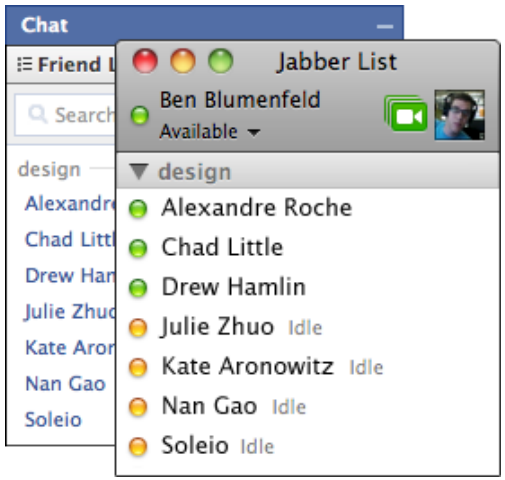
Outline

- **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**
- **Type-based Approach for Context-sensitive Automatic Sanitization in Web Templating Languages**
- **Conclusions**

Rich Web Applications

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

Gmail by Google Google maps Google docs



Client-side Validation(CSV) Vulnerabilities

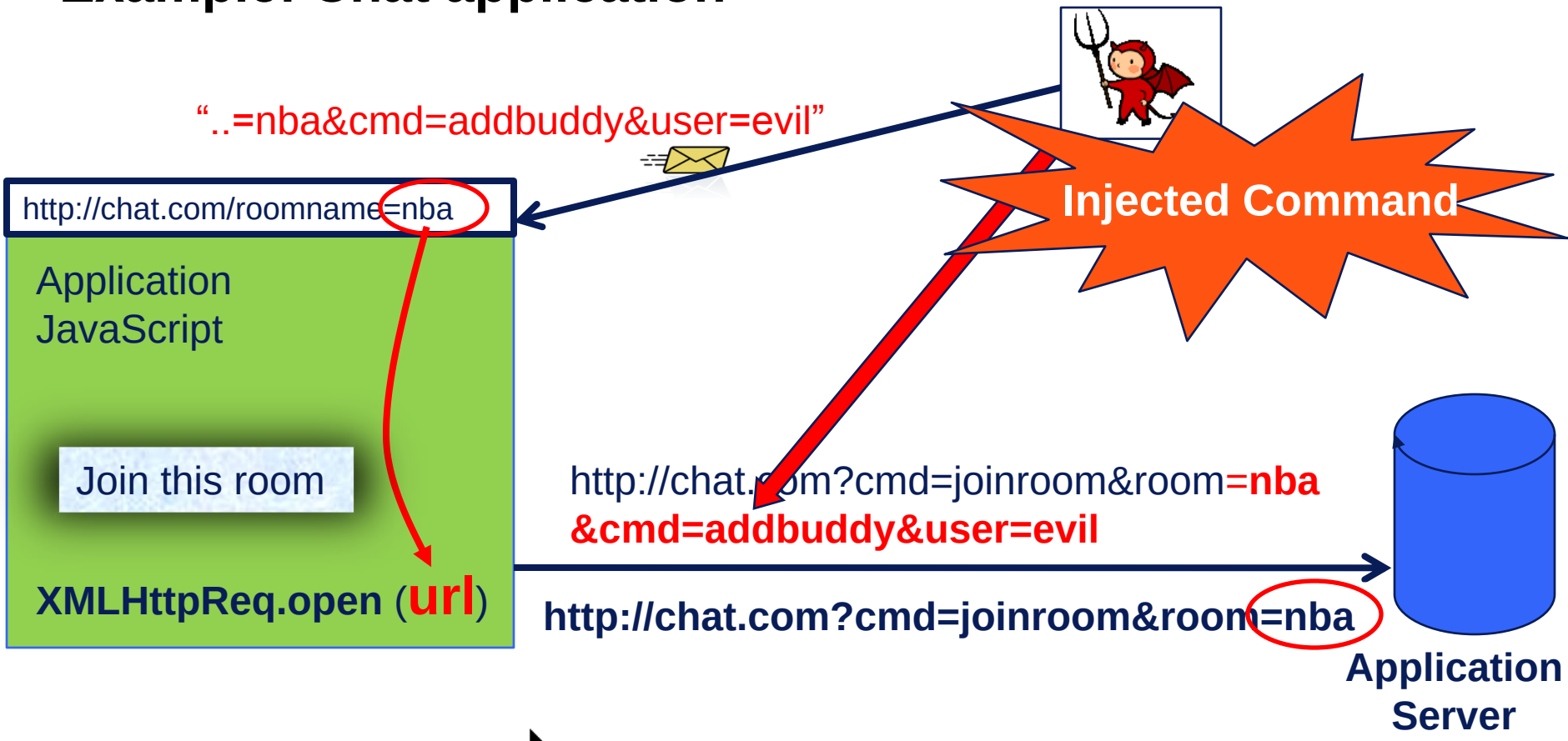
- **Most previous security analysis focuses on server side**
- **A new class of input validation vulnerabilities**
  - **Analogous to server-side bugs**
  - **Unsafe data usage in the client-side JS code**
  - **Different forms of data flow**
    - **Purely client-side, data never sent to server**
 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

Problem Definition

Automatically Find Code-Injection Vulnerabilities in JS Applications

- **Two challenges**
- **#1: Automatic exploration of the execution space**
- **#2: Automatically check if data is sanitized sufficiently**
  - **Can't distinguish parsing ops. from custom validation checks**
  - **Can't assume validation, false negatives vs. false positives.**

