09141 Abstracts Collection
Web Application Security
— Dagstuhl Seminar —

Dan Boneh[1], Ulfar Erlingsson[2], Martin Johns[3] and Benjamin Livshits[4]

[1] Stanford University, USA
dabo@cs.stanford.edu
[2] Microsoft - Mountain View, USA
ulfar@ru.is
[3] Universität Passau, D
mj@martinjohns.com
[4] Microsoft Research - Redmond, USA
livshits@microsoft.com

Abstract. From 29th March to 3rd April 2009 the Dagstuhl Seminar 09141 *Web Application Security* was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar are put together in this paper. Links to full papers (if available) are provided in the corresponding seminar summary document.

Keywords. Web applications, Security, Ajax, Web 2.0, Analysis for security, Browser design, Distributed applications

09141 Executive Summary – Web Application Security

Web applications are ubiquitous nowadays. Consequently, the field of Web application security is of ever rising significance. This Dagstuhl seminar was conducted to assemble researchers active in the domain to gain a first comprehensive overview of this young discipline in security research. From a content perspective, the topic was explored in a great variety of directions, including for instance Web browser-based security measures, language-based techniques, software engineering centric methods, run-time enforcement, static analysis, or formal approaches.

Keywords: Web applications, Security, Ajax, Web 2.0, Analysis for security, Browser design, Distributed applications

Joint work of: Boneh, Dan; Erlingsson, Ulfar; Johns, Martin; Livshits, Benjamin

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2010/2725

Network Layer 8: The Information Layer

Kevin Borders (Univ. of Michigan - Ann Arbor, US)

Each layer in the network stack adds a level of abstraction to simplify communication. The current top of the network stack, the application layer (layer 7), contains arbitrary-length messages between client and server. Large portions of these messages, however, are constrained by web protocols. Much like TCP and IP headers, constrained fields in HTTP messages serve only as a vehicle for higher-level communication. This work identifies a new top layer in the network stack: the information layer. Consider the search term in a web search, such as a "security." In a HTTP request, this term is part of the information layer, while everything else, including headers and fixed protocol fields, is part of the lower application layer.

Identifying and isolating the information layer enables a variety of novel security systems. The lower application-layer fields can be checked against the HTTP and HTML specifications to discover illicit communication. An information-layer proxy can receive keyboard and mouse events from the client, and then generate HTTP requests on its behalf. Such a proxy has the potential to completely thwart malicious tunneling over the Internet as we know it today. This talk explores the design and security implications of systems that separate information layer data from today's web traffic, and suggests future research directions in the area.

Joint work of: Borders, Kevin; Prakash, Atul

Know Thyself

Dieter Gollmann (TU Hamburg-Harburg, DE)

We suggest to treat current challenges in web applications security such as cross-site scripting not as code-injection attacks but as failures of authentication. Here, authentication refers to the origin of HTML requests and responses, and of their individual components. In this framework, we can investigate to which extent limited flavours of authentication, such as being able to recognize ones own requests, can be an effective basis of enforcing origin-based security policies.

Keywords: Authentication, recognition, cross-site scripting

Blue Sky: Topics and summary

Ulfar Erlingsson (Reykjavik University, IS)

An open discussion among seminar attendees: Can we get a clearer, usable definition of Web application security?

What about Web applications by themselves? If we step back from the attacks and mechanisms that we have been focusing on, can we reach agreement on a security policy that we want our mechanisms to enforce? Even if we have to do away with legacy support, and start from first principles, could we get precise, useful working definitions? A summary describes a handful of points where the seminar seemed in concensus.

Safe Extensions for the Web: Overview of technologies

Ulfar Erlingsson (Reykjavik University, IS)

Numerous technologies can be used to implement extension frameworks that provide basic safety properties. Safe higher-level languages with garbage collection are not the only option. Hardware-based isolation, fast machine-code interpretation, and statically-verifiable machine-code binaries are all viable alternatives, each with slightly different safety properties and tradeoffs. By correctly using such technologies, one can get to the hard questions, which are the same as for JavaScript extensions: how to ensure safety for the interfaces at the boundary of an extension.

