

Web Application Security

Edited by

Lieven Desmet¹, Martin Johns², Benjamin Livshits³, and
Andrei Sabelfeld⁴

1 KU Leuven, BE, Lieven.Desmet@cs.kuleuven.be

2 SAP Research CEC – Karlsruhe, DE, mj@martinjohns.com

3 Microsoft Research – Redmond, US, livshits@microsoft.com

4 Chalmers UT – Göteborg, SE, andrei@chalmers.se

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 12401 “Web Application Security”. The seminar brought 44 web security researchers together, coming from companies and research institutions across Europe and the US.

The seminar had a well-filled program, with 3 keynotes, 28 research talks, and 15 5-minute talks. As web application security is a broad research domain, a diverse set of recent research results was presented during the talks, covering the web security vulnerability landscape, information-flow control, JavaScript formalization, JavaScript confinement, and infrastructure and server hardening. In addition to the plenary program, the seminar also featured three parallel break-out sessions on Cross-Site Scripting (XSS), JavaScript and Information-flow control.

Seminar 30. September – 05. October, 2012 – www.dagstuhl.de/12401

1998 ACM Subject Classification H.3.5 On-line Information Services – Web-based services, K.6.5 Security and Protection, D.4.6 Security and Protection

Keywords and phrases Web application security, JavaScript, Secure interaction, Information flow, Secure composition, Application security, Web 2.0

Digital Object Identifier 10.4230/DagRep.2.10.1

1 Executive Summary

Lieven Desmet

Martin Johns

Benjamin Livshits

Andrei Sabelfeld

The Dagstuhl seminar on *Web Application Security* aimed to bring researchers together in the field of web security, both from academia and industry. The seminar is a follow-up of the Dagstuhl Seminar 09141 on Web Application Security in 2009 [10, 9].

Research context

Since its birth in 1990, the web has evolved from a simple, stateless delivery mechanism for static hypertext documents to a fully-fledged run-time environment for distributed multi-party applications. Recently, the web technologies have gradually shifted from a central server technology towards a rich/stateful client paradigm and lively interaction models. The wave of popular peer-to-peer web applications and web mashup applications confirm this emerging trend. But the shift from the server-centered paradigm poses a significant



Except where otherwise noted, content of this report is licensed under a Creative Commons BY-NC-ND 3.0 Unported license

Web Application Security, *Dagstuhl Reports*, Vol. 2, Issue 10, pp. 1–37

Editors: Lieven Desmet, Martin Johns, Benjamin Livshits, and Andrei Sabelfeld



DAGSTUHL
REPORTS

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

challenge of securing web applications in the presence of multiple stakeholders, including security-ignorant end-users. This motivates the need for solid *web application security*.

The seminar aimed to address the open question of how to protect against the pervasive threats to web applications. Some of the key objectives put forward are (i) over-viewing the state of the art to consolidate and structure it, (ii) identifying key challenges, and (iii) brainstorming on new ideas and approaches towards resolving these challenges.

The inception of this Dagstuhl seminar was strongly inspired by the following emerging trends and challenges in the web security landscape:

Fine-grained access control. Fine-grained access control policies define how the application authenticates and authorizes end users, from which application contexts the application can be consulted, and which interaction sequences maintain the application's integrity (i.e. control-flow integrity). Our objective was to address a range of questions from formal foundation of authentication policies and protocols to the practicalities of authentication such as secure session management.

Information-flow control. Information-flow control specifies how sensitive data, possibly originating from multiple content providers in multiple trust domains, can be used in data aggregations, and client-side and server-side processing as is typically done in mashups. Challenges here include reconciling information-flow policies from several involved parties, with possibly conflicting goals. Moreover, tracking end-to-end information flow in web applications remains an open question. Our objective was to establish an enhanced understanding of how to make information-flow control policies and mechanisms practical in a web setting.

Secure composition. Secure composition policies specify how active third-party components, for instance written in JavaScript, can be securely integrated into applications via client-side and server-side mashups. By nature, web mashups heavily depend on interaction and communication across different origins, but contradictory, mashup security relies on separation techniques for protecting both code and data. As a result, traditional HTML techniques (mainly based on the same-origin policies) fail to address both the interaction and separation needs. We wanted to explore principled approaches to achieve the delicate balance between interaction and separation in security composition.

Cross-domain interaction. One of the original and still unresolved problems of the web is the inherent incompatibility between the cross-domain nature of the hyperlink and the same-origin security policy of its active content. In the recent past the situation has become even more complex with the introduction of client-side primitives for cross-domain interaction, such as CORS. Our objective was to assess the impact of current developments and identify promising directions for solutions.

Recent advances in JavaScript and HTML5. There are several technological advances in the latest versions of JavaScript (such as strict mode, frozen objects, proxies and SES), that might contribute to the security of web applications. In addition, the research community did make important steps forward in understanding and improving the language by formalizing its semantics. At the same time, web specification (including HTML5 and CSP) are adding tons of new features as well as security measures as part of the browsing environment. Our objective was to have an enhanced understanding of the latest trends and research advances in JavaScript and HTML5 with respect to security.

Seminar program

The seminar attracted 44 participants, coming from companies and research institutions across Europe and the US. The group represented a nice mix of participants of academia and industry (including researchers of Siemens, SAP, Trend Micro, and Microsoft as well as two banks) and a good balance between junior and senior researchers.

The seminar had a well-filled program, with 3 keynotes, 28 research talks, 3 break-out sessions and 15 5-minute talks. The organizers aimed at keeping enough time during the breaks and in the evening for informal discussion. In addition, the participants went on a hike to the lake on Wednesday afternoon, as part of the social program.

Keynotes

The first three days, the floor was opened by keynotes to set the scene and inspire the discussions. The organizers invited the following three keynote speakers and the keynote abstracts can be found in section 3.

- Martin Johns (SAP Research – Karlsruhe, DE) – *Web Application Security: Are we there yet?*
- Shriram Krishnamurthi (Brown University – Providence, US) – *Browser Extension Analysis and Other JavaScript Adventures*
- John C. Mitchell (Stanford University, US) – *Science of Web Security and third-party tracking*

Martin Johns opened the Dagstuhl seminar on Monday by assessing the current state-of-practice in web security, 3 years after the previous Dagstuhl Seminar on Web application security. He sketched the evolving web landscape, and surveyed to what extent the results achieved so far suffice, and what is still missing. In particular, Martin gave a heads-up on client-side complexity and server-driven security, as being developed in the EU-FP7 project WebSand¹.

Shriram Krishnamurthi discusses techniques based on typing to verify web applications, and demonstrated how these techniques can also be used to verify browser extensions. Such a verification can for instance assure that no unsafe functions are called within an extension, while operating in *private browsing mode*. As part of the underlying toolkit, Shriram presented core semantics of JavaScript in λ_{JS} (lambdaJS), and showed how a JavaScript program can be desugared in λ_{JS} [30].

John C. Mitchell focused on the *science of security and principles*, and demonstrated this by means of relevant web security examples. He emphasized the importance of defining system models, adversary models as well as desired properties of system, and argued that it seems feasible to verify web security properties. An interesting research question to be answered by such a *scientific* approach would be “Does CSP prevent XSS?”, and John challenged the audience to tackle this challenge. In addition, he discussed the importance of experimental studies and gave some highlights on recent research results on web tracking.

Research talks

The organizers invited all the participants to take the floor during the seminar, and encouraged the presenters to step away from typical conference presentations, but rather strive for

¹ EU-FP7 STREP WebSand, <https://www.websand.eu/>

interaction with the audience and engage discussions.

Web security is a broad research domain, and the seminar was able to attract web security researchers with various backgrounds. As a result, a diverse set of recent research results was presented during the seminar, and these can be grouped in 5 topical clusters:

1. Web security vulnerability landscape
2. Information-flow control
3. JavaScript formalization
4. JavaScript confinement
5. Infrastructure and server hardening

For each of the clusters, the list of talks is enumerated in this section. For more detailed information about each of the talks, we refer to the talk abstracts in section 4.

Cluster 1: Web security vulnerability landscape

This first cluster of talks discussed the evolving landscape of web vulnerabilities and presented some novel attack vectors. In addition, some of the talks gave some more insights in setting up large-scale security assessments.

- John Wilander gave us some fruitful insights in the banking domain by highlighting the security pains of an online bank.
- Boris Hemkemeier discussed the state-of-practice in authentication and authorization techniques used in online banks.
- Fabio Massacci illustrated with data from anti-virus vendors the mismatch between vulnerabilities studied by security researchers, and vulnerabilities exploited by bad guys in the wild.
- Nick Nikiforakis gave us insights on the risk of third-party scripts in web applications, based on a large-scale evaluation of remote script inclusions [52].
- Steven Van Acker gave us a view behind the scene in setting up large-scale web security experiments on the basis of the FlashOver research [65].
- Mario Heiderich discussed a novel set of scriptless injection attacks via Cascading Style Sheets (CSS), HTML, SVG and font files [34].

Cluster 2: Information-flow control

Secondly, a set of novel enforcement mechanisms have been presented for information-flow control for JavaScript, as well as quantitative information-flow policies.

- Frank Piessens presented FlowFox, a fully functional web browser that implements the Secure Multi-Execution (SME) technique on top of Firefox [21].
- Cormac Flanagan discussed the the Facets mechanism to simultaneously and efficiently simulate multiple executions for different security levels [3].
- Nataliia Bielova discussed an information-flow analysis technique to quantify leakage by browser fingerprinting.
- Daniel Hedin presented a dynamic type system that guarantees information-flow security for a core subset of JavaScript [33].
- Arnar Birgisson showed how to overcome the permissiveness limit for dynamic information-flow analysis by a novel use of testing [8].
- Michael Hicks introduced knowledge-based security for collaborative web applications [47].
- Martin Ochoa presented a quantification approach for cache side channels [41].

- Sebastian Schinzel discussed timing-based [59] and storage-based [27] side channel attacks for the web.

Cluster 3: JavaScript formalization

Thirdly, the research community did progress quite substantially in formalizing the semantics of JavaScript, in verifying JavaScript code, and using JavaScript as an assembly language.

- Shriram Krishnamurthi discussed how to verify browser extensions written in JavaScript.
- Ravi Chugh presented dependent types for a large subset of JavaScript [17].
- Nikhil Swamy presented recent work on F^* , to generate JavaScript from cleaner semantics [26].
- Joe Gibbs Politz presented semantics for Getters, Setters and Eval in JavaScript [53].
- Arjun Guha presented a core calculus for scripting languages with support of Setters to avoid run-time errors.

Cluster 4: JavaScript confinement

Fourthly, several techniques have been presented to confine the execution of JavaScript code, and to detect attacks at run-time.