Our Contributions

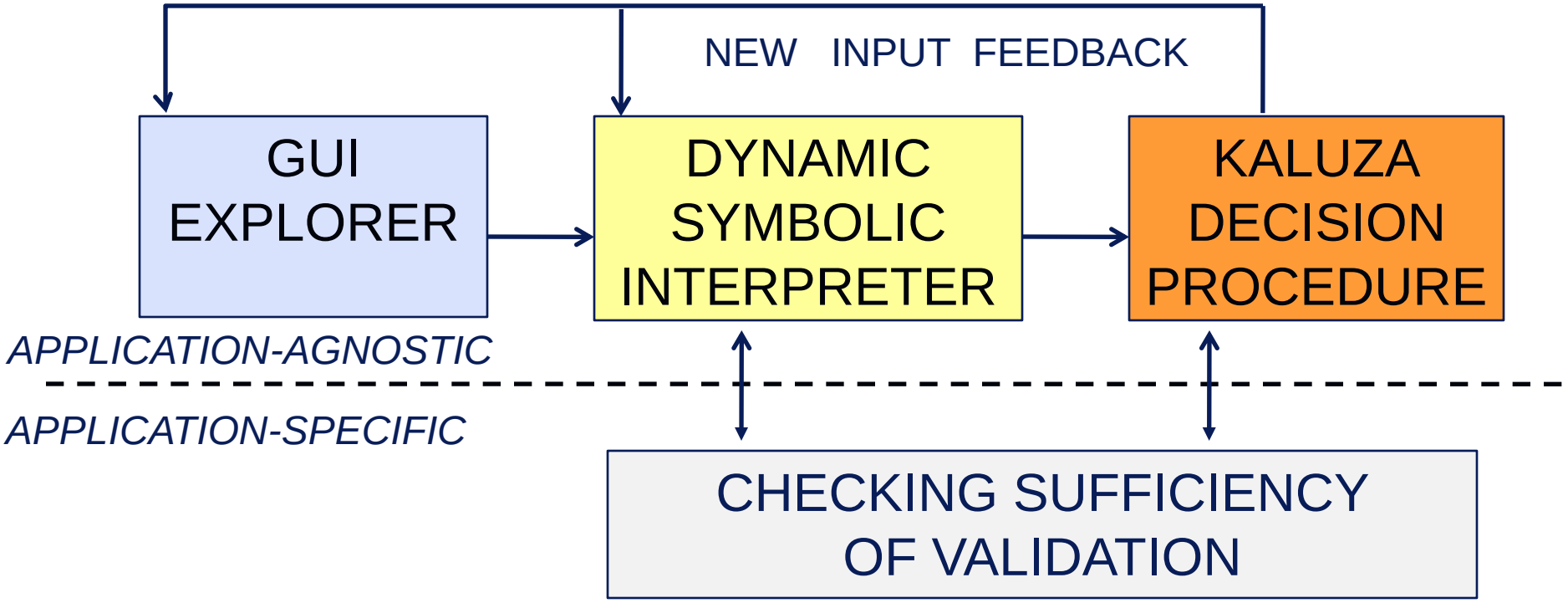
- **Existing Approaches**
  - **Static Analysis** *[Gatekeeper'09, StagedInfoFlow '09]*
  - **Taint-enhanced blackbox fuzzing** *[Flax'10]*

- **Drawbacks**
  - **Either assumes an external test suite to explore paths** *[Flax'10]*
  - **Or, does not generate an exploit instance, can have FPs** *[Gatekeeper'09, StagedInfoFlow '09]*

- **Our Contributions**
  - ***A Symbolic Analysis approach***
  - ***Kudzu:*** **An end-to-end symbolic execution tool for JavaScript**
  - **Identify a sufficiently expressive "*theory of strings*"**
  - ***Kaluza*: A new expressive, efficient decision procedure**
    - » **Supports strings, integers and booleans as first-class input variables**

