secure sba-research.org

COMET

Competence Centers for Excellent Technologies

Combinatorial Security Testing: Combinatorial Testing Meets Information Security

Dimitris E. Simos

SBA Research

Applied & Computational Mathematics Division Seminar Series
National Institute of Standards and Technology (NIST)
Gaithersburg, MD, USA
September 22, 2015

secure sba-research.org

bmvit Bundesministerium für Verkehr, Innovation und Technologie
bmwfj Bundesministerium für Wirtschaft, Familie und Jugend
Christian Doppler Forschungsgesellschaft
FFG
wirtschaftsagentur wien
zit Die Technologieagentur der Stadt Wien
MARIE CURIE ACTIONS

Who is Talking?

- Current Positions
  - 03.2014 - now: Key Researcher, SBA Research, Austria
  - 03.2014 - now: Combinatorics, Codes and Information Security (CCIS) Group Leader, SBA Research, Austria
    - Design Combinatorics and Codes
    - Error Correcting Codes for Post-quantum Cryptography
    - Combinatorial Testing for Information Security
  - 03.2014 - now: Adjunct Lecturer, Vienna University of Technology
- Past Positions
  - 03.2013 – 02.2015: Marie Curie Fellow, SBA Research, Austria
  - 03.2012 – 02.2013: Marie Curie Fellow, INRIA Paris-Rocquencourt, SECRET Team, France
- Ph.D. Thesis
  - 11.2011: Discrete Mathematics & Combinatorics, NTUA, Greece
- Honors and Awards
  - 03.2012: Fellow of the Institute of Combinatorics and its Applications (FTICA), ICA, Canada
  - 12.2011: ERCIM "Alain Bensoussan" Fellowship, ERCIM/EU co-fund
- Publication Record
  - Around 60 papers in Discrete Mathematics and their applications in Computer Science

Acknowledgements for this Talk

- **CCIS Group @ SBA Research:** Bernhard Garn, Kristoffer Kleine, Ludwig Kampel, Peter Aufner
- **CCIS Alumni:** Manuel Leitner, Raschin Tavakoli, Ioannis Kapsalis
- **Collaborators @ SBA Research:** Artemios Voyiatzis, Martin Graf, Severin Winkler, Andreas Bernauer
- **External Collaborators:** Raghu Kacker, Rick Kuhn, Jeff Lei, Franz Wotawa, Josip Bozic, Paris Kitsos, Jose Torres-Jimenez

SBA Research at a Glance

Mission

- Advance the field of Information Security through basic & applied research
- The largest non-profit research center in Austria that exclusively addresses Information Security ($\approx$ 80 researchers & security experts)

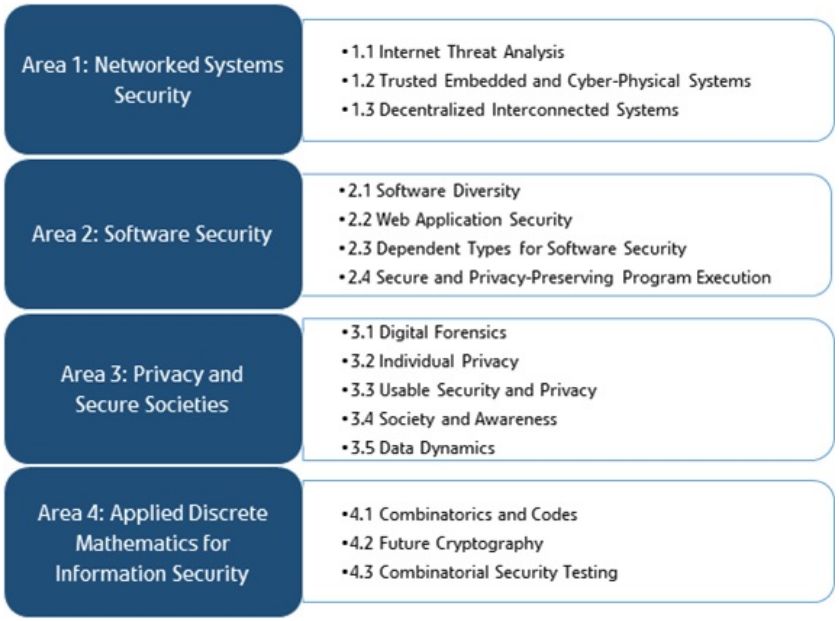


Figure: Research Programme for 2017-2025

Outline of the Talk

Introduction
- Combinatorial Testing
- Recent Results

Outline of the Talk

Introduction
- Combinatorial Testing
- Recent Results

Web Security Testing
- Challenges
- Milestones

Outline of the Talk

Introduction
- Combinatorial Testing
- Recent Results

Web Security Testing
- Challenges
- Milestones

Kernel Testing
- Challenges
- Milestones

Outline of the Talk

Introduction
- Combinatorial Testing
- Recent Results

Web Security Testing
- Challenges
- Milestones

Kernel Testing
- Challenges
- Milestones

Combinatorial Security Testing
- Achievements
- Vision
- Network Security
- Hardware Malware

Outline of the Talk

Introduction
- Combinatorial Testing
- Recent Results

Web Security Testing
- Challenges
- Milestones

Kernel Testing
- Challenges
- Milestones

Combinatorial Security Testing
- Achievements
- Vision
- Network Security
- Hardware Malware

Research Problems

Combinatorial Testing

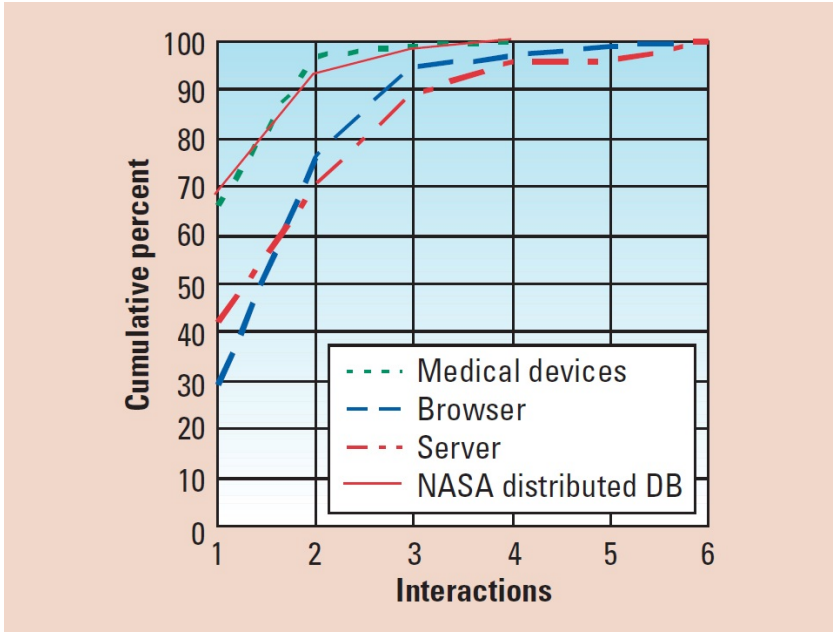
Motivation: Why Combinatorial Testing for Information Security?

- We cannot test everything
- **Combinatorial explosion:** Exhaustive search of input space increases time needed exponentially
- **Domain-specific:** Modeling of security vulnerabilities

Combinatorial Testing (CT)

- Provide 100% coverage of t-way combinations of input parameters; higher interaction strength t reveals more faults (conjecture)
- Ensure automation during test generation
- Fault localization, coverage measurement

Empirical Evidence: Fault Coverage vs. Interactions



- Rick Kuhn, Yu Lei, and Raghu Kacker. 2008. Practical Combinatorial Testing: Beyond Pairwise. IT Professional 10, 3 (May 2008), 19-23. `http://dx.doi.org/10.1109/MITP.2008.54`
- 1 interaction: enter value *age* > 100 and device crashes
- 2 interactions: *age* > 100 and *zip-code* $= 5001$, DB push fails
- 3 interactions: $a = 2$ and $b =$ *FALSE* and *update* $=$ *Tuesday*, system enters infinite loop

Technical Challenges

- Generation of optimal covering arrays is NP-hard
  - These arrays form test suites
- Modeling parameters, values, constraints (domain specific)
  - Generate test inputs or system configurations