- Akhawe Devdatta presented a privilege separation approach for HTML5 applications, and applied the technique to browser extensions [2].
- Thorsten Holz and Mario Heiderich introduced IceShield, a JavaScript based tool that enables dynamic code analysis on websites, and JSAgents, a client-side framework to detect and mitigate live attacks against web pages and browsers, based on anomaly detection on the DOM.
- Lieven Desmet presented the JSand approach to sandbox third-party JavaScript within the existing browser infrastructure [1].
- Michael Franz proposed compiler-driven diversity to run diversified programs in parallel to detect attacks at run-time.

Cluster 5: Infrastructure and server hardening

Finally, a set of infrastructure and server hardening techniques has been proposed to increase the end-to-end security of the web applications.

- Cédric Fournet presented recent work towards the formal certification of the TLS protocol [6].
- Juraj Somorovsky presented a practical attack on XML Encryption, which allows to decrypt a ciphertext by sending related ciphertexts to a Web Service and evaluating the server response [38, 37].
- John Wilander did put forward some initial ideas to achieve a stateless anti-CSRF mechanism, while ensuring the same level of security.
- Bastian Braun presented server-side techniques to achieve Control-Flow Integrity in web applications [12].
- Sergio Maffei formalized several configurations of the OAuth 2.0 protocol and verified these with ProVerif [4].

Break-out sessions

To complement the keynotes and the research talks, the organizers opted to have three parallel break-out sessions as part of the seminar program. The break-out sessions enabled participants to discuss selected topics in web security research in an informal setting and in smaller teams. The three topics of the break-out sessions were:

- Cross-Site Scripting (XSS)
- JavaScript
- Information-Flow

The main purpose of the break-out sessions was to informally discuss the most important state-of-the-art and research challenges. As part of the break-out sessions, the teams identified and enlisted in a bottom-up way the most relevant state-of-the-art work, as well as the set of main challenges and research directions for the specific web security research area. The break-out sessions consisted of three slots of 70 minutes on Monday, Tuesday and Thursday. Participants joined the break-out sessions of their choice on Monday and Tuesday, and were encouraged to take part of two different sessions. The session on Thursday was used to report back the results of the three break-out sessions to the full group by means of a small presentation. The reports of the three break-out sessions are summarized in section 5.

5-minute talks

Finally, to encourage participants to pitch new research ideas, or highlight some relevant results, we also had two sessions specifically targeted at **5-minute talks**. The list of speakers and their topics looks as follows:

- Thomas Jensen: Certified analysis of JavaScript
- Daniele Filaretti: JsCert
- Andrei Sabelfeld: GlassTube
- Devdatta Akhawe: Dusting The Web
- Michael Hicks: Build it, break it
- Sebastian Schinzel: Remote fingerprinting of programming libraries
- Joe Gibbs Politz: Finding bugs with type systems
- Cédric Fournet: ZQL: cryptographic compiler for data privacy
- Nikhil Swamy: TypeScript
- Boris Hemkemeier: Why to get rid of the SSL CA
- Valentin Dallmeier: WebMate – Exploring web 2.0 applications
- Mario Heiderich: JsAgents
- Joachim Posegga: Multi-party application platform
- Jorge Cuellar: Location privacy
- Egon Börger: Model web application frameworks

Conclusion

The Dagstuhl seminar on Web Application Security was a timely follow-up of the previous Dagstuhl seminar on this topic in 2009. The research domain has been maturing over the last five years, and new challenges have emerged such as the client-side complexity, the need of information-flow control enforcement, and hardening of JavaScript code.

The seminar brought 44 web security researchers together, coming from companies and research institutions across Europe and the US. The seminar had a well-filled program, with 3 keynotes, 28 research talks, and 15 5-minute talks. As web application security is a broad research domain, a diverse set of recent research results was presented during the talks, covering the web security vulnerability landscape, information-flow control, JavaScript formalization, JavaScript confinement, and infrastructure and server hardening.

In addition to the plenary program, the seminar also featured three parallel break-out sessions on Cross-Site Scripting (XSS), JavaScript and Information-flow control. The main goal of the break-out sessions was to informally discuss the most important state-of-the-art work, as well as to identify the main challenges and research directions for future research, as documented in this report.

Finally, the organizers of the Dagstuhl seminar have set up a *Special Issue on Web Application Security* as part of the *Journal of Computer Security*, specifically devoted to a selection of promising results presented at the seminar. Four participants have been invited to submit an extended paper of their talk to the special issue, and the manuscripts are currently under review.

2 Table of Contents

Executive Summary

<i>Lieven Desmet, Martin Johns, Benjamin Livshits, and Andrei Sabelfeld</i>	1
---	---

Keynote talks

Web Application Security: Are we there yet? <i>Martin Johns</i>	10
Browser Extension Analysis and Other JavaScript Adventures <i>Shriram Krishnamurthi</i>	10
Science of Web Security and third-party tracking <i>John C. Mitchell</i>	10

Overview of Talks


Quantifying the Leakage of Browser Fingerprints by Quantitative Information Flow Analysis <i>Nataliia Bielova</i>	12
Boosting the Permissiveness of Dynamic Information-Flow Tracking by Testing <i>Arnar Birgisson</i>	12
Control-Flow Integrity in Web Applications <i>Bastian Braun</i>	13
Dependent Types for JavaScript <i>Ravi Chugh</i>	13
Contribute your Location Privacy Solution <i>Jorge Cuéllar</i>	14
JSand: Server-driven Sandboxing of Third-party JavaScript <i>Lieven Desmet</i>	14
Privilege Separation for HTML5 Applications <i>Akhawe Devdatta</i>	15
Towards Certified Verification for Web Programming <i>Daniele Filaretti</i>	15
Multiple Facets for Dynamic Information Flow <i>Cormac Flanagan</i>	15
Software Immunity via Large-Scale Diversification <i>Michael Franz</i>	16
First Class Field Names <i>Arjun Guha</i>	16
JSFlow/SnowFox – Implementation of a Dynamic Information Flow Monitor for JavaScript <i>Daniel Hedin</i>	17
Scriptless Attacks: Stealing the Pie Without Touching the Sill <i>Mario Heiderich</i>	17

JSAgents	
<i>Mario Heiderich</i>	18
Toward decentralized collaborative webapps via knowledge-based security	
<i>Michael Hicks</i>	18
IceShield: Detection and Mitigation of Malicious Websites with a Frozen DOM	
<i>Thorsten Holz</i>	19
Discovering Concrete Attacks on Website Authorization by Formal Analysis	
<i>Sergio Maffei</i>	19
My software has a vulnerability should i worry?	
<i>Fabio Massacci</i>	20
You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions	
<i>Nick Nikiforakis</i>	20
Automatic Quantification of CPU Cache Side-channels	
<i>Martin Ochoa</i>	21
FlowFox: an experiment on bringing information flow control to the browser	
<i>Frank Piessens</i>	21
A Tested Semantics for Getters, Setters, and Eval in JavaScript	
<i>Joe Gibbs Politz</i>	22
Progressive Types	
<i>Joe Gibbs Politz</i>	22
GlassTube: A Lightweight Approach to Web Application Integrity	
<i>Andrei Sabelfeld</i>	22
Side channel attacks on the web	
<i>Sebastian Schinzel</i>	23
How To Break XML Encryption	
<i>Juraj Somorovsky</i>	23
Verifying JavaScript programs with the Dijkstra State Monad	
<i>Nikhil Swamy</i>	24
Fully Abstract Compilation to JavaScript	
<i>Nikhil Swamy</i>	24
behind FlashOver: Automated Discovery of Cross-site Scripting Vulnerabilities in Rich Internet Applications	
<i>Steven Van Acker</i>	25
The Security Pains of an Online Bank	
<i>John Wilander</i>	25
Stateless CSRF Protection	
<i>John Wilander</i>	26
Break-out sessions	
Break-out session: Cross-Site Scripting (XSS)	27
Break-out session: JavaScript	28
Break-out session: Information-Flow	29
Participants	37

3 Keynote talks

3.1 Web Application Security: Are we there yet?

Martin Johns (SAP Research – Karlsruhe, DE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Martin Johns


More than twenty years after the invention of the Web, twelve years after the first public report of the XSS vulnerability class, approximately 7 years since the academic world become interested in the topic, and 3 years after the previous Dagstuhl Seminar on Web application security, this talk tries to assess the current state of Web security:

- How has the landscape of real-life Web application changed in the recent past?
- How does this affect our previously achieved security results?
- Are we addressing the whole picture?
- How far have we come and what is still missing?

In the context of the talk recent advances and challenges in the field of server-driven Web security will be shown and the problem of ever growing client-side complexity will be discussed.

3.2 Browser Extension Analysis and Other JavaScript Adventures

Shriram Krishnamurthi (Brown University – Providence, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Shriram Krishnamurthi

Modern web browsers implement a *private browsing* mode that is intended to leave behind no traces of a user's browsing activity on their computer. This feature is in direct tension with support for *extensions*, which let users add third-party functionality into their browser. I will discuss the dimensions of this problem, present our approach to verifying extensions, and sketch our findings on several real, third-party extensions. I will then generalize this work to talk about other related problems, the toolkit we deploy to tackle them, and open issues for the community to consider.

3.3 Science of Web Security and third-party tracking

John C. Mitchell (Stanford University, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© John C. Mitchell


This talk focuses on the science of security and principles and examples relevant to web security. We have developed a scientific framework for security that is based on system models, adversary, and desired properties. In this approach, a system is secure if, for all actions of system users and system adversaries, the desired properties hold in spite of attack. Using this perspective, we propose a model of the web, three adversary models, and identify a set of security properties relevant to web security. This model supports automated analysis, which we have used to find and repair vulnerabilities in various mechanisms. With this

background, the second half of the talk presents an experimental web security study and some analysis of web privacy and third-party tracking. In the experimental study, we studied the result of approximately 20 web development teams and correlate security of their sites with features such as the programming language used, developed familiarity with security concepts, and whether the developers are freelancers or part of a startup team. Our web privacy study is based on a web measurement tool, called FourthParty, that uses an instrumented browser to capture events that occur with a site is visited and stores them in a SQL database. With this tool, we have identified sites and companies that violate their stated privacy policies or intentionally subvert privacy mechanisms.

4 Overview of Talks

4.1 Quantifying the Leakage of Browser Fingerprints by Quantitative Information Flow Analysis

Nataliia Bielova (INRIA – Rennes, FR)


License  Creative Commons BY-NC-ND 3.0 Unported license
© Nataliia Bielova

Joint work of Bielova, Nataliia; Jensen, Thomas; Besson, Frederic

Web tracking technologies allow the websites to create profiles about its users, but at the same time the users' privacy is breached. Tracking technologies can be separated into stateful and stateless. The first type stores the unique identifier in the user browser, and the second one creates an identifier on the fly. From the legal side, EU ePrivacy directive restricts the usage of stateful technologies, but stateless technologies are left outside of the scope of this law. We address the problem of stateless tracking called fingerprinting. It is based on the properties of the web browser and the OS (a fingerprint) and uses this information to distinguish between the different users visiting the site. We propose a definition of quantified interference that tells how much information about a browser fingerprint a tracker can learn by observing the public outputs of the program. We then present ongoing work on two quantitative information flow analyses: purely dynamic and hybrid. Both techniques safely over-approximate the amount of the information leakage that an attacker could learn from the browser fingerprint. The hybrid analysis is more precise than the dynamic one. Compared to purely static, quantitative information flow analysis, our analyses can give a more precise number for a leaked information because we use the concrete values of the user's browser fingerprint instead of calculating the information entropy for all possible fingerprints.