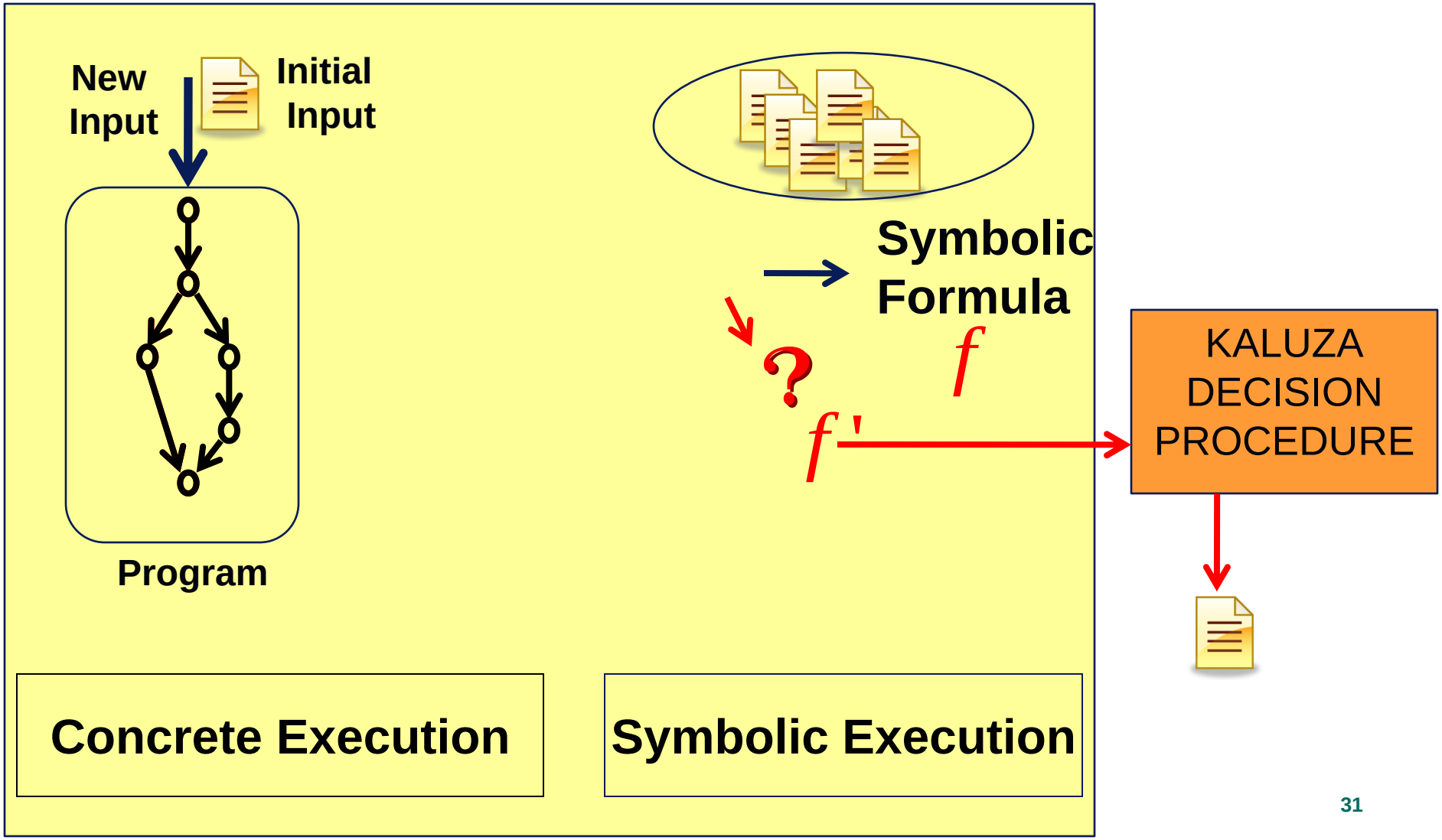
Kudzu: Approach and Design

- **Input space has 2 components**
  - **Event Space:** ***GUI explorer***
  - **Value Space:** ***Dynamic Symbolic Execution***
- **Checking sufficiency of validation checks**
  - **Symbolic analysis of validation operations on code-evaluated data**



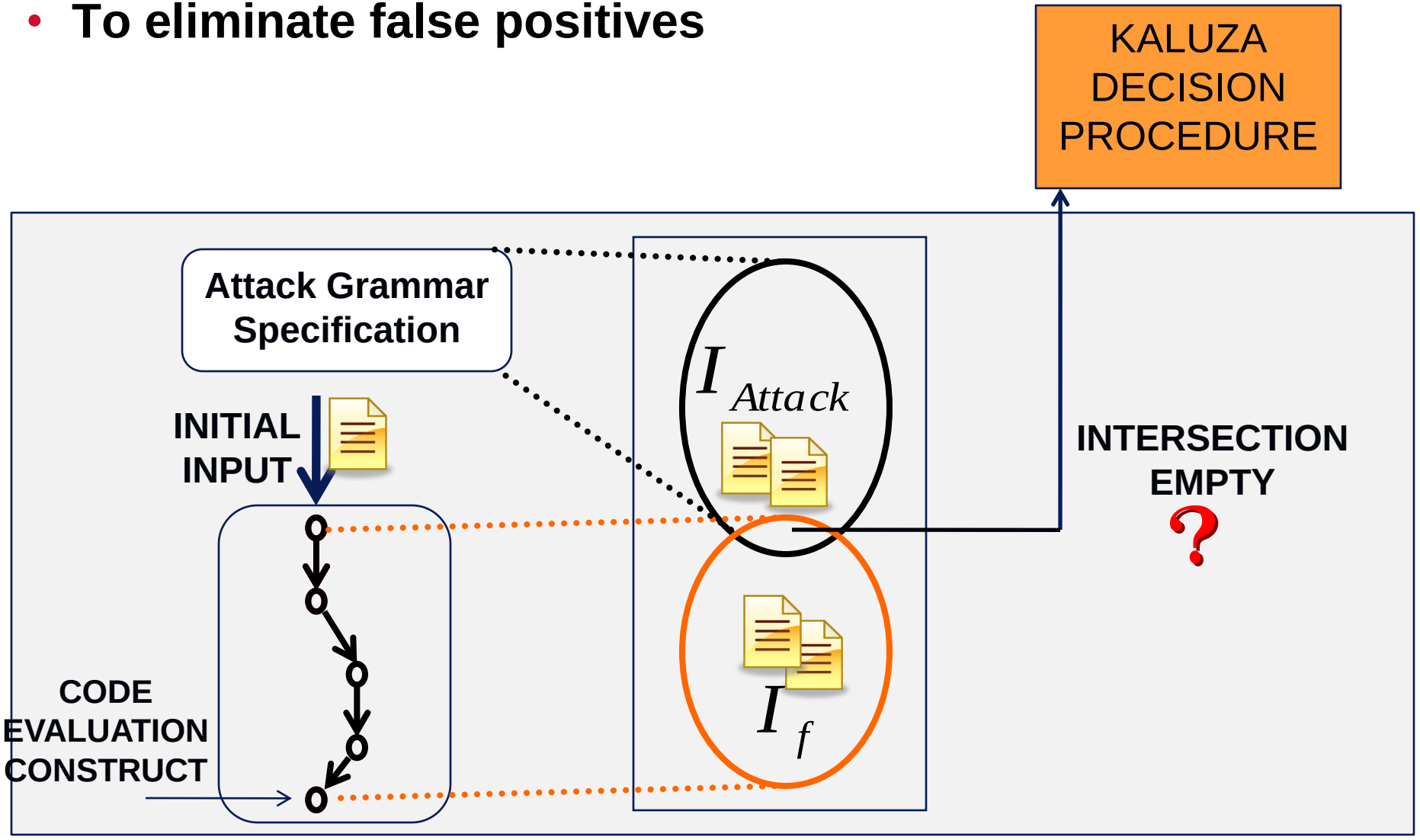
Dynamic Symbolic Interpreter for JavaScript