User-Interface Shepherding: Thwarting UI Redressing Attacks with Verifiable User Actions

David Evans (University of Virginia, US)

User intentions are at the heart of security, but not considered by current systems in any systematic way. Information about user intentions can be used to construct better access control policies, thwart misdirection attacks such as phishing and clickjacking, and support other security techniques such as anomaly-based intrusion detection. Any decisions based on user intentions require a secure method for collecting user behavior including information about the user interface elements with which the user is interacting.

We present a method for securely recording user interactions with graphical user interfaces. Our approach combines direct observation of user interface elements within the operating system with external observation of user input and graphical output. We use comparisons with bitmap images to verify the visual consistency of the user interface elements. We demonstrate that policies taking advantage of information about user intentions can limit damage caused by malware and thwart misdirection attacks such as clickjacking and cross-site-request forgery.

Joint work of: Shirley, Jeffrey; Evans, David

Universally Composable Framework for the Analysis of Browser-based Protocols

Sebastian Gajek (Ruhr-Universität Bochum, DE)

Browser-based security protocols perform cryptographic tasks within the design constraints of commodity browsers. They are the bearer protocols for many security critical applications on the Internet. Roughly speaking, they are the offspring of key exchange and secure sessions protocols. Although browser-based protocols are widely deployed, their security has not been thourghly proved under standard cryptographic assumptions. Unfortunately, the lack of precise protocol definitions and of underlying formal models hampers the systematic design and a cryptographically faithful analysis of browser-based protocols.

We present a security model for the analysis of this important class of protocols based on the Universal Composability framework [Canetti, FOCS'01].

Keywords: Universal Composition, Design, Analysis, Browser-based protocols

Mostly Static Enforcement of Security and Reliability Policies for JavaScript

Salvatore Guarnieri (University of Washington, US)

The advent of Web 2.0 has lead to the proliferation of client-side code that is typically written in JavaScript. This code is often combined or mashed-up with other code and content from disparate, mutually untrusting parties, leading to undesirable security and reliability consequences.

This work proposes GATEKEEPER, a mostly static approach for soundly enforcing security and reliability policies for JavaScript programs. GATEKEEPER is a highly extensible system with a rich, expressive policy language, allowing the hosting site administrator to formulate their policies as succinct Datalog queries. The primary application of GATEKEEPER is in reasoning about JavaScript widgets such as those hosted by widget portals Live.com and Google/IG. Widgets submitted to these sites can be either malicious or just buggy and poorly written, and the hosting site has the authority to reject the submission of widgets that do not meet the site's security or reliability policies. To show the practicality of our approach, we describe nine representative security and reliability policies. While no obviously malicious widgets were discovered, widgets were found to pollute the global namespace, modify built-in objects, and violate suggestions provided by the widget hosting site.

Joint work of: Guarnieri, Salvatore; Livshits, Benjamin

Penetration Testing with Improved Input Vector Identification

William Halfond (Georgia Institute of Technology, US)

Penetration testing is widely used to help ensure the security of web applications. It discovers vulnerabilities by simulating attacks from malicious users on a target application. Identifying the input vectors of a web application and checking the results of an attack are important parts of penetration testing, as they indicate where an attack could be introduced and whether an attempted attack was successful. Current techniques for identifying input vectors and checking attack results are typically ad-hoc and incomplete, which can cause parts of an application to be untested and leave vulnerabilities undiscovered. In this paper, we propose a new approach to penetration testing that addresses these limitations by leveraging two recently developed analysis techniques. The first is used to identify a web application's possible input vectors, and the second is used to automatically check whether an attack resulted in an injection. To empirically evaluate our approach, we compare it against a state-of-the-art, alternative technique.

Our results show that our approach performs a more thorough penetration testing and leads to the discovery of more vulnerabilities.

Joint work of: Halfond, William G.J.; Roy Choudhary, Shauvik; Orso, Alessandro

Security Challenges in Enterprise Mashups

Jochen Haller (SAP AG - Walldorf, DE)

The Web 2.0 motivates end users to create and share contents. Numerous new technologies, e.g. to create service mashups, emerged that allow for simplified creation, sharing and composition of this user created content. When used in a private context, security and privacy issues are frequently neglected. This changes when mashup technologies are applied to relate and combine not only personal, but also security critical corporate data in so-called enterprise mashups.

This talk presents work in progress that intends to provide a software architecture for a secure and privacy aware provisioning of enterprise mashup technology to corporate end users. Security challenges from a technology, but also organisational and process perspective are presented, as well as measures addressing those according to corporate policy.