4.2 Boosting the Permissiveness of Dynamic Information-Flow Tracking by Testing

Arnar Birgisson (Chalmers UT – Göteborg, SE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Arnar Birgisson

Joint work of Birgisson, Arnar; Hedin, Daniel; Sabelfeld, Andrei

Main reference A. Birgisson, D. Hedin, A. Sabelfeld, "Boosting the Permissiveness of Dynamic Information-Flow Tracking by Testing," in Proc. of the European Symp. on Research in Computer Security (ESORICS 2012), Pisa, Italy, LNCS, Vol. 7459, Springer, 2012.

URL http://dx.doi.org/10.1007/978-3-642-33167-1_4

Tracking information flow in dynamic languages remains an open challenge. It might seem natural to address the challenge by runtime monitoring. However, there are well-known fundamental limits of dynamic flow-sensitive tracking of information flow, where paths *not* taken in a given execution contribute to information leaks. This paper shows how to overcome the permissiveness limit for dynamic analysis by a novel use of testing. We start with a program supervised by an information-flow monitor. The security of the execution is guaranteed by the monitor. Testing boosts the permissiveness of the monitor by discovering paths where the monitor raises security exceptions. Upon discovering a security error, the program is modified by injecting an annotation that prevents the same security exception on the next run of the program. The elegance of the approach is that it is sound no matter how much coverage is provided by the testing. Further, we show that when the mechanism has

discovered the necessary annotations, then we have an accuracy guarantee: the results of monitoring a program are at least as accurate as flow-sensitive static analysis. We illustrate our approach for a simple imperative language with records and exceptions. Our experiments with the QuickCheck tool indicate that random testing accurately discovers annotations for a collection of scenarios with rich information flows.

4.3 Control-Flow Integrity in Web Applications

Bastian Braun (Universität Passau, DE)

License © ⓘ ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Bastian Braun

Joint work of Braun, Bastian; Gemein, Patrick; Reiser, Hans P.; Posegga, Joachim

Main reference B. Braun, P. Gemein, H.P. Reiser, J. Posegga, “Control-Flow Integrity in Web Applications,” in Proc. of the Int’l Symp.on Engineering Secure Software and Systems (ESSoS 2013).

Modern web applications frequently implement complex control flows, which require the users to perform actions in a given order. Users interact with a web application by sending HTTP requests with parameters and in response receive web pages with hyperlinks that indicate the expected next actions. If a web application takes for granted that the user sends only those expected requests and parameters, malicious users can exploit this assumption by crafting unintended requests.

We analyze recent attacks on web applications with respect to user-defined requests and identify their root cause in the missing explicit control-flow definition and enforcement. Based on this result, we provide our approach, a control-flow monitor that is applicable to legacy as well as newly developed web applications. It expects a control-flow definition as input and provides guarantees to the web application concerning the sequence of incoming requests and carried parameters. It protects the web application against race condition exploits, a special case of control-flow integrity violation. Moreover, the control-flow monitor supports modern browser features like multi-tabbing and back button usage.

4.4 Dependent Types for JavaScript

Ravi Chugh (University of California – San Diego, US)

License © ⓘ ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Ravi Chugh

Joint work of Chugh, Ravi; Herman, David; Jhala, Ranjit


Main reference R. Chugh, D. Herman, R. Jhala, “Dependent types for JavaScript,” in Proc. of the ACM Int’l Conf. on Object oriented Programming Systems Languages and Applications (OOPSLA ’12), pp. 587–606. ACM, USA, 2012

URL <http://dx.doi.org/10.1145/2384616.2384659>

Static reason for JavaScript is hard. We describe recent progress towards building a statically typed dialect called Dependent JavaScript (DJS) that reasons about the mutable, prototype-based objects and arrays found in idiomatic JavaScript, and we show how to track security policies using DJS.

4.5 Contribute your Location Privacy Solution

Jorge Cuéllar (Siemens – München, DE)


License  Creative Commons BY-NC-ND 3.0 Unported license
© Jorge Cuéllar

After nine years of discussion, draft-ietf-geopriv-policy-27 is finally in RFC-EDITOR state. The result is that only quite trivial solutions have been standardized. But, on the other hand, a mechanisms has been created for adding algorithms, together with a context description of when they apply and a list of informal security properties (requirements) it has or implements. Suppose a user wants to share his location with a group of people, but not to be too precise about his location. The document defines an authorization policy language for controlling access to location information. This language can used by end-users in order to share their location with different friends, or other people, with chosen levels of uncertainty (say, "within 5km"). This may be seen as a declassification problem: What are good declassification functions (for different purposes)? There is no solution to all requirements, in all contexts. One solution could work well for moving in densely populated areas and provide good privacy properties (which ones?).

If you have a partial solution, submit your algorithms: we need them.

4.6 JSand: Server-driven Sandboxing of Third-party JavaScript

Lieven Desmet (KU Leuven, BE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Lieven Desmet

Joint work of Agten, Pieter; Van Acker, Steven; Brondsema, Yoran; Phung, Phu H.; Desmet, Lieven; Piessens, Frank

Main reference P. Agten, S. Van Acker, Y. Brondsema, P.H. Phung, L. Desmet, F. Piessens, "JSand: Complete client-side sandboxing of third-party JavaScript without browser modifications," Annual Computer Security Applications Conference (ACSAC 2012), Orlando, Florida, USA, 3-7 December 2012.

URL <http://dx.doi.org/10.1145/2420950.2420952>

The inclusion of third-party scripts in web pages is a common practice, but such script inclusions carry risks, as the included scripts operate with the privileges of the including website.

In this talk, I briefly present JSand, a server-driven but client-side JavaScript sandboxing framework. JSand requires no browser modifications: the sandboxing framework is implemented in JavaScript and is delivered to the browser by the websites that use it. Enforcement is done entirely at the client side.

Furthermore, I discuss some of the challenges and open problems to enforce security policies purely at the client side.

4.7 Privilege Separation for HTML5 Applications

Akhawe Devdatta (University of California – Berkeley, US)

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Akhawe Devdatta

Joint work of Devdatta, Akhawe; Saxena, Prateek; Song, Dawn

Main reference D. Akhawe, P. Saxena, D. Song, “Privilege separation in HTML5 applications,” in Proc. of the 21st USENIX Security Symposium (Security’12). USENIX Association, Berkeley, CA, USA, 16 pp., 2012.

URL <https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final168.pdf>

The standard approach for privilege separation in web applications is to execute application components in different web origins. This limits the practicality of privilege separation since each web origin has financial and administrative cost. In this paper, we propose a new design for achieving effective privilege separation in HTML5 applications that shows how applications can cheaply create arbitrary number of components. Our approach utilizes standardized abstractions already implemented in modern browsers. We do not advocate any changes to the underlying browser or require learning new high-level languages, which contrasts prior approaches. We empirically show that we can retrofit our design to real-world HTML5 applications (browser extensions and rich client-side applications) and achieve reduction of 6x to 10000x in TCB for our case studies. Our mechanism requires less than 13 lines of application-specific code changes and considerably improves auditability.

4.8 Towards Certified Verification for Web Programming

Daniele Filaretti (Imperial College London, GB)

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Daniele Filaretti

A brief overview of the projects I’ve been working on during the first year of my PhD at Imperial College London.

4.9 Multiple Facets for Dynamic Information Flow

Cormac Flanagan (University of California – Santa Cruz, US)

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Cormac Flanagan

Joint work of Austin, Tom; Flanagan, Cormac

Main reference T.H. Austin, C. Flanagan, “Multiple facets for dynamic information flow,” in Proc. of the 39th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL’12). pp. 165-178, ACM, USA, 2012.


URL <http://dx.doi.org/10.1145/2103621.2103677>

JavaScript has become a central technology of the web, but it is also the source of many security problems, including cross-site scripting attacks and malicious advertising code. Central to these problems is the fact that code from untrusted sources runs with full privileges. We implement information flow controls in Firefox to help prevent violations of data confidentiality and integrity. Most previous information flow techniques have primarily relied on either static type systems, which are a poor fit for JavaScript, or on dynamic analyses that sometimes get stuck due to problematic implicit flows, even in situations where the target web application correctly satisfies the desired security policy. We introduce faceted

values, a new mechanism for providing information flow security in a dynamic manner that overcomes these limitations. Taking inspiration from secure multi-execution, we use faceted values to simultaneously and efficiently simulate multiple executions for different security levels, thus providing non-interference with minimal overhead, and without the reliance on the stuck executions of prior dynamic approaches.

4.10 Software Immunity via Large-Scale Diversification


Michael Franz (Univ. California – Irvine, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Michael Franz

We have been investigating compiler-generated software diversity as a defense mechanism against software attacks. Imagine an “App Store” containing a diversification engine (a *multicompiler*) that automatically generates a unique version of every program for every user. All the different versions of the same program behave in exactly the same way from the perspective of the end-user, but they implement their functionality in subtly different ways. As a result, any specific attack will succeed only on a small fraction of targets. An attacker would require a large number of different attacks and would have no way of knowing a priori which specific attack will succeed on which specific target. Equally importantly, this approach makes it much more difficult for an attacker to generate attack vectors by way of reverse engineering of security patches. We have built such a multicompiler which is now available as a prototype. We can diversify large software distributions such as the Chromium web browser or a complete Linux distribution. In this talk, I present some preliminary benchmarks and also address some practical issues such as the problem of reporting errors when every binary is unique, and updating of diversified software.