- Employed for Value Space Exploration



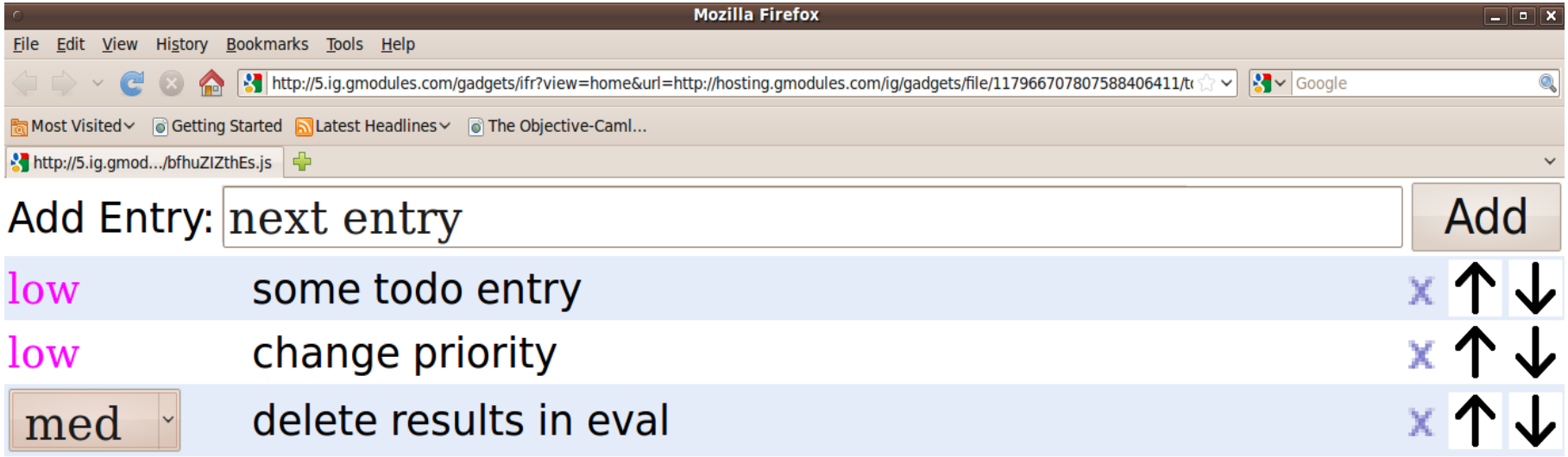
Checking Sufficiency of Validation Checks