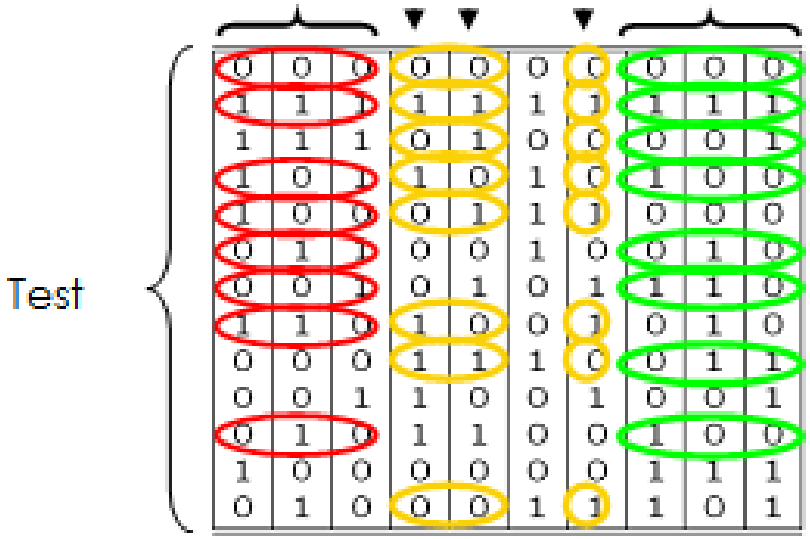


Figure: A covering array (CA) with 10 boolean parameters and 13 tests. Every 3-way combination is covered at least once

Recent Results

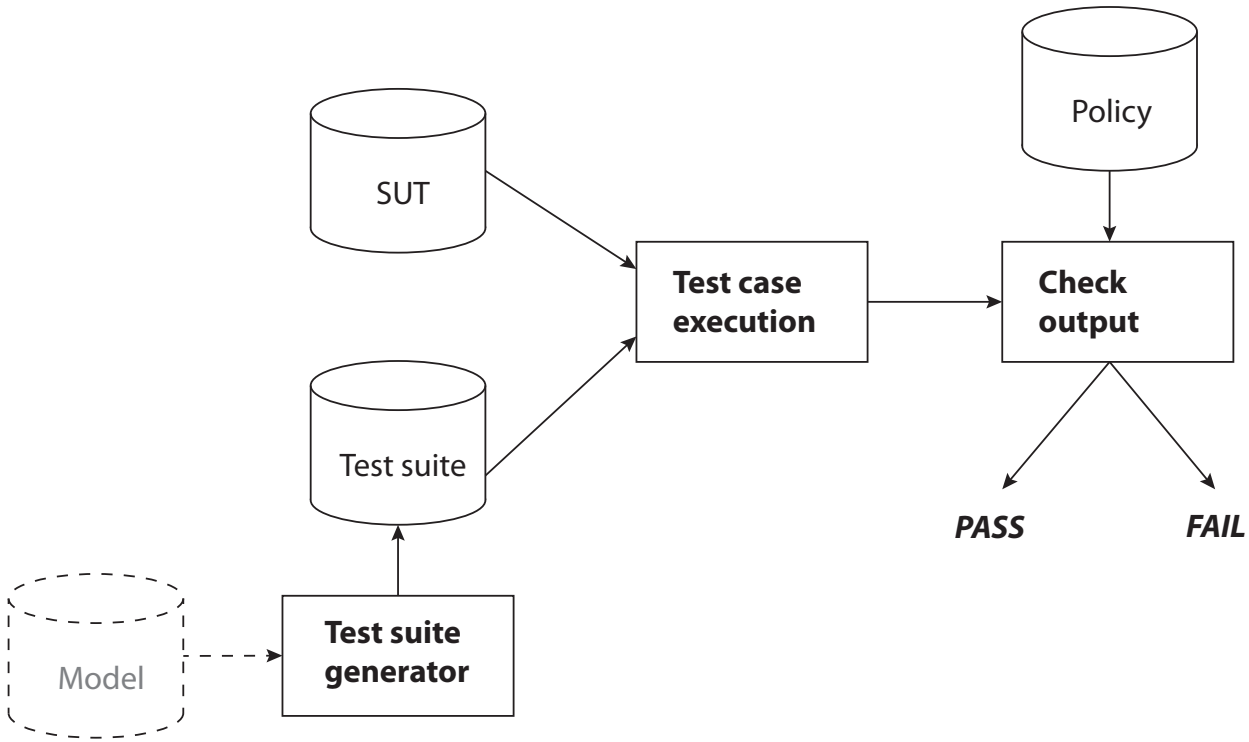
Focus

- Modelling of Covering Arrays (CAs)
- Optimization algorithms for combinatorial testing
- Relation of CAs with error-correcting codes

Milestones (ACA2015, RTA2015)

- Modelling vertical extension of In-Parameter-Order (IPO) strategy (Lei et al.) in terms of computational algebra algorithms
- Construction of symbolic test suites
  - Expressing the constraints as systems of multivariate polynomial systems
  - Rewriting techniques (equational unification) via Groebner bases

Components of a Testing Framework



This Talk

- **Automated generation** of test cases for security testing
- **Evaluation** of the applicability of combinatorial testing

Web Security Testing

Focus

- Modelling of attack vectors and exploitation of XSS vulnerabilities
- SUTs: Everything running in your browser!

Challenges

- Reduce the JavaScript language complexity to (XSS) injection attacks
  - Semantically infeasible to determine
  - XSS one of top vulnerabilities in OWASP Top 10
- Ensure automation, generate quality vectors and saving of resources
  - Most testing tools (BURP, ZAP) require interaction from the tester; reduction of test suites w.r.t. bypassing defense mechanisms;
- Real-world testing far away from academic approaches
  - Translation between combinatorics, software testing and penetration testing;

Generation of XSS Attack Vectors

Cross-Site-Scripting (XSS)

- Inject client-side script(s) into web-pages viewed by other users
- Malicious (JavaScript) code gets executed in the victim's browser

Valid URLs vs Attack Vectors

- normal case: `http://www.foo.com/error.php?msg=hello`
- attacker injects client-side script in parameter msg:
  `http://www.foo.com/error.php?msg=<script>alert(1)</script>`

Input Parameter Modelling for XSS Attack Vectors

$$AV := (parameter_1, parameter_2, \ldots, parameter_k)$$

A BNF Grammar for XSS Attack Vectors

A Fragment of The Grammar G

```
JSO(15)::= <script> | <img | ...
WS1(3)::= tab | space | ...
INT(14)::= "'; | ">> | ...
WS2(3)::= tab | space | ...
EVH(3)::= onLoad( | onError( | ...
WS3(3)::= tab | space | ...
PAY(23)::= alert('XSS') | ONLOAD=alert('XSS') | ...
WS4(3)::= tab | space | ...
PAS(11)::= ') | '> | ...
WS5(3)::= tab | space | ...
JSE(9)::= </script> | > | ...
```

Table: Different sizes of test suites for $MCA(t, 11, (3, 3, 3, 3, 3, 3, 9, 11, 14, 15, 23))$

Str.	G		G_c	
	IPOG	IPOG-F	IPOG	IPOG-F
2	345	345	250	252
3	4875	4830	1794	2012
4	53706	53130	8761	9760

A Sample of XSS Attack Vectors

values

parameters

- [Root Node]
  - [SYSTEM-XSS-Mode
    - fobracket
      - <
    - tag
      - img
      - frame
      - src
      - script
      - body
      - HEAD
      - BODY
      - iframe
      - IFRAME
      - SCRIPT
    - fcbracket
      - >
    - quote1
    - space
    - event
    - quote2
    - payload
    - lobracket
    - closingtag
    - lcbracket
    - Relations
    - Constraints

	FOBRACKET	TAG	FCBRACKET	QUOTE1	SPACE	EVENT	QUOTE2	PAYLOAD	LOBRACKET	CLOSINGTAG	LCBRA
1	<	img	>	'	\t	onmouseover	'	alert(0)	</	img	>
2	<	frame	>	nil	\r	onerror	nil	alert(document.cookie)	</	frame	>
3	<	src	>	"	\r\n	onfire	"	alert("hacked")	</	src	>
4	<	script	>	'	\a	onbeforeload	nil	alert('hacked')	</	script	>
5	<	body	>	nil	\b	onafterload	"	alert(1)	</	body	>
6	<	HEAD	>	"	\c	onafterlunch	nil	alert(0)	</	HEAD	>
7	<	BODY	>	'	_	onload	"	alert(document.cookie)	</	BODY	>
8	<	iframe	>	nil	\n	onchange	'	alert("hacked")	</	iframe	>
9	<	IFRAME	>	"	\t	onclick	'	alert('hacked')	</	IFRAME	>
10	<	SCRIPT	>	"	\r	onmouseover	nil	alert(1)	</	SCRIPT	>
11	<	img	>	nil	\r\n	onclick	nil	alert(document.cookie)	</	img	>
12	<	frame	>	'	\a	onclick	"	alert("hacked")	</	frame	>
13	<	src	>	nil	\b	onclick	'	alert(0)	</	src	>
14	<	script	>	nil	\c	onclick	'	alert(1)	</	script	>
15	<	body	>	"	_	onclick	nil	alert('hacked')	</	body	>
16	<	HEAD	>	'	\n	onclick	"	alert(1)	</	HEAD	>
17	<	BODY	>	"	\r	onclick	'	alert(0)	</	BODY	>
18	<	iframe	>	'	\r\n	onclick	"	alert('hacked')	</	iframe	>
19	<	SCRIPT	>	nil	\a	onclick	'	alert(document.cookie)	</	SCRIPT	>
20	<	frame	>	"	\b	onmouseover	'	alert('hacked')	</	frame	>
21	<	src	>	'	\c	onmouseover	nil	alert(document.cookie)	</	src	>
22	<	script	>	nil	_	onmouseover	"	alert("hacked")	</	script	>
23	<	body	>	'	\n	onmouseover	'	alert(0)	</	body	>
24	<	HEAD	>	nil	\r\n	onmouseover	'	alert("hacked")	</	HEAD	>
25	<	BODY	>	nil	\a	onmouseover	nil	alert(1)	</	BODY	>
26	<	iframe	>	"	\t	onmouseover	nil	alert(document.cookie)	</	iframe	>
27	<	IFRAME	>	'	\r	onmouseover	"	alert("hacked")	</	IFRAME	>
28	<	img	>	"	\b	onerror	"	alert("hacked")	</	img	>
29	<	src	>	'	_	onerror	'	alert(1)	</	src	>
30	<	script	>	"	\n	onerror	nil	alert(0)	</	script	>
31	<	body	>	nil	\t	onerror	"	alert("hacked")	</	body	>