4.11 First Class Field Names

Arjun Guha (Cornell University – Ithaca, US)


License  Creative Commons BY-NC-ND 3.0 Unported license
© Arjun Guha

Joint work of Politz, Joe Gibbs; Guha, Arjun; Krishnamurthi, Shriram
Main reference J. Gibbs Politz, A. Guha, S. Krishnamurthi, “Semantics and Types for Objects with First-Class Member Names.”
URL <http://www.cs.brown.edu/~sk/Publications/Papers/Published/pgk-sem-type-fc-member-name/>

Objects in many programming languages are indexed by first-class strings, not just first-order names. We define λ_{ob}^S “LambdaSOB”, an object calculus for such languages. We then develop a type system for LambdaSOB that is built around string pattern types, which describe (possibly infinite) collections of members. We define subtyping over such types, extend them to handle inheritance, and discuss the relationship between the two. We enrich the type system to recognize tests for whether members are present, and briefly discuss exposed inheritance chains. The resulting language permits the ascription of meaningful types to programs that exploit first-class member names for object-relational mapping, sandboxing, dictionaries, We prove that well-typed programs never signal member-not-found errors, even when they use reflection and first-class member names. We briefly discuss the implementation of these types in a prototype type-checker for JavaScript

4.12 JSFlow/SnowFox – Implementation of a Dynamic Information Flow Monitor for JavaScript

Daniel Hedin (Chalmers UT – Göteborg, SE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Daniel Hedin

Joint work of Hedin, Daniel; Sabelfeld, Andrei; Birgisson, Arnar; Bello, Luciano


Main reference D. Hedin, A. Sabelfeld, “Information-Flow Security for a Core of JavaScript,” in Proc. of the IEEE Computer Security Foundations Symposium, Harvard University, Cambridge MA, June 25–27, IEEE CCS Press, 2012.

URL <http://www.cse.chalmers.se/~andrei/jsflow-csf12.pdf>

Web applications and services are rapidly growing more sophisticated and feature rich. This is achieved by including script libraries from different sources; even a standard news paper includes tens of scripts, ranging from libraries for user statistics like Google Analytics, to libraries for ad-provision like DoubleClick, utility like jQuery, or functionality like Tynt. Due to inadequacies in script inclusion, scripts are typically included under full trust. This means that the scripts are able to access any information on the page, including any information the user types into various forms. This raises a number of security concerns. In particular how can we guarantee that the included scripts do not harvest the page for sensitive information? In “Information-Flow Security for a Core of JavaScript” we have previously suggested how to solve this problem using dynamic information flow tracking for a core of JavaScript. In this talk we present JSFlow, an implementation of an information flow monitor for full JavaScript based on non-strict part of Ecma-262 v5. We present an overview of the implementation and show how the monitor is able to stop insecure information flow in actual web pages by viewing them in Firefox using Snowfox, a Firefox extension that replaces the JavaScript engine of Firefox with JSFlow.

4.13 Scriptless Attacks: Stealing the Pie Without Touching the Sill

Mario Heiderich (Ruhr-Universität Bochum, DE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Mario Heiderich

Joint work of Heiderich, Mario; Niemietz, Marcus; Schuster, Felix; Holz, Thorsten; Schwenk, Jörg

Main reference M. Heiderich, M. Niemietz, F. Schuster, T. Holz, J. Schwenk, “Scriptless attacks: stealing the pie without touching the sill,” in Proc. of the 2012 ACM Conf. on Computer and Communications Security (CCS ’12), pp. 760–771, ACM, New York, NY, USA, 2012.

URL <http://dx.doi.org/10.1145/2382196.2382276>

Due to their high practical impact, Cross-Site Scripting (XSS) attacks have attracted a lot of attention from the security community members. In the same way, a plethora of more or less effective defense techniques have been proposed, addressing the causes and effects of XSS vulnerabilities. As a result, an adversary often can no longer inject or even execute arbitrary scripting code in several real-life scenarios.


In this talk, we examine the attack surface that remains after XSS and similar scripting attacks are supposedly mitigated by preventing an attacker from executing JavaScript code. We address the question of whether an attacker really needs JavaScript or similar functionality to perform attacks aiming for information theft. The surprising result is that an attacker can also abuse Cascading Style Sheets (CSS) in combination with other Web techniques like plain HTML, inactive SVG images or font files. Through several case studies, we introduce the so called scriptless attacks and demonstrate that an adversary might not need to execute code to preserve his ability to extract sensitive information from well protected websites. More

precisely, we show that an attacker can use seemingly benign features to build side channel attacks that measure and exfiltrate almost arbitrary data displayed on a given website.

We conclude this talk with a discussion of potential mitigation techniques against this class of attacks. In addition, we have implemented a browser patch that enables a website to make a vital determination as to being loaded in a detached view or pop-up window. This approach proves useful for prevention of certain types of attacks we here discuss.

4.14 JSAgents

Mario Heiderich (Ruhr-Universität Bochum, DE)


License  Creative Commons BY-NC-ND 3.0 Unported license
© Mario Heiderich

Joint work of Heiderich, Mario; Holz, Thorsten;

This presentation provides an overview on the current status of the JSAgents Project, carried out by the department NDS of the Ruhr-University Bochum. Among a small introductory section, outlines of the used technologies a demonstration is being presented, showing off the XSS mitigation capabilities of the chosen approach.

4.15 Toward decentralized collaborative webapps via knowledge-based security

Michael Hicks (University of Maryland – College Park, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Michael Hicks

Joint work of Hicks, Michael; Mardziel, Piotr; Magill, Stephen; Srivatsa, Mudhakar

Main reference P. Mardziel, S. Magill, M. Hicks, S. Srivatsa, “Dynamic Enforcement of Knowledge-Based Security Policies,” in Proc. of 24th IEEE Computer Security Foundations Symposium (CSF), pp. 114 -128, 2011.

URL <http://dx.doi.org/10.1109/CSF.2011.15>

Cloud- and server-based services own your data. Facebook stores your personal information, likes, contacts, and posts. Other cloud-based services, like Google Drive, similarly store your data. You trust them to do the right thing, but privacy policies are often not in your interests, and are subject to sudden change. On the other hand, it is undeniable that cloud-based services facilitate collaboration because they are extremely easy to use.

We are interested in supporting services with the same ease of use, but with stronger technical guarantees of privacy. One line of work is to maintain your data on a separate, trusted storage server, which controls accesses by outside services, like Facebook. The idea is to maintain a personal information archive under your control. Outside parties can send queries to this archive, to request particular bits of information. Thus an important question is how to decide whether to answer a particular query. To do so, one must determine whether the answer (in combination with answers to previous queries) might reveal too much information, thus defeating the goal of storing data separately in the first place. In my talk, I will present our work on using Bayesian reasoning to assess how much a querier can learn from a particular query, and how to use that information as the foundation of a security policy.

4.16 IceShield: Detection and Mitigation of Malicious Websites with a Frozen DOM

Thorsten Holz (Ruhr-Universität Bochum, DE)

License © ⓘ ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Thorsten Holz

Joint work of Heiderich, Mario; Frosch, Tilman; Holz, Thorsten

Main reference M. Heiderich, T. Frosch, T. Holz, “Iceshield: Detection and mitigation of malicious websites with a frozen dom,” in *Recent Advances in Intrusion Detection*, pp. 281–300. Springer Berlin/Heidelberg, 2011.

URL http://dx.doi.org/10.1007/978-3-642-23644-0_15

In this talk, we provide an overview of a recent project called IceShield and briefly introduce JSAgents, a follow-up project we started a few months ago. Due to its flexibility and dynamic character, JavaScript has become an important tool for attackers. The widespread scripting language often helps them to perform a broad variety of malicious activities, for example to initiate drive-by download exploits or to execute clickjacking attacks. Current defense mechanisms as well as reactive analysis and forensic approaches are often slow or complicated to set up and conduct since an attacker can use many different ways to obfuscate the code or make it hard to obtain a copy of the code in the first place.

In this talk, we introduce a novel approach to analyze this class of attacks by demonstrating how dynamic analysis of websites can be accomplished directly in the browser. We present IceShield, a JavaScript based tool that enables in-line dynamic code analysis as well as de-obfuscation, and a set of heuristics to detect attempts of attacking either a website or the user accessing its contents. Special care needs to be taken to implement the instrumentation in a robust and tamper resistant way since an attacker should not be able to bypass our detection process. We show how features of ECMA Script 5 can be used to freeze object properties, so they cannot be modified during runtime. We implemented a prototype version of IceShield and demonstrate that it detects malicious websites with a small overhead even on devices with limited computing power such as smartphones. Furthermore, IceShield can mitigate detected attacks by changing suspicious elements, so they do not cause harm anymore, thus actually protecting users from such attacks.

4.17 Discovering Concrete Attacks on Website Authorization by Formal Analysis

Sergio Maffeis (Imperial College London, GB)

License © ⓘ ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Sergio Maffeis

Joint work of Chetan, Bansal; Bhargavan, Karthikeyan; Maffeis, Sergio

Main reference C. Bansal, K. Bhargavan, S. Maffeis, “Discovering concrete attacks on website authorization by formal analysis,” in *Proc. of 25th IEEE Computer Security Foundations Symposium (CSF)*, pp. 247–262. IEEE, 2012.


URL <http://dx.doi.org/10.1109/CSF.2012.27>

Social sign-on and social sharing are becoming an ever more popular feature of web applications. This success is largely due to the APIs and support offered by prominent social networks, such as Facebook, Twitter, and Google, on the basis of new open standards such as the OAuth 2.0 authorization protocol. A formal analysis of these protocols must account for malicious websites and common web application vulnerabilities, such as cross-site request forgery and open redirectors. We model several configurations of the OAuth 2.0 protocol

in the applied pi-calculus and verify them using ProVerif. Our models rely on WebSpi, a new library for modeling web applications and web-based attackers that is designed to help discover concrete website attacks. Our approach is validated by finding dozens of previously unknown vulnerabilities in popular websites such as Yahoo and WordPress, when they connect to social networks such as Twitter and Facebook.

4.18 My software has a vulnerability should i worry?

Fabio Massacci (University of Trento – Povo, IT)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Fabio Massacci

Joint work of Massacci, Fabio; Allodi, Luca

In this talk I show that the work of 90% of security researchers (including many people of the audience) to find vulnerabilities or protect against them is utterly useless. The bad guys are exploiting in the wild only a tiny tiny fraction of them.

4.19 You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions

Nick Nikiforakis (KU Leuven, BE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Nick Nikiforakis

Joint work of Nikiforakis, Nick; Invernizzi, Luca; Kapravelos, Alexandros; Van Acker, Steven; Joosen, Wouter; Kruegel, Christopher; Piessens, Frank; Vigna, Giovanni

Main reference N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, G. Vigna, “You are what you include: large-scale evaluation of remote javascript inclusions,” in Proc. of the 2012 ACM Conference on Computer and Communications Security (CCS’12), pp. 736–747, 2012.

URL <http://dx.doi.org/10.1145/2382196.2382274>

An important and widely used feature of JavaScript, is the ability to combine multiple libraries from local and remote sources into the same page, under the same namespace. While this enables the creation of more advanced web applications, it also allows for a malicious JavaScript provider to steal data from other scripts and from the page itself. Today, when developers include remote JavaScript libraries, they trust that the remote providers will not abuse the power bestowed upon them.

In this presentation, we report on a large-scale crawl of more than three million pages of the top 10,000 Alexa sites, and identify the trust relationships of these sites with their library providers. We show the evolution of JavaScript inclusions over time and develop a set of metrics in order to assess the maintenance-quality of each JavaScript provider, showing that in some cases, top Internet sites trust remote providers that could be successfully compromised by determined attackers and subsequently serve malicious JavaScript. In this process, we identify four, previously unknown, types of vulnerabilities that attackers could use to attack popular web sites. Lastly, we review some proposed ways of protecting a web application from malicious remote scripts and show that some of them may not be as effective as previously thought.