- **To eliminate false positives**

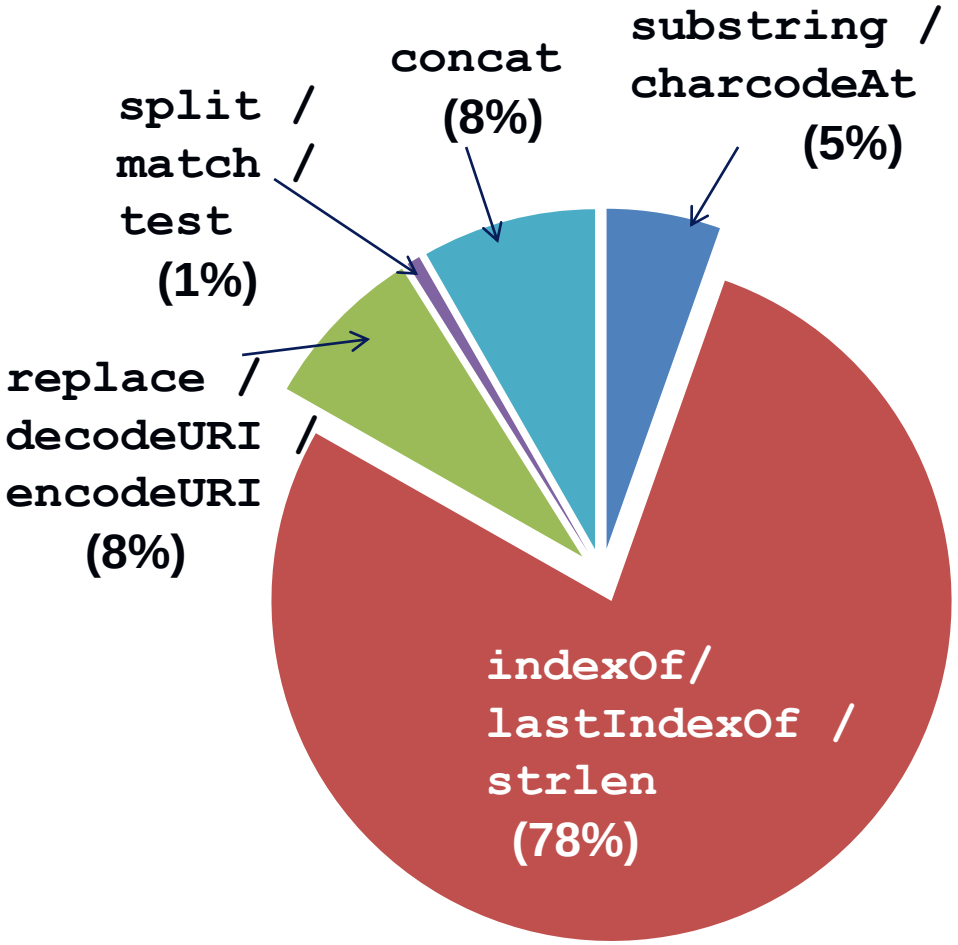


GUI Exploration

- **Events: State of GUI elements, mouse and link clicks**
- **Event Sequence: A sequence of state-altering GUI actions**
- **Event Space Exploration using a *GUI explorer***
- **Practically enhances coverage benefits**
  - **Example:**
  - **1 Gadget Vulnerability: reachable with a sequence of events executed:** dropdown box value is changed, delete hit



Empirical Motivation for A Theory of Strings



- **Combined string and integer solver**
- **Regular Expression based operations are 1/3rd of the `match, split, test, replace` operations (9%)**
- **Multiple string variables**

```
/\\(?:["\\\/bfnrt]|u[0-9a-fA-F]{4})/
```

33% regexes have Capture Groups

A Sufficiently Expressive Theory for JS

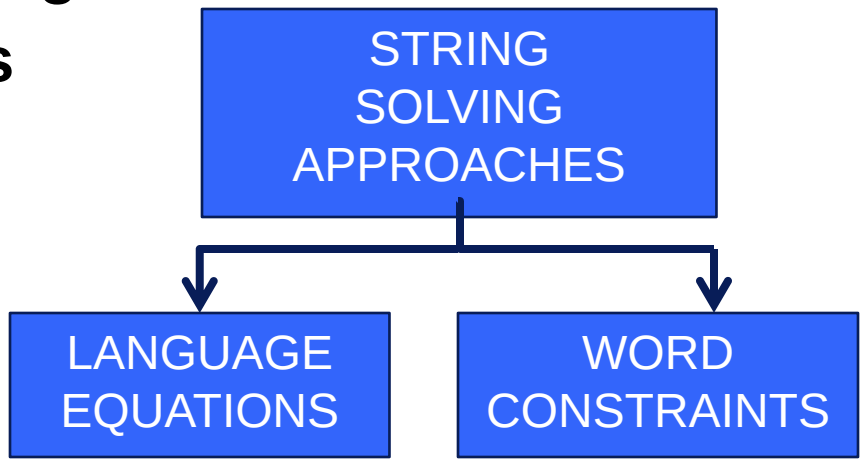
- **Practical Requirements to support**

	[DPRLE'09]	[HAMPI'09]	[PEX'09]
Concatenation (Word Equations)	✓	✓	✓
Regular Language Membership	✓	✓	✗
String Length	✗	✗	✓
Equality	✗	✓	✓
Multiple String Variables	✓	✗	✓
Boolean and Integer Logic	✗	✗	✓

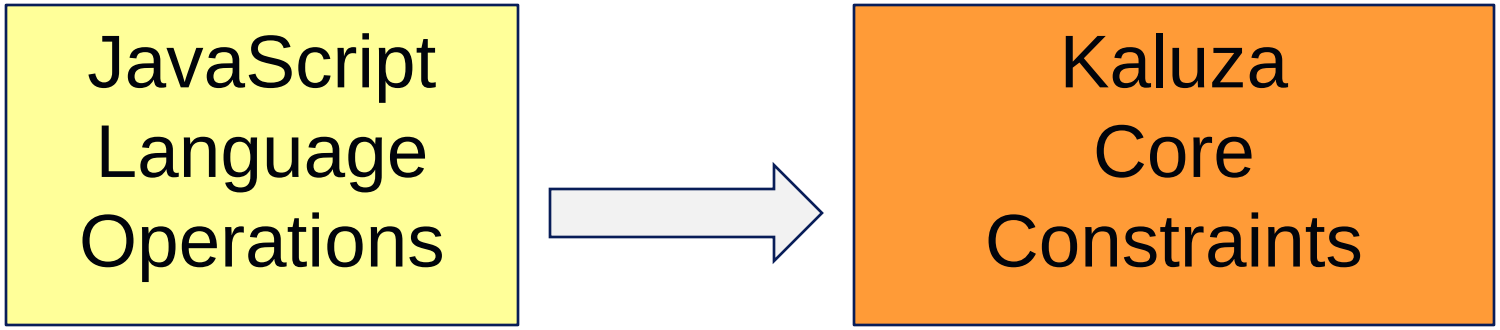
Existing solvers not sufficiently expressive

Kaluza: A New Solver Decision Procedure

- **Input: A boolean combination of constraints over *multiple* integer and *variable-length* string variables**
- **Decidability vs Expressiveness**
  - **Equality between reg language variables undecidable** *[STOC'81]*
  - **Full generality of `replace` in word constraints undecidable** *[TACAS'09]*