Figure: Figure in ACTS generation tool (Courtesy of NIST)

Evaluation Results

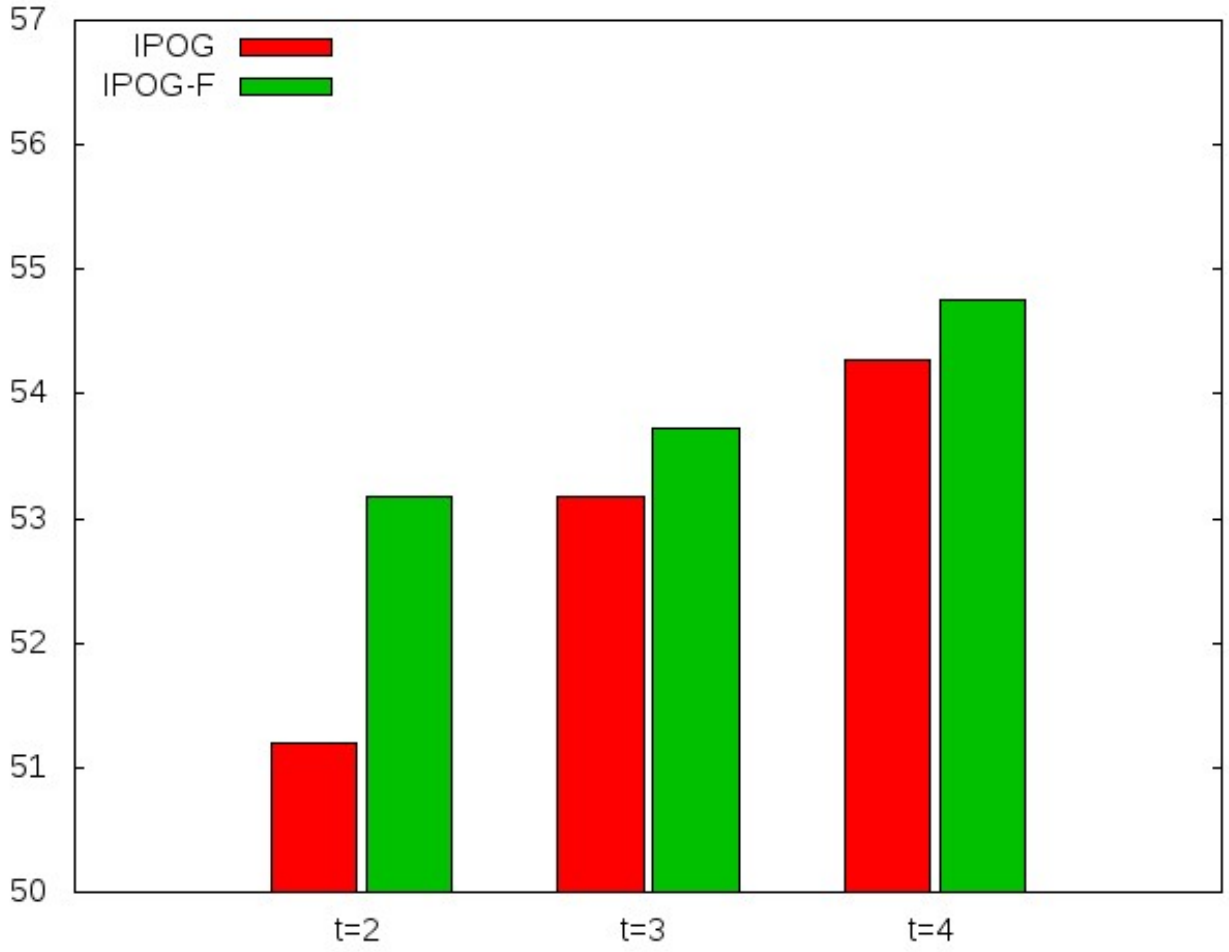


Figure: Exploitation Rate ($\frac{\#pos}{\#tot}$) Comparison: IPOG vs IPOG-F for `G_c` using BURP in DVWA

Comparison: CT vs fuzzers

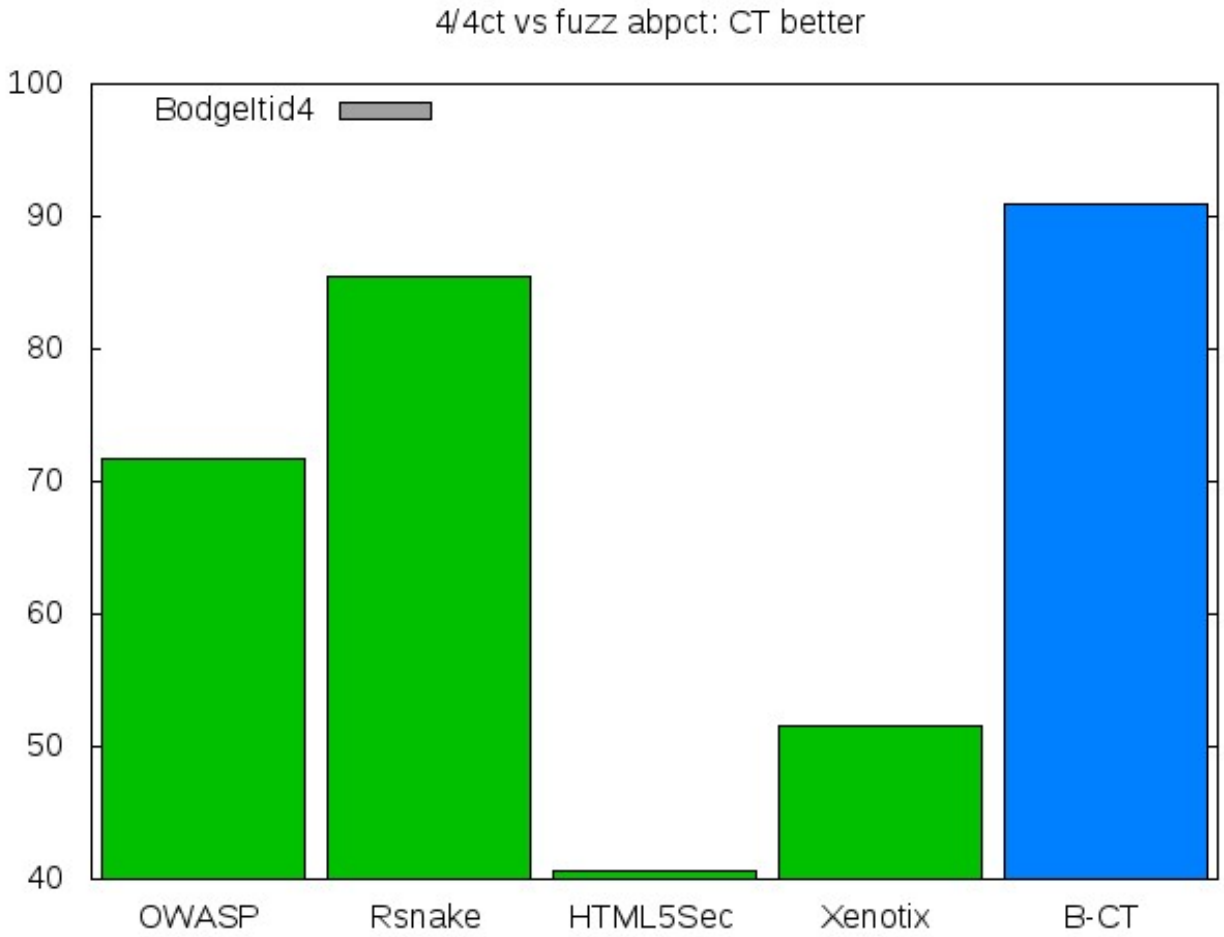


Figure: Exploitation Rate ($\frac{\#pos}{\#tot}$) Comparison: Attack Pattern-based CT vs fuzzers