4.20 Automatic Quantification of CPU Cache Side-channels

Martin Ochoa (Siemens – München, DE)

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Martin Ochoa

Joint work of Köpf, Boris; Mauborgne, Laurent; Ochoa, Martin

Main reference B. Köpf, L. Mauborgne, M. Ochoa, “Automatic quantification of cache side-channels,” in Proc. of the 24th Int’l Conf. on Computer Aided Verification (CAV’12), pp. 564–580, Springer-Verlag, 2012.

URL http://dx.doi.org/10.1007/978-3-642-31424-7_40

The latency gap between caches and main memory has been successfully exploited for recovering sensitive input to programs, such as cryptographic keys from implementations of AES and RSA in SSL. So far, there are no practical general-purpose countermeasures against this threat. In this talk we describe a novel method for automatically deriving upper bounds on the amount of information about the input that an adversary can extract from a program by observing the CPU’s cache behavior. At the heart of our approach is a novel technique for efficient counting of concretizations of abstract cache states that enables us to connect state-of-the-art techniques for static cache analysis and quantitative information-flow. We implement our counting procedure on top of the AbsInt TimingExplorer, one of the most advanced engines for static cache analysis. We use our tool to perform a case study where we derive upper bounds on the cache leakage of a 128-bit AES executable on an ARM processor with a realistic cache configuration. We also analyze this implementation with a commonly suggested (but until now heuristic) countermeasure applied, obtaining a formal account of the corresponding increase in security.

4.21 FlowFox: an experiment on bringing information flow control to the browser

Frank Piessens (KU Leuven, BE)

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Frank Piessens

Joint work of De Groef, Willem; Devriese, Dominique; Nikiforakis, Nick; Piessens, Frank


Main reference W. De Groef, D. Devriese, N. Nikiforakis, F. Piessens, “FlowFox: a web browser with flexible and precise information flow control,” in Proc. of the 2012 ACM Conference on Computer and Communications Security (CCS ’12), pp. 748–759, ACM, 2012.

URL <http://dx.doi.org/10.1145/2382196.2382275>

FlowFox is a fully functional web browser that implements information flow control for scripts using secure multi-execution. I will briefly sketch FlowFox’s architecture, and briefly report on our experiences with the browser. The main focus of the talk will be on some of the issues that remain unsolved. In particular, I will discuss how the way in which FlowFox specifies and enforces policies may break some of the theoretical properties of secure multi-execution. I will also propose a number of approaches to deal with these issues and hope to get feedback from the audience on which of these approaches makes most sense.

4.22 A Tested Semantics for Getters, Setters, and Eval in JavaScript

Joe Gibbs Politz (Brown University – Providence, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Joe Gibbs Politz

Joint work of Politz, Joe Gibbs; Carroll, Matthew J; Lerner, Benjamin S; Pombrio, Justin; Krishnamurthi, Shriram


Main reference J. Gibbs Politz, M.J. Carroll, B.S. Lerner, J. Pombrio, S. Krishnamurthi, “A tested semantics for getters, setters, and eval in JavaScript,” in Proc. of the 8th Symp. on Dynamic languages (DLS ’12), pp. 1–16, ACM, 2012.

URL <http://dx.doi.org/10.1145/2384577.2384579>

We present S5, a semantics for the strict mode of the ECMAScript5.1 (JavaScript) programming language. S5 shrinks the large source language into a manageable core through an implemented transformation. The resulting specification has been tested against real-world conformance suites for the language. This paper focuses on two aspects of S5: accessors (getters and setters) and eval. Since these features are complex and subtle in JavaScript, they warrant special study. Variations on both features are found in several other programming languages, so their study is likely to have broad applicability.

4.23 Progressive Types

Joe Gibbs Politz (Brown University – Providence, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Joe Gibbs Politz

Joint work of Politz, Joe Gibbs; Quay-de la Vallee, Hannah; Krishnamurthi, Shriram


Main reference J. Gibbs Politz, H. Quay-de la Vallee, S. Krishnamurthi, “Progressive types,” in Proc. of the ACM Int’l Symp. on New ideas, new paradigms, and reflections on programming and software (Onward! ’12), pp. 55-66, ACM, 2012.

URL <http://dx.doi.org/10.1145/2384592.2384599>

As modern type systems grow ever-richer, it can become increasingly onerous for programmers to satisfy them. However, some programs may not require the full power of the type system, while others may wish to obtain these rich guarantees incrementally. In particular, programmers may be willing to exploit the safety checks of the underlying run-time system as a substitute for some static guarantees. Progressive types give programmers this freedom, thus creating a gentler and more flexible environment for using powerful type checkers. In this paper we discuss the idea, motivate it with concrete, real-world scenarios, then show the development of a simple progressive type system and present its (progressive) soundness theorem.

4.24 GlassTube: A Lightweight Approach to Web Application Integrity

Andrei Sabelfeld (Chalmers UT – Göteborg, SE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Andrei Sabelfeld

Joint work of Hallgren, Per; Mauritzson, Daniel; Sabelfeld, Andrei

The HTTP and HTTPS protocols are the main corner stones of the modern web. From a security point of view, they offer an all-or-nothing choice to web applications: either no security guarantees with HTTP or both confidentiality and integrity with HTTPS. However,

in many scenarios confidentiality is not necessary and even undesired, while integrity is essential to prevent attackers from compromising the data stream. We propose GlassTube, a lightweight approach to web application integrity. GlassTube guarantees integrity at application level, without resorting to the heavyweight HTTPS protocol. GlassTube provides a general method for integrity in web applications and smartphone apps. GlassTube is easily deployed in the form of a library on the server side, and offers flexible deployment options on the client side: from dynamic code distribution, which requires no modification of the browser, to browser plugin and smartphone app, which allow smooth key predistribution. The results of a case study with a web-based chat indicate a boost in the performance compared to HTTPS, achieved with no optimization efforts.

4.25 Side channel attacks on the web

Sebastian Schinzel (Universität Erlangen-Nürnberg, DE)

License © ⓘ ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Sebastian Schinzel

Joint work of Schinzel, Sebastian; Freiling, Felix

Main reference F. Freiling, S. Schinzel, “Detecting hidden storage side channel vulnerabilities in networked applications,” *Future Challenges in Security and Privacy for Academia and Industry* (2011), pp. 41–55.

URL http://dx.doi.org/10.1007/978-3-642-21424-0_4

Side channels are vulnerabilities that can be attacked by observing the behavior of applications and by inferring sensitive information just from this behavior. Storage side channels are a new type of side channels which leak information through redundancies in protocols such as HTTP or languages such as HTML. We implement a detection method and test it by applying it to real-world web applications and we find that several widely used web applications are prone to this type of attack.

Because side channel vulnerabilities appear in such a large spectrum of contexts, there does not seem to be a generic way to prevent all side channel attacks once and for all. A practical approach is to research for new side channels and to specifically tailor mitigations for new side channel attacks. We present a new method to mitigate timing side channels in web applications. The method works by padding the response time using a deterministic and unpredictable delay (DUD). We show that DUD offers security guarantees that can be freely traded with performance reduction. By applying this method to vulnerable web applications, we show that the method offers an effective and performance efficient way to mitigate timing side channels.

4.26 How To Break XML Encryption

Juraj Somorovsky (Ruhr-Universität Bochum, DE)

License © ⓘ ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Juraj Somorovsky

Joint work of Jager, Tibor; Somorovsky, Juraj

Main reference T. Jager, J. Somorovsky, “How to break XML encryption,” in *Proc. of the 18th ACM Conf. on Computer and Communications Security (CCS '11)*, pp. 413–422, ACM, 2011.

URL <http://dx.doi.org/10.1145/2046707.2046756>


XML Encryption was standardized by W3C in 2002, and is implemented in XML frameworks of major commercial and open-source organizations like Apache, redhat, IBM, and Microsoft.

It is employed in a large number of major web-based applications, ranging from business communications, e-commerce, and financial services over healthcare applications to governmental and military infrastructures. Our work describes several adaptive chosen-ciphertext attacks against PKCS#1 v1.5 and AES-CBC in XML Encryption. In case of PKCS#1 v1.5, the attacker could recover the secret key used for symmetric encryption by issuing a few millions of messages. In case of AES-CBC, the attacker needs to issue about 14 requests to recover one message byte directly. In a sense, our work applies the attacks of Bleichenbacher (Crypto 1998) and Vaudenay (Eurocrypt 2002) on XML Encryption by exploiting various XML specific side-channels. It shows how complicated it could be to mitigate these attacks and thus motivates for usage of secure cryptographic primitives.

This work is of a particular importance as similar side-channels could arise by implementing different specifications such as JSON Web Encryption or Web Cryptographic API.

4.27 Verifying JavaScript programs with the Dijkstra State Monad

Nikhil Swamy (Microsoft – Redmond, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Nikhil Swamy

Joint work of Swamy, Nikhil; Weinberger, Joel; Schlesinger, Cole; Chen, Juan; Livshits, Benjamin

Several special-purpose systems have been proposed to analyze programs in JavaScript and other dynamically typed languages. However, none of these prior systems support automated, modular verification for both higher-order and stateful features.

This paper proposes a new refinement of the state monad, the Dijkstra state monad, as a way of structuring specifications for higher-order, stateful programs. Relying on a type inference algorithm for the Dijkstra monad, we obtain higher-order verification conditions (VCs) for programs that use a dynamically typed higher-order store. Via a novel encoding, we show that these higher-order VCs can be discharged by an off-the-shelf automated SMT solver. We put the Dijkstra monad to use by building a tool chain to verify JavaScript programs. Our tool chain begins by translating JavaScript programs to F^* , a dependently typed dialect of ML. Within F^* , we define a library for dynamic typing idioms based on the Dijkstra monad. We then infer and solve precise verification conditions for translated JavaScript clients of this library. We report on our experience using this tool chain to verify a collection of web browser extensions for the absence of JavaScript runtime errors. Despite some limitations of our work (e.g., we do not model asynchrony), we conclude that the Dijkstra monadic approach is a promising and powerful way to structure the verification of JavaScript programs within a general purpose dependently typed programming language.

4.28 Fully Abstract Compilation to JavaScript

Nikhil Swamy (Microsoft – Redmond, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Nikhil Swamy

Joint work of Fournet, Cedric; Swamy, Nikhil; Chen, Juan; Dagand, Pierre-Evariste; Strub, Pierre-Yves; Livshits, Benjamin

Many tools allow programmers to develop applications in high-level languages and deploy them in web browsers via compilation to JavaScript. While practical and widely used, these

compilers are ad hoc. No guarantee is provided on their correctness for whole programs, nor their security for programs executed within arbitrary JavaScript contexts.