Insight: JS to Kaluza Reduction uses Dynamic Information



Kudzu System Evaluation

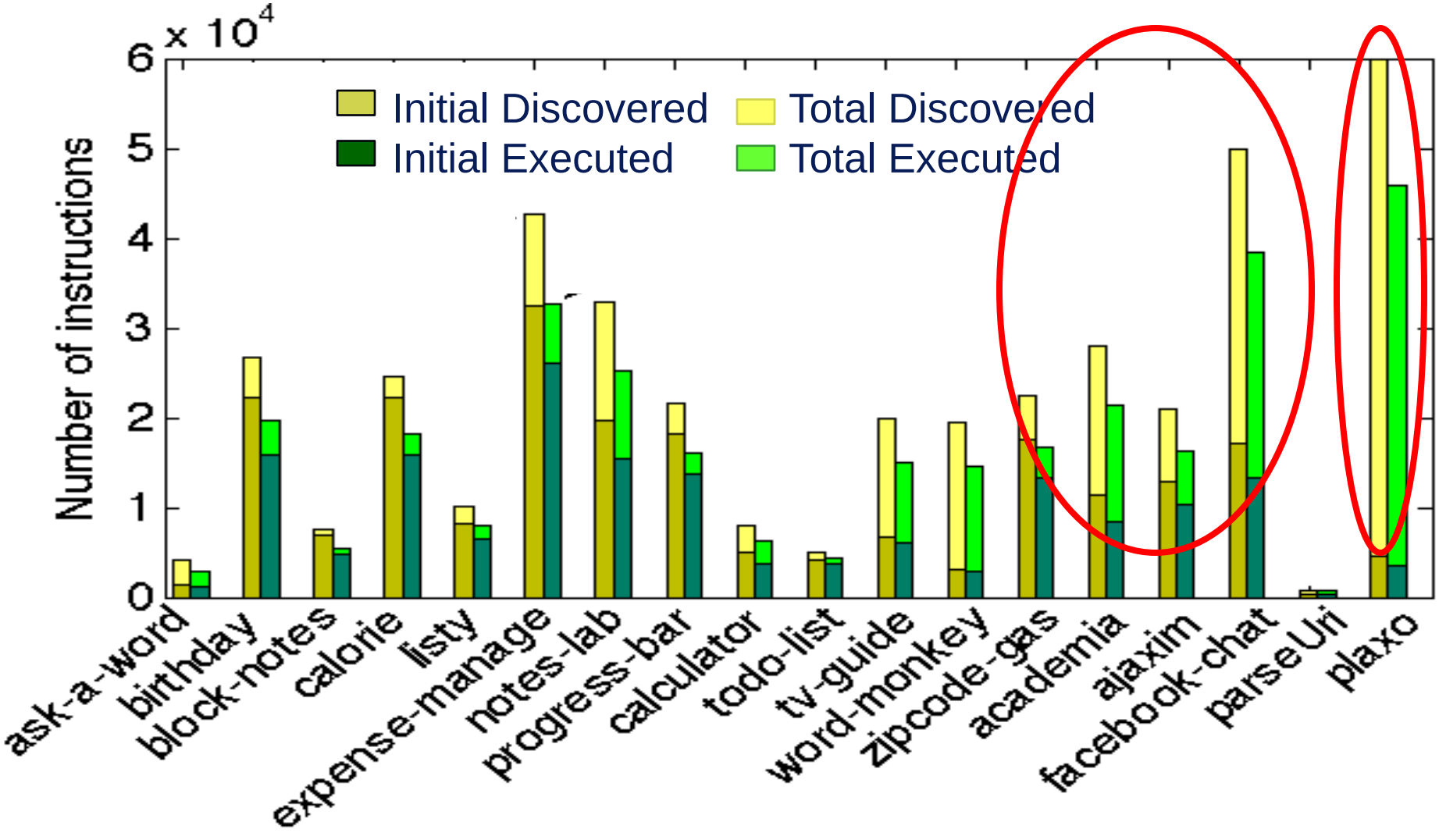
- **18 Live Applications**
  - **13 iGoogle gadgets**
  - **5 AJAX application**
    - **Social networking: Academia, Plaxo**
    - **Chat applications: AjaxIM, Facebook Chat,**
    - **Utilities: parseURI**
- **Setup**
  - **Untrusted sources**
    - **All cross-domain channels**
    - **Text boxes**
  - **Critical sinks**
    - **Code evaluation constructs**