Measurement Analysis in CCM Tool

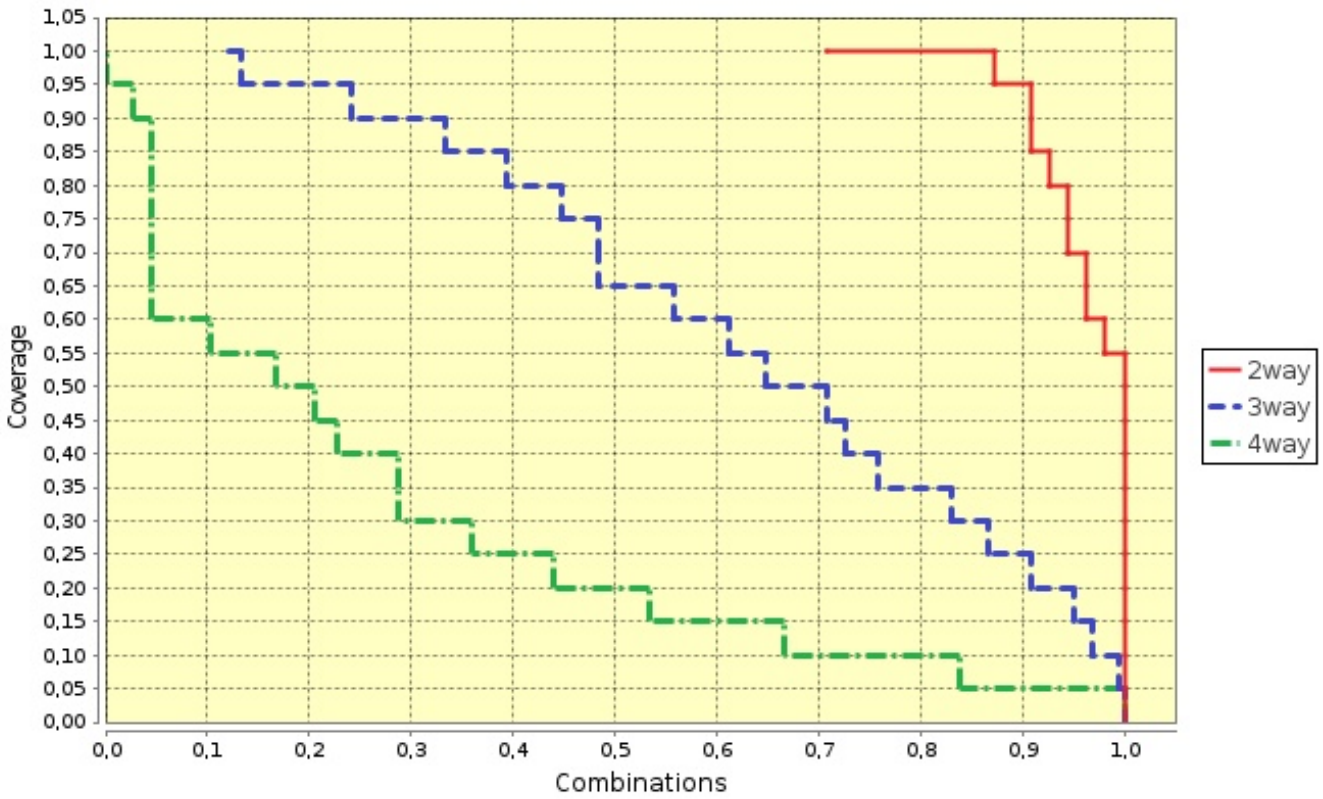


Figure: Comparison of combination coverage measurement for passing tests in DVWA (inp_id 1, DL 0) when their respective test suites are generated in IPOG-F with interaction strength $t = 2$.

Multiple XSS Vulnerabilities in Koha Library

Penetration Tests for Koha Library

- **SUT:** open source Integrated Library System (used by Museum of Natural History in Vienna, UNESCO, Spanish Ministry of Culture)
- **Results:** unauthenticated SQL Injection, Local File Inclusions, XSS
- **References:** CVE-2015-4633, CVE-2015-4632, CVE-2015-4631

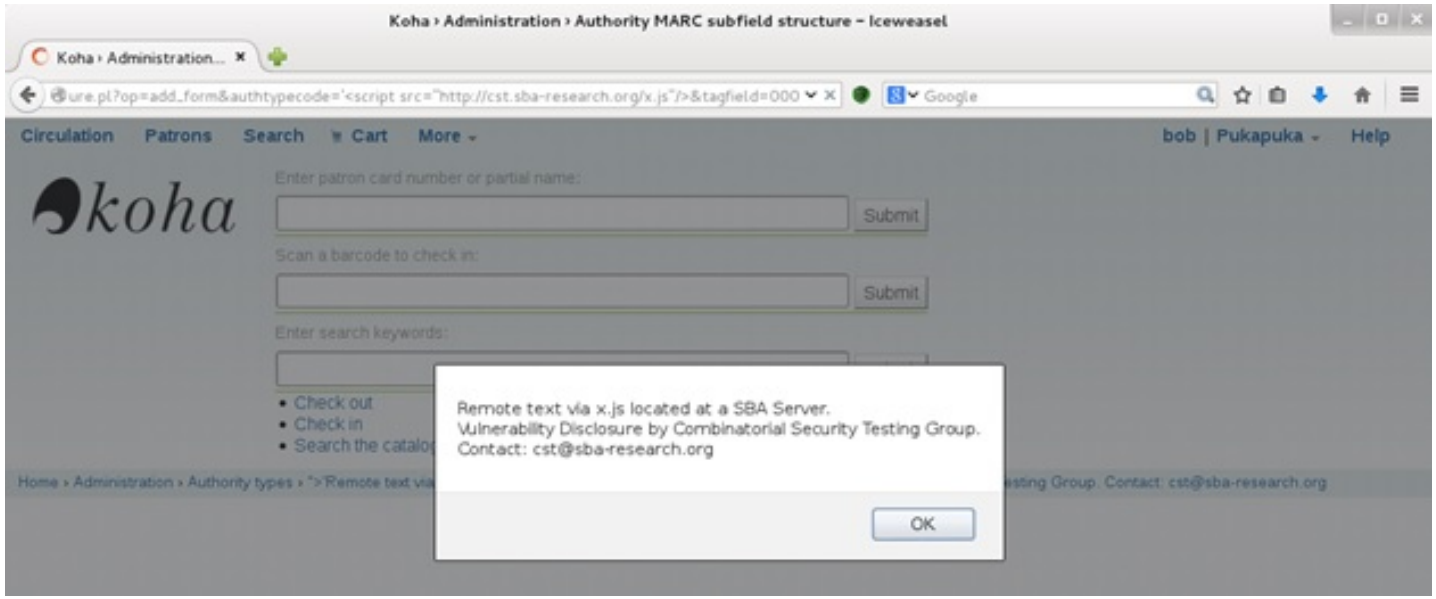


Figure: One of the vulnerabilities found by XSSInjector (Prototype tool for automated mounting of XSS attacks)

W3C Vulnerability

Scan of the Whole W3C Website

- **www:** 122 URLs, **Services:** 1 URL, **Validator:** 56 URLs
- **Acknowledgements**: Ted Guild and Rigo Wenning (W3C Team)