Type-directed coercion insertion for security enforcement

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

A number of important program rewriting scenarios can be recast as type-directed coercion insertion.

These range from more theoretical applications such as coercive subtyping and supporting overloading in type theories, to more practical applications such as integrating static and dynamically typed code using gradual typing, and inlining code to enforce security policies such as access control, tainting, and provenance tracking. In this work we give a general theory of type-directed coercion insertion based on generating and inserting coercions. We specifically explore the inherent tradeoff between expressiveness and ambiguity— the more powerful the strategy for generating coercions, the greater the possibility of several, semantically distinct rewritings for a given program. To demonstrate the potential of our approach to security concerns, we work out two examples in detail, one for ensuring sanitization of user data when constructing HTML documents, and one for tracking provenance in database computations.

Keywords: Type coercion, program rewriting, security enforcement

Joint work of: Swamy, Nikhil; Hicks, Michael; Bierman, Gavin S.

Protecting Browsers Against Drive-By Download Attacks

Thorsten Holz (Universität Mannheim, DE)

Nowadays, one common attack vector is a so called drive-by download attack. In this scenario, an attacker lures the victim into opening a website, which then tries to exploit a vulnerability in the visitor's web browser. Recent studies by Provos et al. and our own observations show that this attack vector is getting more and more popular within the attacker community. We thus need better protection mechanisms to inhibit this kind of attacks.

One possibility to prohibit certain kinds of drive-by download attacks lies in the fact that these attacks are commonly carried out with the help of JavaScript and a technique called heap spraying. By analyzing the web site and especially the JavaScript code delivered to the web browser, we can thus detect certain classes of attacks. We implemented a tool to perform both static and dynamic analysis for a given website: In the static analysis part, we check different properties of the site, e.g., whether or not iFrames are included, from which domain additional content is loaded, and if the JavaScript on the site matches specific malicious signatures. In the dynamic analysis phase, we execute the script within the JavaScript interpreter Spidermonkey, instrument the execution, and try to detect anomalies during runtime. The tool itself is implemented as a browser helper object (BHO) to enable an integration into Internet Explorer.

In this talk, we discuss the current state of the project and preliminary results. Furthermore, we present real-world experience with client honeypots, i.e., honeypots that are used to study malicious websites. Based on these observations the limitations of all client-side instrumentations / browser-based protection mechanisms become clear.

Joint work of: Holz, Thorsten; Dewald, Andreas

2020 Foresight: Web application security in 11 years

Trevor Jim (AT&T Research - Florham Park, US)

Transformative technologies often take 40+ years to progress from invention to widespread deployment—I will illustrate this with several historical examples related to security. This fact has both good and bad implications. The bad news is that you may have to wait several decades for your Turing Award. The good news is that the next important advances in web application security have already been in the pipeline for several decades, even preceding the web; we can predict them, and position our research to anticipate their adoption. In this talk I will give my predictions for two security technologies which will come to fruition in the next decade in web applications.

Secure Code Generation for Web Applications

Martin Johns (Universität Passau, DE)

A large percentage of recent security problems, such as Cross-site Scripting or SQL injection, is caused by string-based code injection vulnerabilities. Most of these vulnerabilities exist because of implicit code creation through string serialization. Based on an analysis of the vulnerability classŠ underlying mechanisms, we propose a general approach to outfit modern programming languages with mandatory means for explicit and secure code generation which provide strict separation between data and code. Using an exemplified implementation for the languages Java and HTML/JavaScript respectively, we show how our approach can be realized and enforced.

Practical Prevention of XSS in Google Web Toolkit Applications through Coding Discipline and Light-Weight Static Checking

Christoph Kern (Google Switzerland, CH)

Google Web Toolkit is an open-source development framework for AJAX web applications. Client-side application code is developed in Java and compiled into JavaScript for deployment.

Like all web applications, GWT-based apps can be vulnerable to XSS vulnerabilities unless developers take appropriate precautions. We present a practical approach aimed at helping software engineers develop and maintain GWT applications that are free of XSS with a high degree of confidence. We achieve this goal by combining a (from a developer's point of view reasonable and acceptable) coding discipline with light-weight static checking.