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

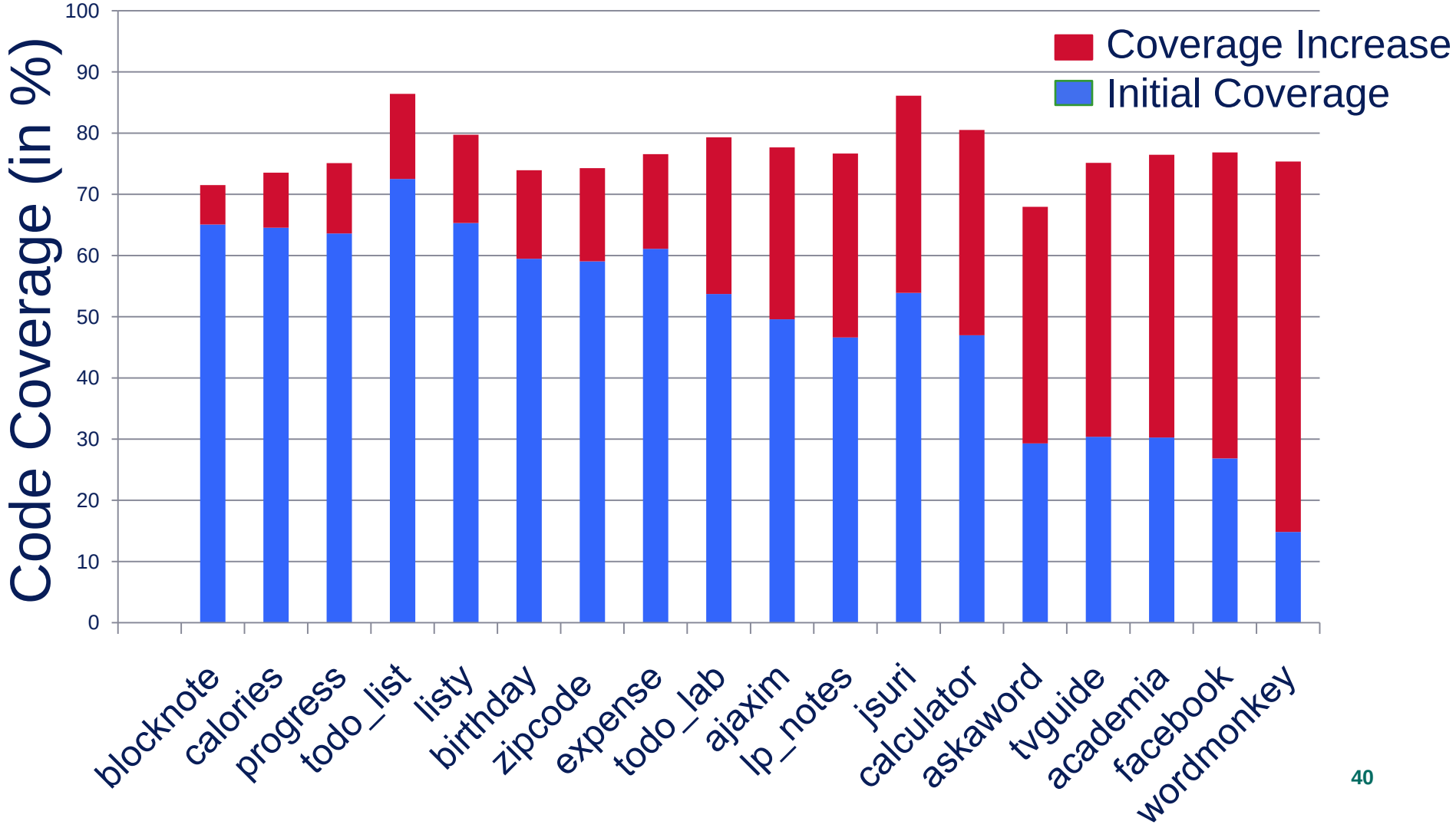
Results: Code Coverage

29% code coverage increase in 6 hours



Results: Code Coverage

29% code coverage increase in 6 hours



Summary

- **Kudzu: An End-to-end Symbolic Execution Tool for JS**
  - **Separates the input space analysis into 2 components**
- **Identified a theory of strings expressive enough for JS**
- **Kaluza: A new decision procedure for the theory**
- **Demonstrated capabilities on 18 live web applications**
- **Found 11 vulnerabilities with no given initial test harness**
- **2 new vulnerabilities**

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**

Type-based Approach for Context-sensitive Automatic Sanitization in Web Templating Languages

- **Can we prevent XSS attacks by construction?**
- **Goal: automatic sanitization in web templating languages**
- **Challenges:**
  - **Context-sensitive**
  - **Support complex language constructs (e.g., if-else, loops)**
  - **Backwards compatibility with existing code**
    - » **Co-exist with existing sanitization code**
  - **Low performance over head**

Type-Qualifier based Approach

- **Context type qualifier**
  - **Representing context where untrusted input can be safely embedded**
- **Type inference during compilation**
- **Automatically insert sanitization routine and runtime instrumentation based on type inference**
- **Deployed in Google Closure Template**
  - **Gmail, GoogleDocs**
- **Efficient: 3-9.6% overhead on CPU intensitve benchmarks**

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]***
  - ***Type-based Context-sensitive Auto-sanitization in web frameworks***
- **Do different web protocols interact securely?**
  - ***Model checking web protocols [CSF 10]***

BitBlaze: Computer Security 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