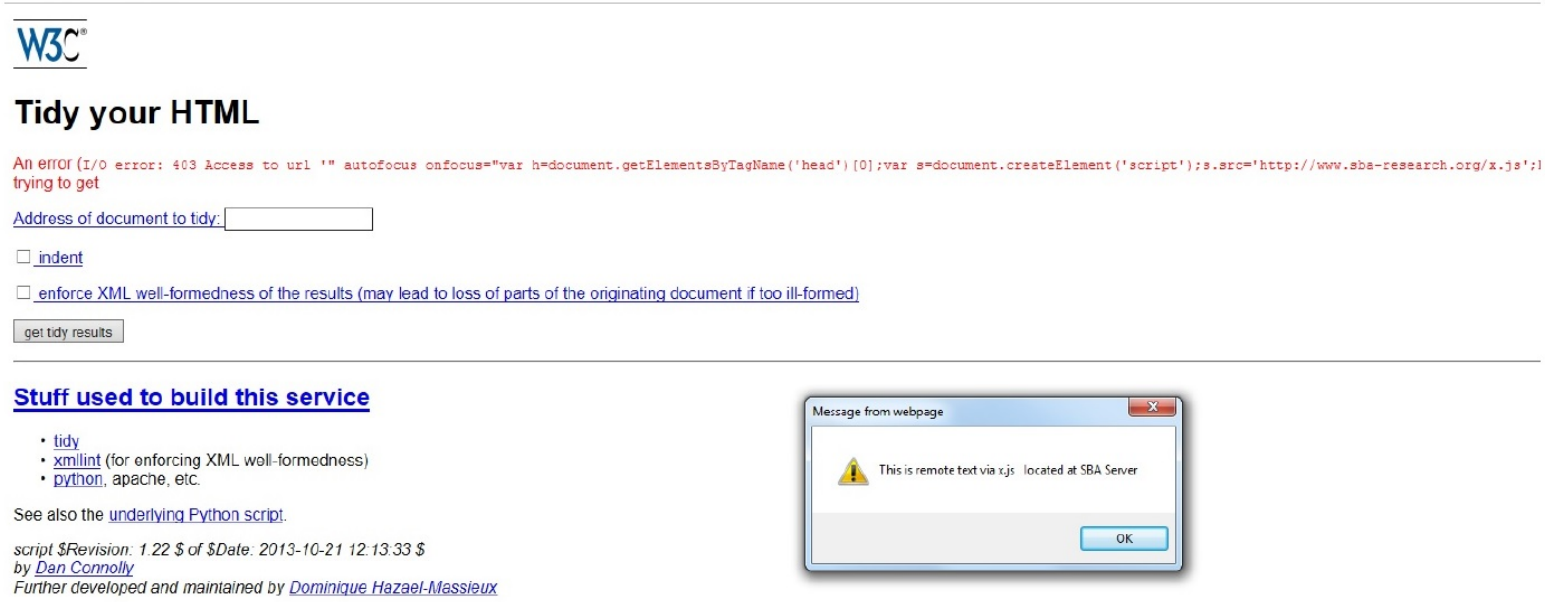


Figure: Vulnerability found in tidy service using XSSInjector (Prototype tool for automated mounting of XSS attacks)

Milestones

Expertise at SBA Research

- Knowledge transfer of combinatorial designs $\Longrightarrow$ combinatorial testing
- Benefit from experts' domain knowledge (penetration testers)

Milestones (AST/ICSE2014, JAMAICA/ISSTA2014, IWCT/ICST2015, QRS2015)

- **Modelling:** Combinatorial attack grammars via IPM
  - Automated translation layers $\Longrightarrow$ largest repo of XSS attack vectors (ahead of IBM AppScan, OWASP Xenotix)
- **XSSInjector:** Prototype tool for automated mounting of XSS attacks
- **Experience Reports:** Multi-dimensional (Comparison of SUTs, attack grammars, algorithms, fuzzers, penetration testing tools)
  - Exploits caused due to interaction of a few parameters
  - Combinatorial coverage measurement (CCM) of passing tests
- **Real-World Vulnerabilities:** XSS in tidy service (HTML validation) of W3C portal, multiple XSS in Koha Library

Kernel Testing

Focus

- Modelling of Linux system call API
- SUTs: Everything (system calls) the kernel needs to be operational!

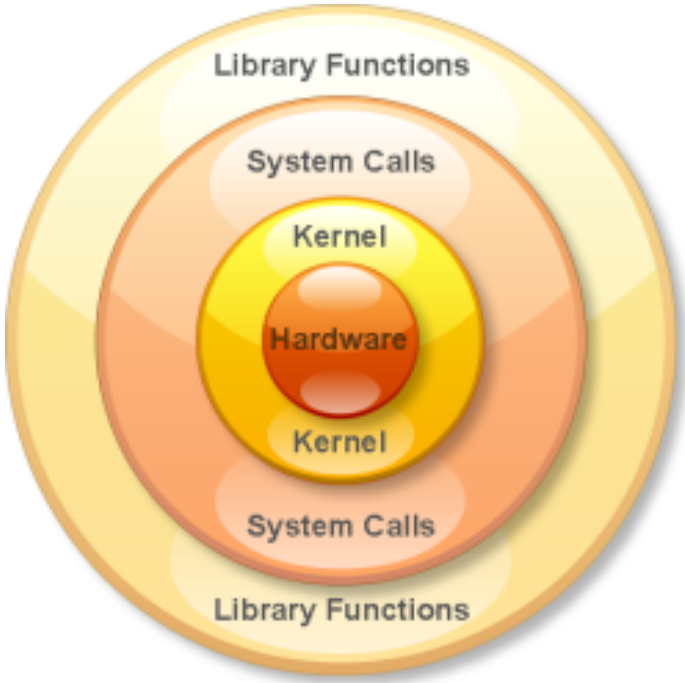
Challenges

- The kernel of an operating system is the central authority to enforce and control security
  - Large user base (e.g. 1.5 million Android devices activated per day, Google 2013); Critical bugs must be detected early enough!
- Ensure automation, reliability and quality assurance
  - Manual testing approaches (TRINITY fuzzer, Linux test project by IBM, Cisco, Fujitsu, OpenSuse, Red Hat) only; reduction of test suites w.r.t. revealing kernel bugs;
- Kernel testing far away from combinatorial modelling
  - Translation between combinatorics, software testing and software engineering;

ERIS: Combinatorial Kernel Testing

ERIS: Combinatorial Kernel Testing

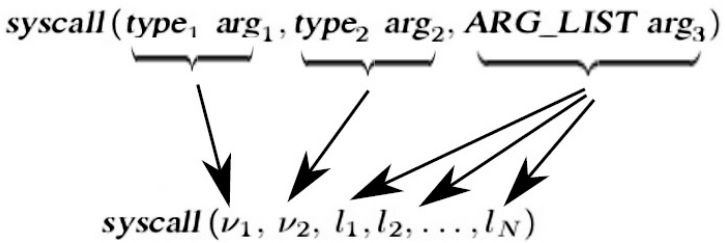
- **Focus:** Reliability and quality assurance of kernel software
- **Motivation:** Kernel is the central authority to ensure security
- **SUTs:** System calls of every git-commit of any (variant of) Linux
- **Evaluation**: Various kernel crashes for RCs and distribution kernels



Combinatorial API Testing

Testing APIs Function Calls

- **Modeling:** Combinatorial models:
  - IPM via equivalence- and category partitioning
  - IPM via novel flattening methodology

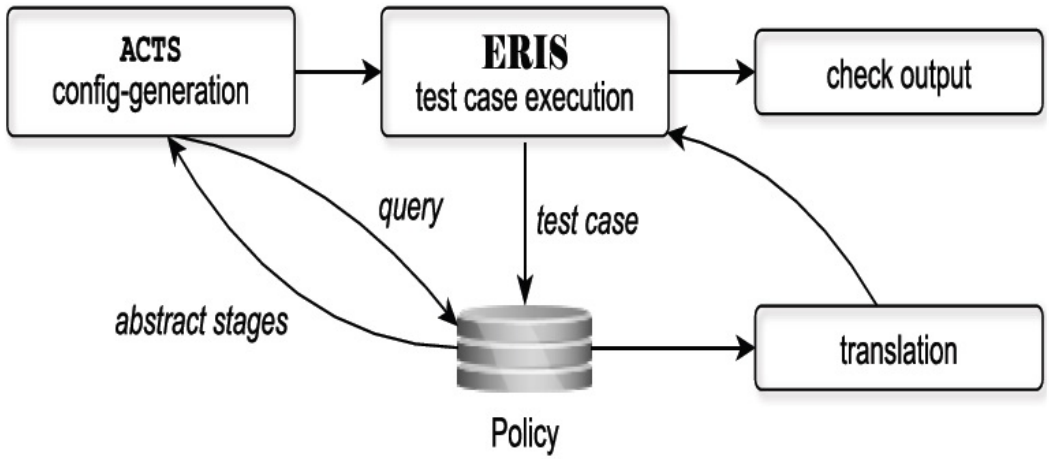


Abstr. Parameter	Parameter values
ARG_CPU	1, 2, 3, 4, ..., 8
ARG_MODE_T	1, 2, 3, 4, ..., 4095, 4096
ARG_PID	-3, -1, $pid_cron, $pid_w3m, 999999999
ARG_ADDRESS	null, $kernel_address, $page_zeros, $page_0xff, $page_allocs, ...
ARG_FD	fd_1, fd_2, fd_3, ..., fd_{15}
ARG_PATHNAME	$pathname_1$, $pathname_2$, $pathname_3$, ..., $pathname_{15}$