In this paper, we present a compiler with such positive guarantees. We compile an ML-like language with higher-order functions and references down to JavaScript, while preserving all source program properties. We evaluate our compiler on sample programs, including a series of secure libraries. We illustrate the dangers of JavaScript contexts with a series of attacks against naive scripts. We then give a semantics to JavaScript by translation to F^* , a variant of ML with richer types. Based on lambdaJS, this semantics reflects the main elements of the EcmaScript 5 standard, as well as our experimental findings on dangerous features in JavaScript implementations (implicit coercions, getters and setters, and special arguments, caller, and callee properties). We present our compilation scheme, expressed as a type-preserving translation between fragments of F^* : each source type is mapped to ‘dyn’, the type of Javascript values, refined with a logical specification of its compiled representation. For whole programs, we show that the translation is a forward simulation. For programs executed in untrusted Javascript contexts, we wrap our translation with defensive filters to import and export values while preserving the translation invariant. Relying on type-based invariants and a new notion of applicative bisimilarity, we show full abstraction: two programs are equivalent in all source contexts if and only if their wrapped translations are equivalent in all Javascript contexts. Thus, programmers can produce JavaScript and still rely on static scopes and types for reasoning about their programs.

4.29 behind FlashOver: Automated Discovery of Cross-site Scripting Vulnerabilities in Rich Internet Applications

Steven Van Acker (KU Leuven, BE)

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Steven Van Acker

Joint work of Van Acker, Steven; Nikiforakis, Nick; Desmet, Lieven; Joosen, Wouter; Piessens, Frank

Main reference S. Van Acker, N. Nikiforakis, L. Desmet, W. Joosen, F. Piessens, “FlashOver: Automated discovery of cross-site scripting vulnerabilities in rich internet applications,” AsiaCCS, Seoul, 2-4 May 2012.

FlashOver is a system to automatically scan Rich Internet Applications for XSS vulnerabilities by using a combination of static and dynamic code analysis that reports no false positives. It was used in a large-scale experiment to analyze Flash applications found on the top 1,000 Internet sites, exposing XSS vulnerabilities that could compromise 64 of those sites, of which six are in the top 50. In this talk, we will focus on some open-ended questions that surfaced while we were performing our FlashOver experiment.

4.30 The Security Pains of an Online Bank


John Wilander (Hägersten, SE)

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© John Wilander

Online banking is often used as a primary example of why we need web application security. This talk presents the security challenges from the inside. Session stickiness, session termination, complexity of deploying new countermeasures, and handing of a large legacy web application.

4.31 Stateless CSRF Protection

John Wilander (Hägersten, SE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© John Wilander

CSRF protections most often involve server-side state which makes it hard to convince developers to use them, especially for RESTful services. The so called double submit CSRF protection can be made stateless but then becomes susceptible to subdomain XSS bypass. The current suggestion for stateless anti- CSRF is a triple submit with only a subdomain XSS cookie overflow as known attack vector. This talk will explain the issue and open up for discussion.

5 Break-out sessions

5.1 Break-out session: Cross-Site Scripting (XSS)

This break-out session focused on the problem of Cross-Site Scripting (XSS). In particular, the participants of this break-out session enlisted mitigation techniques used in practice, compiled a survey of recent research activities and open problems in this domain, and did put forward a first step towards a candidate definition of Cross-Site Scripting.

Cross-Site Scripting is a broad category of HTML/JavaScript injection vulnerabilities, typically categorized in three types. The first type of XSS vulnerabilities (XSS-0) is called DOM-based XSS, and operates purely on the client-side. The second (XSS-1) and third type (XSS-2) inject HTML/JavaScript via the server-side, either in a reflected or persistent manner.

The **most common mitigation strategies** for XSS rely on identifying injection attacks and rendering them harmless in the browser environment. For instance, most modern browsers are equipped with XSS filters [5, 54, 46], and support the Content Security Policy (CSP) [61]. Alternatively, untrusted page fragments and scripts can be sandboxed [35], or can be sanitized by browser libraries such as the *toStaticHTML* method in Internet Explorer [13].

Apart from browser mechanisms, other strategies to detect and protect against XSS are the use of automatic sanitization libraries such as HTML Purifier [67], subject the web application to pentesting and static analysis (e.g. [60, 25]), and the use of confined environments such as the Secure ECMAScript library [50] and JavaScript sandboxing approaches [62, 48, 64, 1, 36].

Recent research on XSS includes the automatic context-sensitive sanitization of web content [56, 58, 31], the automated analysis of widgets and applications [57, 29] and novel XSS attacks beyond the classical pattern of HTML and JavaScript injection [34].

In [56], Samuel *et al* propose a type-qualifier mechanism that can be applied onto existing web templating frameworks to achieve context-sensitive sanitization. ScriptGuard [58] focuses on the incorrect use of sanitization libraries (such as context-mismatched sanitization and inconsistent multiple sanitizations) and is capable of detecting and repairing incorrect placement of sanitizers.

In [57], Saxena *et al* propose the tool Kudzu, based on symbolic execution of JavaScript, to discover in a automated way client-side injection vulnerabilities. Gatekeeper [29] allows site administrators to express and enforce security and reliability policies for JavaScript programs, and was successfully applied to automatically analyze JavaScript widgets, with very few false positives and no false negatives.

Finally, Heiderich *et al* demonstrated that a new set of XSS attacks are feasible that do not rely JavaScript, but operate solely on HTML, Cascading Style Sheets (CSS), SVG images and font files [34].

The participants of this break-out session also tried to formulate an accurate and acceptable **definition of Cross-Site Scripting (XSS)**. There was a lot of discussion between the participants, and the candidate definition of XSS stranded on “An XSS attack occurs when a script from an untrusted source is executed in rendering a page” without reaching unanimity. Some of the comments/critics that were mentioned during the break-out session, as well as during the presentation afterwards for the full group, include:

- XSS can be both a client-side and server-side problem. The phrasing “rendering a page” sounds more like client-side only?
- If the unintended or unwanted execution of JavaScript happens via a trusted third party, can it still be called XSS?

- The interpreter is tricked into executing code. Stems from the mix of data and code.
- Unauthorized execution of JavaScript, escalation of privilege.
- XSS used to be about scripting between frames/pages and mostly type 1 (reflected). Has XSS become an umbrella?
- Violate the control flow integrity of the JavaScript program (assuming there is one).
- A script may arise from a dynamic process influenced by more than one source. Therefore the “source of a script” is not always well-defined.
- The untrusted source reflects the trust policy of the site. In principle, this may only exist in the mind of the designer. Or perhaps more accurately, the designer may not know the policy. The attack is relative to the policy.
- Is the source well-defined? Is it better to use principle than source? Maybe. Maybe not.

To conclude, the participants drafted a set of **open research questions related to XSS**, to inspire and guide junior researchers in this research domain.

- Precise definition of XSS
- Does CSP prevent XSS? Does CSP work, in a precise sense?
- Are the CSP policies expressive enough to prevent XSS?
- Is it feasible to apply CSP to new sites? What about securing legacy sites?
- How effective is the state of the art for securing new sites and legacy sites? Can we get XSS prevention to a situation similar to SQL injection?
- Is there an approach towards preventing XSS that is compatible with the complexity of modern web advertisements?

We would like to thank John Wilander for taking notes during this break-out session, and presenting the break-out results to the full group.

5.2 Break-out session: JavaScript

This break-out session focused on the JavaScript language. In particular, the participants of this break-out session enlisted recent achievements in the JavaScript language research, discussed the semantic notion of properties and policies, compiled a survey of recent enforcement techniques, and did put forward a first step towards future directions.

First, the participants gave an overview of the **recent achievements** in the domain of understanding and securing the JavaScript language.

With respect to **formalizing the language semantics**, several lines of work can be identified. Maffei *et al.* have defined an operational semantic [44] for Javascript and a program logic [28]. Alternatively, a small-step operational semantics for a core set of the language has been proposed [30], as well as a desugaring process that turns JavaScript programs into the core language.

The participants of the break-out stated that the hard, remaining bits of work are in formalizing the error model of JavaScript, the thread model (i.e. the asynchronicity), and the DOM/API.

Also quite some work has been invested in defining **safe subsets of JavaScript**, such as FaceBook JS (FBJS) [63] and AdSafe [20]. Also important here is to mention *strict mode*, introduced in the latest specifications of JavaScript. This mode allows an opt-in to a restricted variant of JavaScript, which eliminates some of the typical JavaScript pitfalls.

The participants of the break-out stated that the hard, remaining bits of work are in stating safety properties, as well as ensuring and using them.

Another emerging trend is the use of **JavaScript as an assembly language**. A well-known example of this is the Google Web Toolkit (GWT), in which the compiler transforms the client-side program (written in Java) into JavaScript. Similarly, the multi-tier language Swift [15] uses JIF as source language, and applies GWT to transform the client-side code to JavaScript. Other practical examples of using JavaScript as an assembly language include CoffeeScript, Dart and TypeScript.

The participants of the break-out stated that the hard, remaining bits of work are to securely isolate compiled JS from *raw* potentially untrusted JavaScript, as well as the safe interaction and cohabitation of JavaScript from multiple sources (Mashic [43], JSand [1]).

An interesting discussion during this break-out session was the **semantic difference between security properties and security policies**. As a conclusion of the discussion, we agreed that properties describe what you want for your system, namely the end-to-end characteristics. An example security property could for instance be the separation of duties. Policies, in contrast, describe how to ensure the security properties.

In the context of security policies become more and more complex to express, an opportunity was raised to converge towards policy templates.

The participants compiled a survey of the various **enforcement techniques** to secure JavaScript. There is a broad domain, and during the discussion the enforcement techniques were categorized in three categories:

Static enforcement techniques In the domain of static enforcement, a first set of techniques (such as AdJail [62], Treehouse [36] and JSand [1]) realizes the enforcement by construction. A second set of techniques uses verification: symbolic execution, static verification [29], types or separation logic [28].

Dynamic enforcement techniques (within the language) The dynamic enforcement typically happens with wrappers. Interesting to mention here is the work on CAJA [49] en AdSafe [20].

Dynamic enforcement techniques (below the language) Enforcement is also possible in the underlying infrastructure, and can happen via proxies or in the browser.

This break-out session ended with summing up some interesting thoughts and possible directions for future research.

- We noticed a transition of JavaScript towards the desktop (Windows 8/Metro), and towards the server (Node.js)
- JavaScript security is hard: Everything is dynamic . . . Everything is mutable . . . Everything leaks . . . Everything . . .
- We have made quite some progress with JavaScript, it is time to rise above JavaScript
- You can break the web if you give something in return...
- We need to identify good programming guidelines

We would like to thank Ranjit Jhala for taking notes during this break-out session, and presenting the break-out results to the full group.