Verifying XSS and XSRF protection and web-based two-party secure computation

Florian Kerschbaum (SAP Research, Karlsruhe, DE)

The first part is going to introduce a formal verification of a simple XSS and XSRF protection method. We model the basics of an XSS attack and a simple protection method of checking the referer string on pages with input in the model checker Alloy. One can verify that the model is powerful enough to capture XSS attacks and second that the protection method prevents them.

The second part presents an experimental architecture for running a two-party secure computation as a web application. Offering the secure computation as a web application allows a different business model of selling the service. On the other hand we face the technical challenge of establishing a secure channel through the web server.

Identity Theft Attacks on Social Networks

Engin Kirda (Institut Eurécom, Sophia-Antipolis Cedex, FR)

Social networking sites have been increasingly gaining popularity. Well-known sites such as Facebook have been reporting growth rates as high as 3% per week. Many social networking sites have millions of registered users who use these sites to share photographs, contact long-lost friends, establish new business contacts and to keep in touch. In this talk, I will discuss how easy it would be for a potential attacker to launch automated crawling and identity theft attacks against a number of popular social networking sites in order to gain access to a large volume of personal user information.

Keywords: Social Networking, Security, Identity theft

Static Analysis of JavaScript (and its application to intrusion detection)

Shriram Krishnamurthi (Brown Univ. - Providence, US)

We present a static control-flow analysis for JavaScript programs running in a web browser. Our analysis tackles numerous challenges posed by modern web applications including asynchronous communication, frameworks, and dynamic code generation. We use our analysis to extract a model of expected client behavior as seen from the server, and build an intrusion-prevention proxy for the server: the proxy intercepts client requests and disables those that do not meet the expected behavior. We insert random asynchronous requests to foil mimicry attacks. Finally, we evaluate our technique against several real applications and show that it protects against an attack in a widely-used web application.

Types for JavaScript

Shriram Krishnamurthi (Brown Univ. - Providence, US)

I would be happy to talk about our in-progress, experimental work on building a type system for JavaScript. Having tried for 15 years to build type systems for "scripting" languages, and having largely failed to build anything that we find suitable, we have adopted a new strategy we call Type Enrichment, which defines the mediated co-existence of typed and untyped code. I would be happy to explain what all this means and how we've been doing it.

Ripley: Automatically Securing Distributed Web Applications Through Replicated Execution (40 min)

Benjamin Livshits (Microsoft Research - Redmond, US)

Rich Internet applications are becoming increasingly distributed, as demonstrated by the popularity of AJAX/Web 2.0 applications such as Hotmail, Google Maps, Facebook, and many others. A typical multi-tier AJAX application consists of a server component implemented in Java J2EE, PHP or ASP.NET and a client-side component executing in JavaScript. The resulting application is more performant and responsive because computation is moved closer to the client, and thus avoids unnecessary network round trips for frequent user actions.

However, once a portion of the code is moved to the client, a malicious user can easily subvert the client side of the computation and potentially jeopardize sensitive server state. In this paper we propose RIPLEY, a system that uses replicated execution to automatically preserve the integrity of a distributed computation. RIPLEY replicates a copy of the client-side computation on the trusted server tier. Every client-side event is transferred to the replica of the client for execution. RIPLEY observes results of the computation, both as computed on the client-side and on the server side using the replica of the client-side code. Any discrepancy is flagged as a potential violation of computational integrity. Our evaluation of RIPLEY on five complex and representative AJAX applications suggests that RIPLEY is a promising method for building secure distributed web applications.

Language-Based Isolation of Untrusted Javascript

Sergio Maffeis (Imperial College London, GB)

Web sites that incorporate untrusted content may use browser- or language-based methods to keep such content from maliciously altering pages, stealing sensitive information, or causing other harm.

We study language-based methods for filtering and rewriting JavaScript code, using Yahoo!'s AdSafe and Facebook's FBJS as motivating examples.

We explain the core problems by describing previously unknown vulnerabilities and subtleties, and develop a foundation for improved solutions based on an operational semantics of the full ECMA-262 language.

We also discuss how to apply our analysis to address the JavaScript isolation problems we discovered.

Keywords: Mashup, security, javascript, semantics

The Motivation, Design And Implementation Of Caja

Jasvir Nagra (Google Inc. - Mountain View, US)

Caja is a compiler that provides isolation and safe, efficient communication between html, css and javascript objects on a webpage. Its goal is to provide defensive code (where the container page and other objects are protected from a given object), offensive code (where an object is able to share references with other objects safely), support large number of existing legacy code and tools.