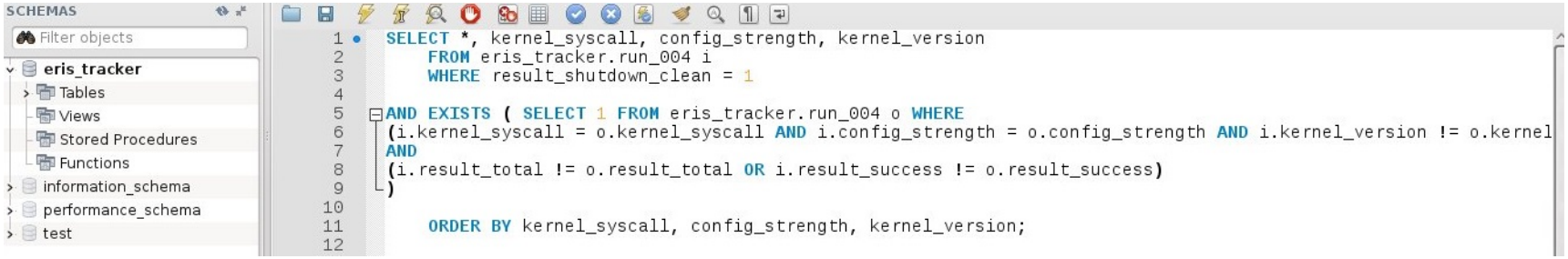
Automated Test Execution Framework

Some Features

- **Ease of use:** Only high-level parameters needed, everything else handled by the system
- **Test generation:** Your favorite CT generation tool
- **Test-runs:** Each invocation runs in a dedicated virtual machine
- **Logging:** Extensive information is captured
  - Adjustable to user demands / needs
- **Database:** Allows sophisticated post-processing queries



Sample Query and Results



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ılt_shutdown_clean	result_kill	result_segfault	result_total	result_success	result_reject	expected_total	kernel_syscall	config_strength	kernel_version
0	0	98	28	70	98		sched_getparam	2	v3.18
0	0	98	28	70	98		sched_getparam	2	v3.19
0	0	98	28	70	98		sched_getparam	2	v3.19-rc1
0	0	98	28	70	98		sched_getparam	2	v3.19-rc2
0	0	98	28	70	98		sched_getparam	2	v3.19-rc3
0	0	98	28	70	98		sched_getparam	2	v3.19-rc4
0	0	98	28	70	98		sched_getparam	2	v3.19-rc5
0	0	97	28	69	98		sched_getparam	2	v3.19-rc6
0	0	98	28	70	98		sched_getparam	2	v3.19-rc7
0	0	196	45	151	196		settimeofday	2	v3.18
0	0	196	45	151	196		settimeofday	2	v3.19
0	0	196	54	142	196		settimeofday	2	v3.19-rc1
0	0	196	45	151	196		settimeofday	2	v3.19-rc2
0	0	196	45	151	196		settimeofday	2	v3.19-rc3
0	0	196	45	151	196		settimeofday	2	v3.19-rc4

Milestones

Expertise at SBA Research

- Knowledge transfer of combinatorial designs $\Longrightarrow$ combinatorial testing
- Benefit from senior developers and software engineering experts

Milestones (IWCT/ICST2014, RTA2015)

- **Modelling:** System call arguments via IPM and categories
  - Automated t-way testing and translation layers
- **ERIS:** Highly configurable testing framework encompassing CT, execution environment, logging and database infrastructure
- **Experience Reports:** Real-world testing
  - Multiple kernel versions
  - Reproducibility of interesting test vectors (compared to other manual-testing approaches)
- **Evaluation:** Various kernel failures depending on crash oracles; two system calls flagged for detailed analysis;

Achievements

Our Contributions

- Proven applicability of combinatorial testing to
  - Web security testing
  - Operating system kernels
- Automation during the testing process
  - Greatly in demand in both academia and industry
  - Two prototype (combinatorial) testing frameworks
- Extensive experience reports for academia and real-world scenarios
  - Bridging the gap between discrete mathematics and information security through combinatorial testing
  - Security flaws are caused due to the interaction of a few parameters of the SUT

Critical Reflection

Established combinatorial security testing as a viable alternative to traditional testing methods (fuzzing, learning approaches, etc.)

Vision

Spoiler

- Efficient combinatorial security testing everywhere!
  - Deploy to all layers of Information Security
    - Application Security, Network Security, Systems Security
  - Advancement of the theory of combinatorial testing is needed
- Standardize the research methodologies

Long-term Plan and Objectives

- Combinatorial security testing will go mainstream in 2016-2021
- Optimization and automation of security tests
  - Interplay between basic and applied research
  - New application domains
  - Feedback cycle to secure software engineering

Next Steps in Web Security Testing

- Automated testing for real world applications
  - Access to large-scale test servers and automated setup environments is needed!
- Prioritization of XSS attack vectors; Guided combinatorial testing
- Wider study of how CT algorithms affect XSS (report)
- Release of XSSInjector to the research community

Future Directions

- Modelling SQL Injection attacks
- Directly applicable to database and application security
- **Notorious Examples:** Microsoft SQL Server Databases (2009), Yahoo! stolen credentials $\approx 500k$ (2012), Russian hackers theft of 1.2 billion credentials (2014)

ERIS$_\infty$ for Combinatorial Kernel Testing

- Extend to Android APIs/SELinux targeting mobile security features
- OS Kernels $\Longrightarrow$ Systems security
  - Testing of security patches to ensure attack-free environments
  - Industrial Automation Control Systems (IACS) testing
  - Cyber-physical Systems (CPS) testing

ERIS$_\infty$ in a Nutshell

- Sequences of systems calls
- Continuous integration tests of kernel versions
- Web monitoring platform

Protocol Testing

- **Motivation:** Major security breaches recently; NIST is currently revising the RFCs (standards)
- **SUTs:** TLS, SSL, SSH (cf. Internet Protocol Suite)
- **Goal:** Quality assurance of protocol implementations

Protocol Interaction Testing

- **Aim:** Assure proper error handling
- Test protocol implementations for erroneous configurations that lead to security flaws; IPM for protocol parameters

Table: IPM for TLS Cipher Suite Registry

Value	KEX/Auth	Cipher	Mode	MAC	Standard
0x00,0x03	RSA_EXPORT	RC4_40	-	MD5	RFC4346/RFC6347
0x00,0x23	KRB5	3DES	EDE_CBC	MD5	RFC2712
0x00,0x37	DH/RSA	AES_256	CBC	SHA	RFC5246
...	...	...	...	...	...

Measurement Analysis for TLS Cipher Suites

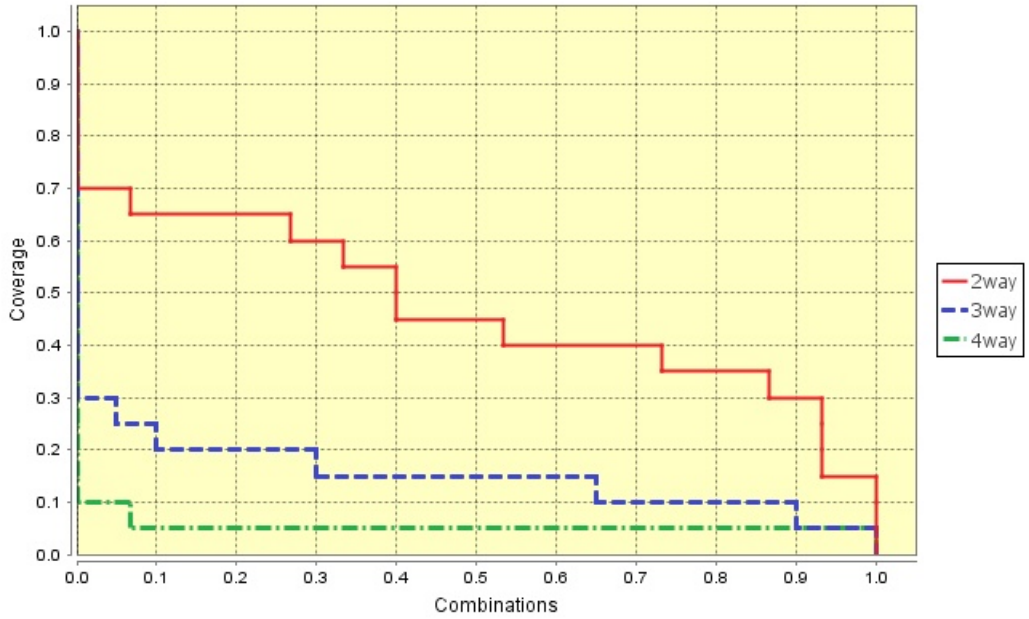


Figure: CCM analysis for the 317 cipher suite specifications of TLS.

Evaluation (Work in Progress)

- Coverage of t-way combinations extremely low
  - For $t \in \{2, 3, 4\} \Longrightarrow \text{cov} = 37.56\%, 9.39\%, 2.03\%$
- **Question:** Can this cause error handling problems?