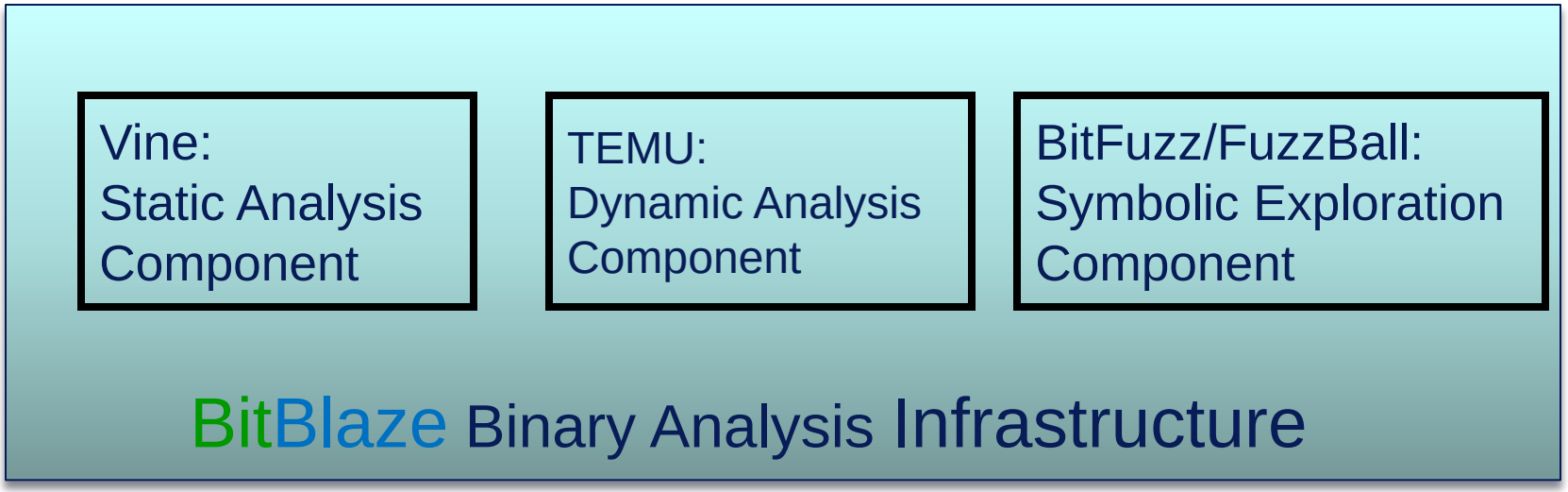
Generating Filters

Dissecting Malware

BitBlaze Binary Analysis Infrastructure

BitBlaze Binary Analysis Infrastructure: Architecture

- **The first infrastructure:**
  - **Novel fusion of static, dynamic, formal analysis methods**
    - » **New symbolic reasoning techniques**
  - **Whole system analysis (including OS kernel)**
  - **Analyzing packed/encrypted/obfuscated code**



BitBlaze in Action (I): Vulnerability Discovery

- **Loop extended symbolic execution [ISSTA09]**
- **Decomposition-&-re-stitching symbolic execution [CCS10]**
  - **Finding vulnerabilities in malware**
- **Statically-Directed Dynamic Automated Test Generation [ISSTA11]**
  - **Dynamic-static-dynamic approach**
- **Model-inference Assisted Concolic Execution [USENIX Security 11]**
- **On-the-spot symbolic execution**
  - **Finding bugs in binary emulators**

BitBlaze in Action (II): Vulnerability Diagnosis & Defense

- **Differential Slicing: Identifying Causal Execution Differences for Security Applications [IEEE S&P 11]**
- **Automatic Patch-Based Exploit Generation is Possible: Techniques and Implications [IEEE S&P 08]**
- **Automatic Generation of Vulnerability Signatures [IEEE S&P 06, CSF 07, RAID09]**

BitBlaze in Action (III): Model Extraction

- **Secure Content Sniffing for Web Browsers [IEEE S&P 09]**

- **Inference and Analysis of Formal Models of Botnet Command and Control Protocols [CCS 07, 09, 10]**

BitBlaze in Action (IV): In-depth Malware Analysis

- **High volume of new malware needs automatic malware analysis**
- **Given a piece of suspicious code sample,**
  - **What malicious behaviors will it have?**
  - **How to classify it?**
    - **Key logger, BHO Spyware, Backdoor, Rootkit**
  - **What mechanisms does it use?**
    - **How does it steal information?**
    - **How does it hook?**
  - **Who does it communicate with? Where does it send information to?**
  - **Does its communication exhibit certain patterns?**
  - **Does it contain trigger-based behavior?**
    - **Time bombs**
    - **Botnet commands**
- **BitBlaze Malware Analysis Engine: a unified framework for in-depth malware analysis**

BitBlaze Summary

- **New techniques on binary analysis for security applications**
  - **Scale to large real-world programs**
  - **Fusion of static, dynamic analysis & symbolic reasoning**
  - **New problem formulation & approaches for security problems**
- **Unified framework for broad spectrum of security problems**
- **Partially open source**
  - **Empower further development worldwide**

Android App Security

- **Billions of android app downloads**
- **Android market**
  - **$25 signup**
  - **Anyone can publish**
  - **Anonymous sign-up possible**
- **Third-party market**
- **How to check & ensure that an android app is secure to download?**

Security Issues of Android Apps

- **Malicious app**
  - **Exploit vulnerabilities in Android kernel & platform**
  - **Exploit vulnerabilities in other apps**
  - **Stealing users' data**
  - **Paid SMS**
  - **Botnets: download malicious payload & launch other malicious activities**

- **Vulnerable app**
  - **Fail to protect its own data**
  - **SQL injection attacks**
  - **Confused deputy attacks: allow other apps to use its permission**

Automatic Analysis of Android Apps

- **Does the app have vulnerabilities?**
- **Is the app malicious?**
- **Interesting behaviors of app**
  - **Network behaviors: where does it communicate to?**
  - **SMS**
  - **Location info**
  - **Download code to execute**
  - **Interact with other apps**
  - **Exploit vulnerabilities in kernel or platform**

DroidBlaze: Automatic Analysis Infrastructure for Android Apps

- **Combining static & dynamic analysis**
- **Automatic exploration of program execution space**
  - **Identifying trigger-based behavior**
- **Similarity and clustering analysis**
- **Behavioral and semantic analysis**
- **Current results**
  - **Over permission analysis**
  - **In-app billing vulnerability analysis**
  - **Malware detection**

Conclusion

- **WebBlaze: New Technologies for Enhancing Web Security**
- **BitBlaze: Binary Analysis for Computer Security**
- **DroidBlaze: Automatic Security Analysis for Android Apps**





bitblaze.cs.berkeley.edu

webblaze.cs.berkeley.edu

dawnsong@cs.berkeley.edu