Caja achieves its goal by first converting html and css into javascript. It has two failstop javascript subsets - valija, which is very close to ECMAscript 3.1; and cajita, which is an object capability subset of javascript. Valija code is translated into cajita which is in turn enforced with a static verifier and runtime checks.

Separating the problem into two stages considerably simplifies the problem of understanding and building the system, keeps the TCB small while supporting a large number of legacy javascript applications.

Keywords: Caja, mashup, mutually suspicious, javascript

Lightweight Self-Protecting JavaScript

Phu Phung (Chalmers UT - Göteborg, SE)

This paper introduces a method to control JavaScript execution. The aim is to prevent or modify inappropriate behaviour caused by e.g. malicious injected scripts or poorly designed third-party code. The approach is based on modifying the code so as to make it self-protecting: the protection mechanism (security policy) is embedded into the code itself and intercepts security relevant API calls. The challenges come from the nature of the JavaScript language: any variables in the scope of the program can be redefined, and code can be created and run on-the-fly. This creates potential problems, respectively, for tamper-proofing the protection mechanism, and for ensuring that no security relevant events bypass the protection. Unlike previous approaches to instrument and monitor JavaScript to enforce or adjust behaviour, the solution we propose is lightweight

in that (i) it does not require a modified browser, and (ii) it does not require any run-time parsing and transformation of code (including dynamically generated code). As a result, the method has low run-time overhead compared to other methods satisfying (i), and the lack of need for browser modifications means that the policy can even be applied on the server to mitigate some effects of cross-site scripting bugs. We describe the implementation, and present an abstract formalisation of the basic method. Based on this formalisation we show, as an example, that it can soundly enforce the class of policies known as security automata.

Keywords: JavaScript, Language Based Security, Inlined Reference Monitors, Security Policy

Joint work of: Phung, Phu H.; Sands, David; Chudnov, Andrey

Security by Contract for web applications

Frank Piessens (Katholieke Universiteit Leuven, BE)

Security by Contract (SxC) is an approach for untrusted code security developed in the context of mobile phones. SxC is a variant of Sekar et al.'s Model Carrying Code, where untrusted code comes with a high level model of its security relevant behaviour. In SxC, this model is formalized as a security automaton in the style of Schneider.

An instantiation of SxC needs techniques to check the compliance of an application with such a model, and techniques to check whether a given model is compatible with a given security policy.

This talk will discuss the challenges that need to be addressed to instantiate SxC for web applications.

Keywords: Security by contract

Joint work of: Piessens, Frank; Massacci, Fabio; Desmet, Lieven

Effective Taint Analysis of Web Applications

Marco Pistoia (IBM TJ Watson Research Center - Hawthorne, US)

Taint analysis is a form of information-flow security analysis that tracks values originating from untrusted methods and parameters, in order to establish whether they flow into security-sensitive areas of the application. There is a rapidly growing body of academic research on taint analysis. However, most of the static taint-analysis algorithms that have been proposed to date do not scale to industry-level code, fail to model essential Web-application-code artifacts soundly, and only address very specific types of information flow and a limited set of attack vectors.

We have designed and implemented a static Taint Analysis for Java (TAJ) that meets the requirements of industry-level applications. In terms of scalability, TAJ can analyze applications of virtually any size, as it employs a set of techniques designed to produce useful answers given limited time and space. In terms of scope, TAJ covers a wide variety of attack vectors, and embodies techniques to handle reflective calls, flow through containers, nested taint and other challenges that have been largely unanswered by previous works.

This seminar provides an in-depth description of the algorithms comprising TAJ, evaluates TAJ against production-level benchmarks, and compares it with alternative solutions.

A full paper on this work has been accepted for publications in the Proceedings of the ACM SIGPLAN 2009 Conference on Programming Language Design and Implementation (PLDI 2009), Dublin, Ireland, June 2009.

Joint work of: Omer Tripp, Marco Pistoia, Stephen Fink, Manu Sridharan and Omri Weisman.