Certificate Testing

- Standards for public key infrastructure (PKI)
- Attack vectors have the purpose to forge certificates
- **Oracle:** Test whether the server/client accepts them as valid

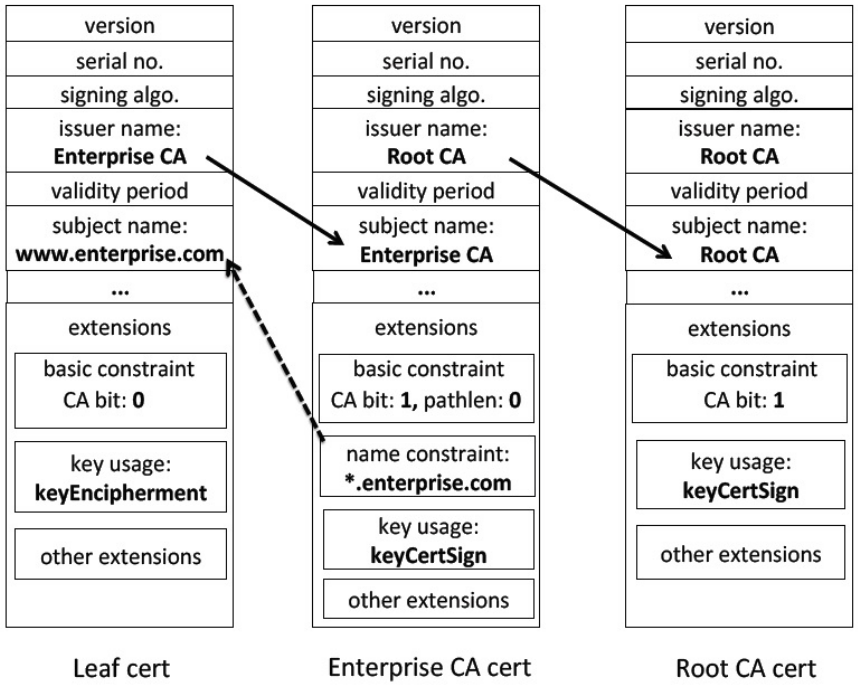
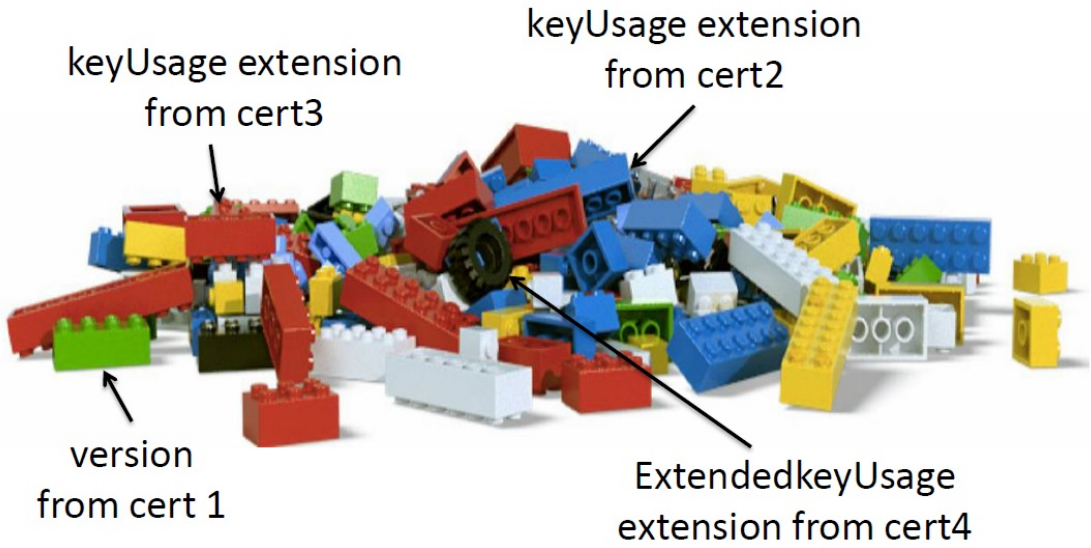


Figure: A sample X.509 certificate chain.

Frankencerts: How to Generate Test Certificates?

Random Selection of CA Parts (Brubaker et al., IEEE S&P 2014)

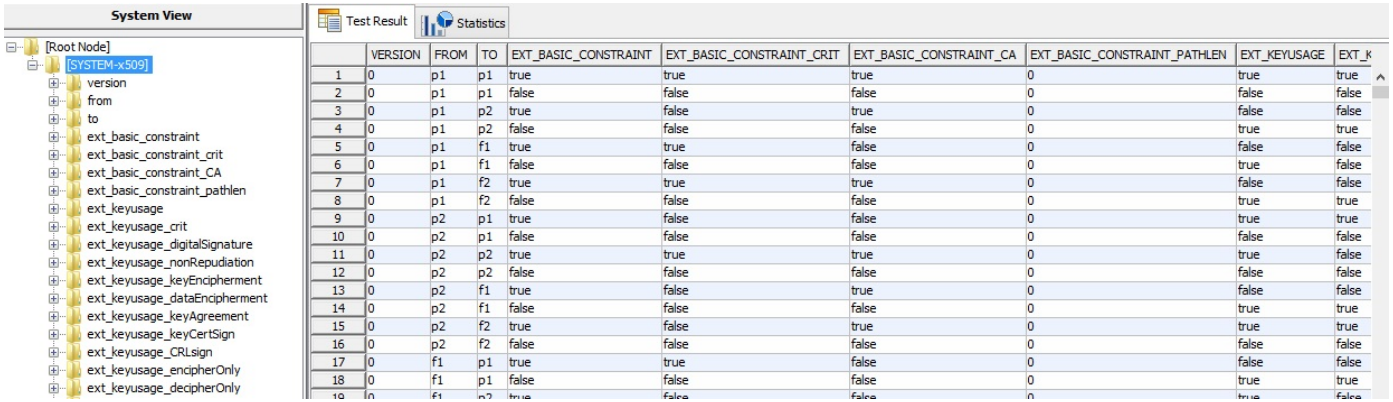
- Create X.509 certificates using randomly picked syntactically valid parts
  - How can we generate a large set of such syntactically valid pieces without reading X.509 specifications?
  - Scan the Internet for input certificates: 243,246 certs
  - Break them into parts and recombine them randomly: $\approx$ 8 millions
- Likely to violate some semantic constraints, e.g. will generate "bad" test certificates (as needed)



Can We Do Better?

CT Approach for Certificate Test Generation (Work in Progress)

- Reverse engineered an input model; generated certificates (up to $t = 5$)
- Use differential testing to check for discrepancies
  - Compared validation results of OpenSSL, GnuTLS, CYASSL, PolarSSL so far



	VERSION	FROM	TO	EXT_BASIC_CONSTRAINT	EXT_BASIC_CONSTRAINT_CRIT	EXT_BASIC_CONSTRAINT_CA	EXT_BASIC_CONSTRAINT_PATHLEN	EXT_KEYUSAGE	EXT_K
1	0	p1	p1	true	true	true	0	true	true
2	0	p1	p1	false	false	false	0	false	false
3	0	p1	p2	true	false	true	0	false	false
4	0	p1	p2	false	false	false	0	true	true
5	0	p1	f1	true	true	false	0	false	false
6	0	p1	f1	false	false	false	0	true	false
7	0	p1	f2	true	true	true	0	false	false
8	0	p1	f2	false	false	false	0	true	true
9	0	p2	p1	true	false	false	0	true	true
10	0	p2	p1	false	false	false	0	false	false
11	0	p2	p2	true	true	true	0	true	false
12	0	p2	p2	false	false	false	0	false	false
13	0	p2	f1	true	false	true	0	false	false
14	0	p2	f1	false	false	false	0	true	true
15	0	p2	f2	true	false	true	0	true	true
16	0	p2	f2	false	false	false	0	false	false
17	0	f1	p1	true	true	false	0	false	false
18	0	f1	p1	false	false	false	0	true	true

Method	Number of tests	Discrepancies
CT 2-way	23	3
CT 3-way	110	3
CT 4-way	470	5
CT 5-way	1670	6
Frankencerts	500,000	8

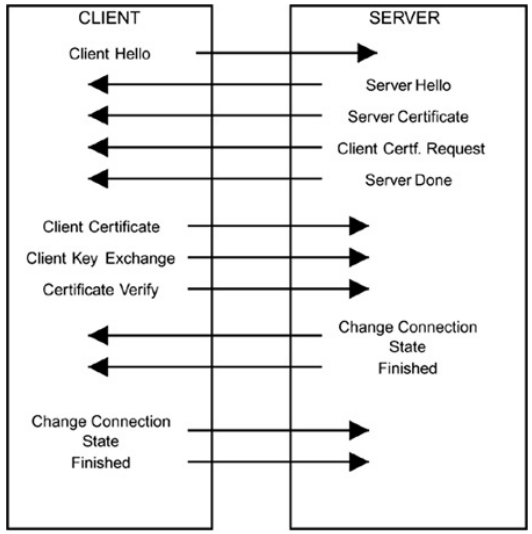


Figure: Handshake Message Sequence Diagram

Research Problem for t-way Sequence Testing?