5.3 Break-out session: Information-Flow

This break-out session focused on information-flow control. In particular, the participants of this break-out session highlighted the importance of information-flow control in modern web application scenarios, and compiled a survey of recent research activities and open problems in this domain.

Information-flow control started in the mid seventies with research on multi-level security operating systems [23], and experienced a revival of interest in the nineties with the rise of mobile code. Recently, the research domain has become fully active thanks to emerging challenges in the web application context, in which code from multiple stakeholders coexists in web mashups [22] or via the integration of third-party JavaScript code [52].

Access control is often used to enforce safety properties in software systems, by controlling access to sensitive resources or APIs at run-time. Examples in the web context are the declarative access control in JEE containers, programmatic access control as part of the application itself, and access control mechanisms in the browser environment such as the Same-Origin Policy [55]. More recently, capability-based security achieves access control guarantees without the need of run-time checks. In the mashup context for instance, JavaScript components can be sandboxed in language subsets such as CAJA [49] and AdSafe [20].

Access control is often easy to implement, and a number of sandboxing techniques are available for JavaScript [50, 62, 48, 64, 1, 36]. However, sometimes it is also **necessary to track what happens to the information once it is accessed**. Some of the interesting web scenario's discussed in this context during this Dagstuhl seminar include Google Analytics, Google Maps, a loan calculator, and Tynt.

In order to achieve confidentiality of the data, beyond the point of access, information-flow control does not rely on a single *secure state* or *secure trace* of the program, but needs to take into account all possible traces and their relation (e.g. to detect implicit flows in the application).

Recent research on information-flow control includes dynamic information-flow enforcement and hybrid analysis techniques for both the client-side as the server-side, and novel security policies.

In recent information-flow control research for JavaScript, there has been a significant thrust on **dynamic information-flow enforcement** [21, 3, 33]. For example, FlowFox [21] is a fully functional web browser that implements the Secure Multi-Execution (SME) [24] technique for information-flow control on top of Firefox. Similarly, the Facets mechanism [3], inspired by SME, uses *facet values* to simultaneously and efficiently simulate multiple executions for different security levels. Finally, JsFlow [33] applies a dynamic type system to a core of JavaScript to prohibit information leaks from the program's secret sources to the program's public sinks. A more complete survey of these techniques can be found in [7].

Several researchers have proposed **static and hybrid analysis** techniques to achieve a more efficient information-flow enforcement. In [18], a framework for staging information-flow properties has been proposed to compute a minimal set of syntactic residual checks that are performed on the remaining code when it is dynamically loaded. Similarly, Hammer *et al.* track information flow dynamically but rely on intra-procedural static analysis to capture implicit flows [40].

Several **empirical studies** have been conducted to detect violations against information-flow policies [66, 39]. In [66], tainting techniques are used to track the flow of sensitive information inside the browser in order to stop Cross-Site Scripting attacks. Moreover, Jang *et al.* surveyed the prevalence of privacy violating flows on the Alexa top 50.000 websites to detect four privacy-violating flows: cookie stealing, location hijacking, history sniffing, and behavior tracking [39].

On the **server-side**, it is important to mention the work of the researchers at Cornell [16, 15, 42]. Swift is a multi-tier approach to build web applications that are secure by construction [15]. The source code, written in Jif, is automatically partitioned by the compiler into JavaScript code running in the browser, and Java code running on the server,

and code placement is constrained by information flow policies that strongly enforce the confidentiality and integrity of server-side information. Alternatively, SELinks [19] is a programming language focused on building secure multi-tier web applications, capable of controlling access to labeled data. Moreover, Off-the-Record Messaging provides a server-side enforcement for cryptographic security to achieve confidential communication with plausible deniability [11].

Finally, there were several directions identified in state-of-the-art policies for information-flow control. First, a set of quantitative security policies takes into account the termination and progress channel, as well as side channels [32]. Secondly, more expressive sets of information-flow control policies take into account mutual distrust and a decentralized authority [51, 45]. In such a setting, policies should allow to share information with distrusted script, while still controlling how the script propagates the shared information. Other research in this domain focuses on defining and evaluating lattices for modern web applications and mashups, and the use of relational reasoning in information-flow control.

To conclude, the participants drafted a set of **promising future directions** in information-flow control research:

- Hybrid access control and information-flow control mechanisms
- Extend facets to integrity
- Increase usability of information-flow control mechanisms
- For systems: distributed information-flow control and tainting
- For databases: the provenance of data
- For crypto: achieve robustness, secure multi-party, zero knowledge, secure information retrieval
- For privacy: differential privacy

We would like to thank Andrei Sabelfeld for taking notes during this break-out session, and presenting the break-out results to the full group.

References

- 1 Pieter Agten, Steven Van Acker, Yoran Brondsema, Phu H. Phung, Lieven Desmet, and Frank Piessens. Jsand: complete client-side sandboxing of third-party javascript without browser modifications. In *Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC '12*, pages 1–10, New York, NY, USA, 2012. ACM.
- 2 Devdatta Akhawe, Prateek Saxena, and Dawn Song. Privilege separation in html5 applications. pages 23–23, 2012.
- 3 Thomas H. Austin and Cormac Flanagan. Multiple facets for dynamic information flow. pages 165–178, 2012.
- 4 Chetan Bansal, Karthikeyan Bhargavan, and Sergio Maffei. Discovering concrete attacks on website authorization by formal analysis. In Chong [14], pages 247–262.
- 5 Daniel Bates, Adam Barth, and Collin Jackson. Regular expressions considered harmful in client-side xss filters. In *Proceedings of the 19th international conference on World wide web, WWW '10*, pages 91–100, New York, NY, USA, 2010. ACM.
- 6 Karthikeyan Bhargavan, Cédric Fournet, Ricardo Corin, and Eugen Zalinescu. Verified cryptographic implementations for tls. *ACM Trans. Inf. Syst. Secur.*, 15(1):3, 2012.
- 7 Nataliia Bielova. Survey on javascript security policies and their enforcement mechanisms in a web browser. 2012. Under submission.
- 8 Arnar Birgisson, Daniel Hedin, and Andrei Sabelfeld. Boosting the permissiveness of dynamic information-flow tracking by testing. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Computer Security ESORICS 2012*, volume 7459 of *Lecture Notes in Computer Science*, pages 55–72. Springer Berlin Heidelberg, 2012.
- 9 Dan Boneh, Ulfar Erlingsson, Martin Johns, and Benjamin Livshits. 09141 abstracts collection – web application security. In Dan Boneh, Ulfar Erlingsson, Martin Johns, and Benjamin Livshits, editors, *Web Application Security*, number 09141 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2010. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.
- 10 Dan Boneh, Ulfar Erlingsson, Martin Johns, and Benjamin Livshits. 09141 executive summary – web application security. In Dan Boneh, Ulfar Erlingsson, Martin Johns, and Benjamin Livshits, editors, *Web Application Security*, number 09141 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2010. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.
- 11 Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society, WPES '04*, pages 77–84, New York, NY, USA, 2004. ACM.
- 12 Bastian Braun, Patrick Gemein, Hans P. Reiser, and Joachim Posegga. Control-flow integrity in web applications. In *International Symposium on Engineering Secure Software and Systems (ESSoS 2013)*, February 2013. [to appear].
- 13 Internet Explorer Developer Center. Making HTML safer: details for toStaticHTML (Windows Store apps using JavaScript and HTML). <http://msdn.microsoft.com/en-us/library/ie/hh465388.aspx>, 2012.
- 14 Stephen Chong, editor. *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*. IEEE, 2012.
- 15 Stephen Chong, Jed Liu, Andrew C. Myers, Xin Qi, K. Vikram, Lantian Zheng, and Xin Zheng. Secure web applications via automatic partitioning. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, SOSP '07*, pages 31–44, New York, NY, USA, 2007. ACM.
- 16 Stephen Chong, K. Vikram, and Andrew C. Myers. Sif: enforcing confidentiality and integrity in web applications. In *Proceedings of 16th USENIX Security Symposium on*

- USENIX Security Symposium*, SS'07, pages 1:1–1:16, Berkeley, CA, USA, 2007. USENIX Association.
- 17 Ravi Chugh, David Herman, and Ranjit Jhala. Dependent types for javascript. In *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, OOPSLA '12, pages 587–606, New York, NY, USA, 2012. ACM.
 - 18 Ravi Chugh, Jeffrey A. Meister, Ranjit Jhala, and Sorin Lerner. Staged information flow for javascript. In *Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '09, pages 50–62, New York, NY, USA, 2009. ACM.
 - 19 Brian J. Corcoran, Nikhil Swamy, and Michael Hicks. Cross-tier, label-based security enforcement for web applications. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, SIGMOD '09, pages 269–282, New York, NY, USA, 2009. ACM.
 - 20 Douglas Crockford. ADsafe – making JavaScript safe for advertising. <http://adsafe.org/>.
 - 21 Willem De Groef, Dominique Devriese, Nick Nikiforakis, and Frank Piessens. Flowfox: a web browser with flexible and precise information flow control. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 748–759, New York, NY, USA, 2012. ACM.
 - 22 Philippe De Ryck, Maarten Decat, Lieven Desmet, Frank Piessens, and Wouter Joosen. Security of web mashups: a survey. In *Proceedings of the 15th Nordic conference on Information Security Technology for Applications*, NordSec'10, pages 223–238, Berlin, Heidelberg, 2012. Springer-Verlag.
 - 23 Dorothy E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, May 1976.
 - 24 Dominique Devriese and Frank Piessens. Noninterference through secure multi-execution. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 109–124, Washington, DC, USA, 2010. IEEE Computer Society.
 - 25 Veracode Fergal Glynn. Static Code Analysis. <http://www.veracode.com/security/static-code-analysis>, 2012.
 - 26 Cedric Fournet, Nikhil Swamy, Juan Chen, Pierre-Evariste Dagand, Pierre-Yves Strub, and Benjamin Livshits. Fully abstract compilation to javascript. In *Proceedings of the Symposium on Principles of Programming Languages (POPL)*, January 2013.
 - 27 Felix Freiling and Sebastian Schinzel. Detecting hidden storage side channel vulnerabilities in networked applications. In Jan Camenisch, Simone Fischer-Hübner, Yuko Murayama, Armand Portmann, and Carlos Rieder, editors, *Future Challenges in Security and Privacy for Academia and Industry*, volume 354 of *IFIP Advances in Information and Communication Technology*, pages 41–55. Springer Berlin Heidelberg, 2011.
 - 28 Philippa Anne Gardner, Sergio Maffei, and Gareth David Smith. Towards a program logic for javascript. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '12, pages 31–44, New York, NY, USA, 2012. ACM.
 - 29 Salvatore Guarnieri and Benjamin Livshits. Gatekeeper: Mostly static enforcement of security and reliability policies for javascript code. In *Proceedings of the Usenix Security Symposium*, August 2009.
 - 30 Arjun Guha, Claudiu Saftoiu, and Shriram Krishnamurthi. The essence of javascript. In *Proceedings of the 24th European conference on Object-oriented programming*, ECOOP'10, pages 126–150. Springer-Verlag, Berlin, Heidelberg, 2010.
 - 31 Google Web Toolkit (GWT). AutoEscape implementation for GWT. <http://code.google.com/p/google-web-toolkit/source/browse/tools/lib/streamhtmlparser/streamhtmlparser-jsilver-r10/streamhtmlparser-jsilver-r10-1.5.jar>, 2010.