Keywords: Web application security, information flow, integrity

Joint work of: Tripp, Omer; Pistoia, Marco; Fink, Stephen; Sridharan, Manu; Weisman, Omri

Modular String-Sensitive Permission Analysis with Demand-Driven Precision

Marco Pistoia (IBM TJ Watson Research Center - Hawthorne, US)

In modern software systems, programs are obtained by dynamically assembling components. This has made it necessary to subject component providers to access-control restrictions. What permissions should be granted to each component? Too few permissions may cause run-time authorization failures, too many constitute a security hole. We have designed and implemented a composite algorithm for precise static permission analysis for Java and the CLR. Unlike previous work, the analysis is modular and fully integrated with a novel slicing-based string analysis that is used to statically compute the string values defining a permission and disambiguate permission propagation paths. The results of our research prototype on production-level Java code support the effectiveness, practicality, and precision of our techniques, and show outstanding improvement over previous work.

Keywords: Access control, static analysis

Joint work of: Geay, Emmanuel; Pistoia, Marco; Tateishi, Takaaki; Ryder, Barbara; Dolby, Julian

Supporting and Securing Programs inside Web Browsers

Charlie Reis (University of Washington, US)

Today's browsers are being placed in the role of operating systems for complex programs from the web. Recent versions of browsers are starting to reflect this new workload, providing better performance and greater robustness to failures. Improving security is an important challenge as well, given the valuable private data that now lives on the web. In this talk, I will discuss how new browser architectures can help protect the user's local computer with sandboxed rendering engines, as in Google Chrome. I will also describe challenges for architecturally isolating a user's web accounts from each other, and ideas about how we might achieve it.

Web Application Anomaly Detection in a Web 2.0 World

Will Robertson (Univ. California - Santa Barbara, US)

During the last decade, web applications have become an extremely popular method of providing a diverse array of services to users.

Unfortunately, web applications have been found to contain large numbers of vulnerabilities, most notably – but in no way limited to – cross-site scripting (XSS) and SQL injection. Consequently, web applications have also become a favored target of cyber-criminals, who leverage web application vulnerabilities to steal sensitive information or host malicious software. In the absence of mitigating improvements in security, this trend is expected to continue as web applications increase in complexity and, accordingly, in attack surface.

In this talk, we approach the problem of securing web applications from two complementary angles. First, we discuss the state-of-the-art in web application anomaly detection, an effective black-box technique to mitigate vulnerabilities in existing web applications, and how anomaly-based WAFs can adapt to the increasingly distributed nature of modern web applications. Then, we present approaches for improving the security of web applications from the design stage, by focusing on the languages and tools used to construct web applications.

Keywords: Web applications, anomaly detection

Keynote: Towards tracking information flow and release in web applications

Andrei Sabelfeld (Chalmers UT - Göteborg, SE)

Information-flow tracking in web applications is an attractive, and increasingly popular, alternative for enforcing end-to-end confidentiality and integrity.

However, there is a gap between formal, mostly static, approaches – that lack support for dynamic language features – and practical, mostly dynamic, approaches – that lack soundness arguments. This talk discusses some steps towards bridging this gap, focusing on information release on the policy side and on combinations of static and dynamic techniques on the enforcement side.

The talk is based on joint papers with Aslan Askarov, Andrey Chudnov, and Alejandro Russo.

Browser Based Protocols - Towards a Better Integration of TLS

Joerg Schwenk (Ruhr-Universität Bochum, DE)

Following an analysis of generic weaknesses of today's single sign-on protocols (SAML, MS Cardsapce, Liberty Alliance etc.), we propose three different methods for a better integration of TLS into these protocols. The three proposed variants are provably secure.

Keywords: SSL, TLS, SOP

Cross-tier, Label-based Security Enforcement for Web Applications

Nikhil Swamy (Microsoft Research - Redmond, US)

This paper presents SELinks, a programming language focused on building secure multi-tier web applications. SELinks provides a uniform programming model, in the style of LINQ and Ruby on Rails, with language syntax for accessing objects residing either on the database or at the server. Object-level security policies are expressed as labels, which are themselves fully-customizable, first-class objects. Access to labeled data is mediated via trusted, user-provided policy enforcement functions.

SELinks has two novel features that ensure security policies are enforced correctly and efficiently. First, SELinks uses a novel type system that allows a protected object's type to refer to its protecting label. This way, the type system can check that labeled data is never accessed directly by the program without first passing through the appropriate policy enforcement function. Second, SELinks compiles policy enforcement code to database-resident user-defined functions, which can be called directly during query processing. Database-side checking avoids transferring data to the server needlessly, while still allowing policies to be expressed in a customizable and portable manner.