Model the event sequence of TLS Handshake (conformance testing)

- Every t-way permutation of events $x_1, \ldots, x_p$ is covered by at least one test (events not necessarily adjacent)
- **Example:** six events a, b, c, d, e, f; (a, d, c, f, b, e) covers the 3-way sequence $(d, b, e) \Longrightarrow$(Client Hello, Server Hello, Finish)

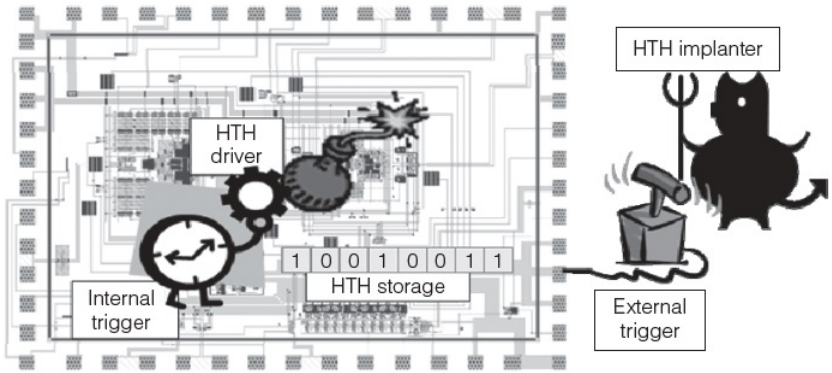
Hardware Malware

Hardware Malware

- **Scenario:** Trojans reside inside cryptographic circuits that perform encryption and decryption in FPGA technologies
  - **Examples:** Block ciphers (AES), Stream Ciphers (Mosquito)
- **Goal:** Hardware Trojan horse (HTH) detection

Instances of Hardware Trojan Horses

- **Combinational:** Activate when a specific combination of key bits appear
- **Sequential:** Activate after a counter has elapsed (time-bombs)



Hardware Malware

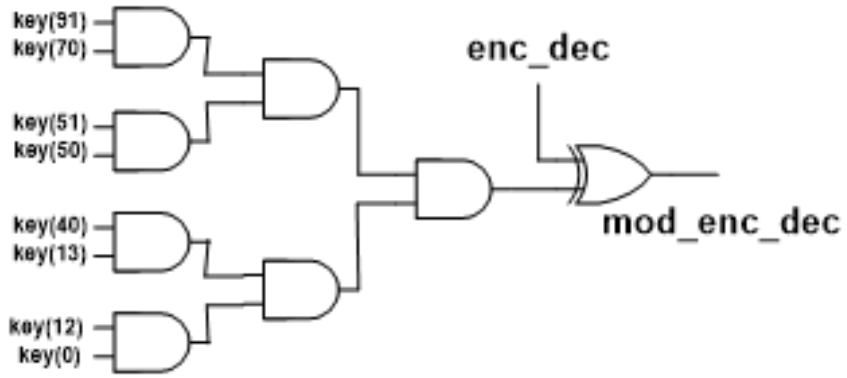


Figure: Design of a (Combinational) Hardware Trojan Horse

Hardware Trojan Horse Function

- The trigger part consists of 7 AND gates and monitors 8 key bits
- When at least one input is "0", the Trojan does nothing malicious
- When all inputs are "1", the Trojan payload part (just 1 XOR gate!) is activated
- The payload part reverses the mode of operation (encryption or decryption) $\Longrightarrow$ DoS attack until key is changed

Exciting (Triggering) Hardware Trojan Horses

Threat Model

- The attacker can control the key or the plaintext input and can observe the ciphertext output
- The attacker combines only a few signals for the activation

IPM for Ciphers

- **Triggering Sequence:** Trojan monitors $k << 128$ key bits of AES-128
- **Attack vectors:** Model triggering sequences of the Trojan (**black-box** testing); 128 binary parameters for AES-128
- **Input space:** $2^{128} = 3.4 \times 10^{38}$ for 128 bits key

Test Execution

- **Hardware implementation:** Verilog-HDL model with the Sakura-G FPGA board
- **Oracle:** Compare the output with a Trojan-free design of AES-128 (e.g. software implementation)

Hardware Malware Detection

Table: Comparison of test suite sizes using the constant weight vectors (CWV) procedure and the CA generation methods.

n	t	other	CWV	ours
128	2	2^7	129	11
128	3	-	256	37
128	4	2^{13}	8, 256	112
128	5	-	16, 256	252
128	6	-	349, 504	720
128	7	-	682, 752	2, 462
128	8	2^{23}	11, 009, 376	17, 544

Table: Test suite strength (t) vs. Trojan length (k)

t	Suite size	Number of activations		
		$k = 2$	$k = 4$	$k = 8$
2	11	5	3	0
3	37	12	4	0
4	112	32	7	1
5	252	62	14	1
6	720	307	73	6
7	2462	615	153	10
8	17544	4246	1294	178

Finding a Key in the Haystack

Our Results (to appear in ISSRE2015)

- There are about 366 ***trillion*** possible combinations for the Trojan activation;
- The whole space is covered with less than 18 ***thousands*** vectors
- .. and these vectors activate the Trojan ***hundreds*** of times.

What about Sequential Trojans?

- Recall that they activate after a counter has elapsed
- **Good news:** Model triggering sequences with orthogonal arrays (every t-way combination covered exactly λ times)
- **Bad news:** Orthogonal arrays do not exist for all numbers of possible parameters (mainly in power of two)

Analysis of Security Vulnerabilities

Analysis of Test Suites

- Countless Common Vulnerabilities and Exposures (CVEs)
- Dedicated CVE database for security community

Can we Do Better?

- No notion of combinatorial coverage measurement has been applied
- Number of parameters of the SUT and exact parameter value configurations that trigger security vulnerabilities is (mostly) undetermined so far (e.g. Heartbleed bug, Hardware Trojans)

Measuring t-wise Coverage for Combinatorial Security Testing

- **Requirements:** CCM, classifiers, feature model extraction
- Analogue to NIST study for NASA spacecrafts, medical devices
- **Goal:** Automation, reduction, fault-localization as proactive defenses

(Generic) Research Problems

Related to Security Protocols

- Recently Post-quantum variants of security protocols have emerged (e.g. Lattice-based crypto for TLS)
- Testing of their implementations (weakest link)?
- Fault location analysis (FLA)

Related to Hardware Malware Detection

- Generate (optimal) test suites for sequential Trojans
- Identification of key bit locations (FLA)

Other Application Domains for Combinatorial Security Testing

Internet of Things (IoT)? Your ideas?

Summary

Highlights

1. Proven applicability of combinatorial testing to:
   - security testing of web applications
   - quality assurance of kernel software
2. Extend it to new promising application domains for
   - assuring proper error handing of security protocols
   - ensuring Hardware malware detection
3. Vision for combinatorial security testing

Future Work

Solve (some) of the open research problems!

References

L. Yu, Y. Lei, R. Kacker, and D. Kuhn, “ACTS: A combinatorial test generation tool,” in Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on, 2013, pp. 370–375.

P. Kitsos, D. E. Simos, J. Torres-Jimenez and A. G. Voyiatzis, “Exciting FPGA Cryptographic Trojans using Combinatorial Testing”, to appear in the 26th IEEE International Symposium on Software Reliability Engineering (ISSRE 2015).

J. Bozic, B. Garn, I. Kapsalis, D. E. Simos, S. Winkler, and F. Wotawa, “Attack pattern-based combinatorial testing with constraints for web security testing,” 2015, The 2015 IEEE International Conference on Software Quality, Reliability and Security (QRS).

C. Brubaker, S. Jana, B. Ray, S. Khurshid und V. Shmatikov, “Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations,” IEEE Symposium on Security and Privacy, pp. 114-129, 2014.

B. Garn, I. Kapsalis, D. E. Simos, and S. Winkler, “On the applicability of combinatorial testing to web application security testing: A case study,” in Proceedings of the 2nd International Workshop on Joining AcadeMiA and Industry Contributions to Testing Automation (JAMAICA’14). ACM, 2014.

J. Bozic, B. Garn, D. E. Simos, and F. Wotawa, “Evaluation of the IPO-Family algorithms for test case generation in web security testing," in Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on, vopp.1-10, 2015

B. Garn and D. E. Simos, “Eris: A Tool for Combinatorial Testing of the Linux System Call Interface," In Proceedings of the 2014 IEEE International Conference on Software Testing, Verification, and Validation Workshops (ICSTW ’14). IEEE Computer Society, Washington, DC, USA, 58-67.

Questions - Comments

Thank you for your Attention!

dsimos@sba-research.org