- 32 D. Hedin and A. Sabelfeld. A perspective on information-flow control. *Proc. of the 2011 Marktoberdorf Summer School*, 2011.
- 33 Daniel Hedin and Andrei Sabelfeld. Information-flow security for a core of javascript. In Chong [14], pages 3–18.
- 34 Mario Heiderich, Marcus Niemietz, Felix Schuster, Thorsten Holz, and Jörg Schwenk. Scriptless attacks: stealing the pie without touching the sill. In *Proceedings of the 2012 ACM conference on Computer and communications security, CCS '12*, pages 760–771, New York, NY, USA, 2012. ACM.
- 35 I. Hickson and D. Hyatt. HTML 5 Working Draft - The sandbox Attribute. <http://www.w3.org/TR/html5/the-iframe-element.html#attr-iframe-sandbox>, June 2010.
- 36 Lon Ingram and Michael Walfish. TreeHouse: JavaScript sandboxes to help web developers help themselves. In *USENIX ATC*, 2012.
- 37 Tibor Jager, Sebastian Schinzel, and Juraj Somorovsky. Bleichenbacher’s attack strikes again: Breaking pkcs#1 v1.5 in xml encryption. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Computer Security ESORICS 2012*, volume 7459 of *Lecture Notes in Computer Science*, pages 752–769. Springer Berlin Heidelberg, 2012.
- 38 Tibor Jager and Juraj Somorovsky. How to break xml encryption. In *Proceedings of the 18th ACM conference on Computer and communications security, CCS '11*, pages 413–422, New York, NY, USA, 2011. ACM.
- 39 Dongseok Jang, Ranjit Jhala, Sorin Lerner, and Hovav Shacham. An empirical study of privacy-violating information flows in javascript web applications. In *Proceedings of the 17th ACM conference on Computer and communications security, CCS '10*, pages 270–283, New York, NY, USA, 2010. ACM.
- 40 Seth Just, Alan Cleary, Brandon Shirley, and Christian Hammer. Information flow analysis for javascript. In *Proceedings of the 1st ACM SIGPLAN international workshop on Programming language and systems technologies for internet clients, PLASTIC '11*, pages 9–18, New York, NY, USA, 2011. ACM.
- 41 Boris Köpf, Laurent Mauborgne, and Martín Ochoa. Automatic quantification of cache side-channels. In *Computer Aided Verification*, volume 7358 of *Lecture Notes in Computer Science*, pages 564–580. Springer Berlin Heidelberg, 2012.
- 42 Jed Liu, Michael D. George, K. Vikram, Xin Qi, Lucas Waye, and Andrew C. Myers. Fabric: a platform for secure distributed computation and storage. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, SOSP '09*, pages 321–334, New York, NY, USA, 2009. ACM.
- 43 Zhengqin Luo and Tamara Rezk. Mashic compiler: Mashup sandboxing based on inter-frame communication. In Chong [14], pages 157–170.
- 44 Sergio Maffei, John C. Mitchell, and Ankur Taly. An operational semantics for javascript. In *Proceedings of the 6th Asian Symposium on Programming Languages and Systems, APLAS '08*, pages 307–325. Springer-Verlag, Berlin, Heidelberg, 2008.
- 45 Jonas Magazinius, Aslan Askarov, and Andrei Sabelfeld. Decentralized delimited release. In *Proceedings of the 9th Asian conference on Programming Languages and Systems, APLAS'11*, pages 220–237, Berlin, Heidelberg, 2011. Springer-Verlag.
- 46 Giorgio Maone. NoScript - JavaScript/Java/Flash blocker for a safer Firefox experience! <http://noscript.net/>, 2012.
- 47 Piotr Mardziel, Stephen Magill, Michael Hicks, and Mudhakar Srivatsa. Dynamic enforcement of knowledge-based security policies. *2012 IEEE 25th Computer Security Foundations Symposium*, 0:114–128, 2011.
- 48 Leo Meyerovich and Benjamin Livshits. ConScript: Specifying and enforcing fine-grained security policies for Javascript in the browser. In *Proc. of SP'10*, 2010.

- 49 M. S. Miller, M. Samuel, B. Laurie, I. Awad, and M. Stay. Caja: Safe active content in sanitized javascript. <http://google-caja.googlecode.com/files/caja-spec-2008-01-15.pdf>, January 2008.
- 50 Mark Samuel Miller. Secure EcmaScript 5. <http://code.google.com/p/es-lab/wiki/SecureEcmaScript>.
- 51 Andrew C. Myers and Barbara Liskov. A decentralized model for information flow control. In *Proceedings of the sixteenth ACM symposium on Operating systems principles*, SOSP '97, pages 129–142, New York, NY, USA, 1997. ACM.
- 52 Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 736–747, New York, NY, USA, 2012. ACM.
- 53 Joe Gibbs Politz, Matthew J. Carroll, Benjamin S. Lerner, Justin Pombrio, and Shriram Krishnamurthi. A tested semantics for getters, setters, and eval in javascript. In *Proceedings of the 8th symposium on Dynamic languages*, DLS '12, pages 1–16, New York, NY, USA, 2012. ACM.
- 54 David Ross. IE 8 XSS Filter Architecture / Implementation. <http://blogs.technet.com/b/srd/archive/2008/08/19/ie-8-xss-filter-architecture-implementation.aspx>, 2008.
- 55 J. Ruderman. Same origin policy for javascript, 2009.
- 56 Mike Samuel, Prateek Saxena, and Dawn Song. Context-sensitive auto-sanitization in web templating languages using type qualifiers. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 587–600, New York, NY, USA, 2011. ACM.
- 57 Prateek Saxena, Devdatta Akhawe, Steve Hanna, Feng Mao, Stephen McCamant, and Dawn Song. A symbolic execution framework for javascript. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 513–528, Washington, DC, USA, 2010. IEEE Computer Society.
- 58 Prateek Saxena, David Molnar, and Benjamin Livshits. Scriptgard: automatic context-sensitive sanitization for large-scale legacy web applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 601–614, New York, NY, USA, 2011. ACM.
- 59 S. Schinzel. An efficient mitigation method for timing side channels on the web. In *2nd International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*, 2011.
- 60 HP Enterprise Security. HP Fortify Static Code Analyzer (SCA). <http://www.hpenterprisesecurity.com/products/hp-fortify-software-security-center/hp-fortify-static-code-analyzer>, 2012.
- 61 Brandon Sterne and Adam Barth. Content security policy. <http://www.w3.org/TR/CSP/>, November 2011.
- 62 Mike Ter Louw, Karthik Thotta Ganesh, and V.N. Venkatakrishnan. AdJail: Practical Enforcement of Confidentiality and Integrity Policies on Web Advertisements. In *19th USENIX Security Symposium*, August 2010.
- 63 The FaceBook Team. FBJS. <http://wiki.developers.facebook.com/index.php/FBJS>.
- 64 Steven Van Acker, Philippe De Ryck, Lieven Desmet, Frank Piessens, and Wouter Joosen. WebJail: least-privilege integration of third-party components in web mashups. ACSAC '11, pages 307–316, New York, NY, USA, 2011. ACM.
- 65 Steven Van Acker, Nick Nikiforakis, Lieven Desmet, Wouter Joosen, and Frank Piessens. FlashOver: Automated discovery of cross-site scripting vulnerabilities in rich internet applications. In *AsiaCCS, Seoul, 2-4 May 2012*, May 2012.

- 66 Philipp Vogt, Florian Nentwich, Nenad Jovanovic, Engin Kirda, Christopher Krügel, and Giovanni Vigna. Cross site scripting prevention with dynamic data tainting and static analysis. In *NDSS*. The Internet Society, 2007.
- 67 Edward Z. Yang. HTML Purifier. <http://htmlpurifier.org/>, 2012.

Participants

- Marco Balduzzi
TREND MICRO Italy S.r.l. –
Sesto San Giovanni, IT
- Nataliia Bielova
INRIA – Rennes, FR
- Arnar Birgisson
Chalmers UT – Göteborg, SE
- Egon Börger
University of Pisa, IT
- Bastian Braun
Universität Passau, DE
- Juan Chen
Microsoft Res. – Redmond, US
- Ravi Chugh
University of California – San
Diego, US
- Jorge Cuellar
Siemens – München, DE
- Valentin Dallmeier
Universität des Saarlandes, DE
- Philippe De Ryck
KU Leuven, BE
- Lieven Desmet
KU Leuven, BE
- Akhawe Devdatta
University of California –
Berkeley, US
- Daniele Filaretti
Imperial College London, GB
- Cormac Flanagan
University of California – Santa
Cruz, US
- Cédric Fournet
Microsoft Research UK –
Cambridge, GB
- Michael Franz
Univ. California – Irvine, US
- Dieter Gollmann
TU Hamburg–Harburg, DE
- Arjun Guha
Cornell University – Ithaca, US
- Daniel Hedin
Chalmers UT – Göteborg, SE
- Mario Heiderich
Ruhr–Universität Bochum, DE
- Boris Hemkemeier
Commerzbank AG –
Frankfurt, DE
- Michael Hicks
University of Maryland – College
Park, US
- Thorsten Holz
Ruhr–Universität Bochum, DE
- Thomas Jensen
INRIA – Rennes, FR
- Ranjit Jhala
University of California – San
Diego, US
- Martin Johns
SAP Research – Karlsruhe, DE
- Shriram Krishnamurthi
Brown Univ. – Providence, US
- Benjamin Livshits
Microsoft – Redmond, US
- Sergio Maffeis
Imperial College London, GB
- Fabio Massacci
University of Trento – Povo, IT
- John C. Mitchell
Stanford University, US
- Nick Nikiforakis
KU Leuven, BE
- Martin Ochoa
Siemens – München, DE
- Frank Piessens
KU Leuven, BE
- Joe Gibbs Politz
Brown Univ. – Providence, US
- Joachim Posegga
Universität Passau, DE
- Tamara Rezk
INRIA Sophia Antipolis, FR
- Eric Rothstein
Universität Passau, DE
- Andrei Sabelfeld
Chalmers UT – Göteborg, SE
- Sebastian Schinzel
Univ. Erlangen–Nürnberg, DE
- Juraj Somorovsky
Ruhr–Universität Bochum, DE
- Nikhil Swamy
Microsoft – Redmond, US
- Steven Van Acker
KU Leuven, BE
- John Wilander
Hägersten, SE