Our experience with two sizeable web applications, a model health-care database and a secure wiki with fine-grained security policies, indicates that cross-tier policy enforcement in SELinks is flexible, relatively easy to use, and, when compared to a single-tier approach, improves throughput by nearly an order of magnitude.

Joint work of: Corcoran, Brian; Hicks, Michael; Swamy, Nikhil

The significance and costs of supporting hypertext isolation in existing browsers

V. N. Venkatakrishnan (Univ. of Illinois - Chicago, US)

These are times of tremendous progress for the World Wide Web which has evolved from being a HTML document delivery system to a complex application platform. Yet, despite these advances, today's web standards don't support a simple, pragmatic isolation facility that can instruct a browser to demarcate certain regions of a document as untrusted.

Such a facility will further the goals of defending the web infrastructure from the ongoing spate of cross-site scripting (XSS) attacks.

The absence of such a facility has forced web developers to employ content filtering as the primary defense against XSS. Unfortunately, precise content filtering is made difficult by anomalous web browser parsing behaviors, which are often used as vectors for successful XSS attacks.

We describe an experimental approach and tool called BluePrint that is designed to isolate content in widely deployed existing web browsers, despite anomalous browser behavior. We highlight the strengths of this approach as well as its costs.

Joint work with Mike Ter Louw and Prithvi Bisht

Keywords: XSS, script injection, cross-site scripting, isolation, untrusted content, filtering

Is your web page malicious or you're just happy to see me?

Giovanni Vigna (Univ. California - Santa Barbara, US)

JavaScript is a browser scripting language that allows developers to create sophisticated client-side interfaces for web applications. However, JavaScript code is also used to carry out attacks against the user's browser and its extensions. These attacks usually result in the download of additional malware that takes complete control of the victim's platform, and are therefore called "drive-by downloads." Unfortunately, the dynamic nature of the JavaScript language and its tight integration with the browser make it difficult to detect and block malicious JavaScript code.

In this talk, we review current web security research carried out at UC Santa Barbara, focusing on an approach to the detection and analysis of malicious JavaScript code. This approach combines anomaly detection with emulation to automatically identify malicious JavaScript code and support its analysis.

We developed a system that uses a number of features and machine-learning techniques to establish the characteristics of normal JavaScript code. Then, during detection, the system is able to identify anomalous JavaScript code by emulating its behavior and comparing it to the established profiles. In addition to identifying malicious code, the system is able to support the analysis of obfuscated code and generate detection signatures for signature-based systems.

Keywords: Web Security, JavaScript, Malware

A Survey Study on Security in Application Development

John Wilander (Omegapoint AB - Stockholm, SE)

We've been teaching a professional course on web application security the last two years. 12 course instances and about 120 participating developers from almost all sectors of IT – internet banking, internet gaming, health care systems, e-learning, logistiscs etc. Apart from all the interesting discussions from these courses we've been carrying out a survey collecting data on how security is prioritized and sold in projects and businesses. The results are yet unpublished but interesting. They give a picture of how security fits in the web application picture today.

Keywords: Application security, survey

A Cryptographic Compiler for Information-Flow Security

Cedric Fournet (Microsoft Research UK - Cambridge, GB)

We relate two notions of security: one simple and abstract, based on information flows in distributed programs, the other more concrete, based on cryptography.

In language-based security, confidentiality and integrity policies specify the permitted flows of information between parts of a system with different levels of trust. These policies enable a simple treatment of security, but their enforcement is delicate.

We consider cryptographic enforcement mechanisms for distributed programs with untrusted components. Such programs may represent, for instance, distributed systems connected by some untrusted network. We develop a compiler from a small imperative language with locality and security annotations down to cryptographic implementations in F#. In source programs, security depends on a policy for reading and writing the shared variables. In their implementations, shared memory is unprotected, and security depends instead on encryption and signing.

We show that our compiler preserves all information-flow properties, under standard cryptographic assumptions: an adversary that interacts with the trusted components of our code and entirely controls its untrusted components gains illegal information only with negligible probability.

Joint work of: Fournet, Cédric; le Guernic, Gurvan; Rezk, Tamara