USENIX Conference on Web Application Development (WebApps '10)

June 23–24, 2010
Boston, MA

WebApps '10 shared the opening session and Keynote Address with the 2010 USENIX Annual Technical Conference: please see p. 63 for the report on that session.

JUNE 23, 10:30 A.M.-NOON

Summarized by Rik Farrow (rik@usenix.org)

- ***Separating Web Applications from User Data Storage with BStore***
  Ramesh Chandra, Priya Gupta, and Nickolai Zeldovich, MIT CSAIL

  Won Best Paper Award!

Ramesh Chandra pointed out that while some apps (e.g., Google mail) rely on a single online store, other applications require getting data from one site and doing something with it using a different site. Chandra used an example where a photo editing site needs to get on Flickr to gain access to a photo.

Their solution is BStore, moving data storage within the browser. BStore provides a single, simple (four call) API for storing data and is implemented in JavaScript. Back-end storage can be in the cloud (S3) or local. BStore provides security through tagging data. Only the principal or the user can tag data for sharing with another application. Tagging is used for more than access control, as files may be logically grouped using tags.

They ported several Web apps with a minimal amount of effort, adding about 5% to the size of the app. BSTORE uses postMessage for RPC within the browser, and currently only works in Firefox and Chrome. Performance is similar to using XMLHttpRequest.

Someone asked if BSTORE could use the browser cache for local store, and Chandra pointed out that the browser cache uses the same origin policy for security, so cannot be used. Dan Peek asked if they had tried to extend this to multiuser control, and Chandra said that tagging supports this. Also, they include version numbers with files, and that would prevent a file from being overwritten. Ben Livshits asked if they had thought about XSS or code injection. Chandra replied that each piece of code runs in its own window and process space, and they use an RPC built on top of post-Message.

- ***AjaxTracker: Active Measurement System for High-Fidelity Characterization of AJAX Applications***
  Myungjin Lee and Ramana Rao Kompella, Purdue University; Sumeet Singh, Cisco Systems

Myungjin Lee pointed out that applications are being migrated to the cloud using AJAX (Asynchronous JavaScript and XML). AJAX supports autonomous action on the client side, and this poses problems for measuring performance. The authors' goal is to characterize full application sessions, including flows/servers, request/response distributions, and inter-request time, as well as local operations such as dragging an icon into the trash or clicking on a map.

Their approach involves capturing X events using a program as well as capturing network traffic at the client side using tcpdump. They compared their approach, which collects both X events and network traffic, to a network-only approach, and tested both at four different bandwidth conditions. You can download the tool here: http://www.cs.purdue.edu/synlab/ajaxtracker/.

James Mickens wondered if they were using the DOM object ID in XML files. Lee pointed out that they are using an object ID as defined by users, since they have access to the X event that generated the actual event in the Web browser. Someone else asked how well their approach worked in richly dependent apps, and Lee said that there were limitations to their approach. John Ousterhout wondered if they had overfitted their data as shown in a graph in their paper. Lee said that they had picked one of four bandwidths to characterize the dominant bandwidth based on network traces, but did not use a sophisticated model to fit the curve.

- ***JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications***
  Paruj Ratanaworabhan, Kasetsart University; Benjamin Livshits and Benjamin G. Zorn, Microsoft Research

Ben Zorn stated that benchmarks, such as SunSpider and V8, do a poor job of capturing real-world Web application performance. JSMeter is a Microsoft Research project that instruments Internet Explorer, but is only currently available within Microsoft. The advantage of being able to instrument the browser itself, rather than running benchmarks on top of the browser, is that JSMeter can measure three areas of JavaScript runtime behavior: functions and code; heap-allocated objects and data; and events and handlers. Zorn pointed out that JavaScript is event-driven, and benchmarks run tight loops, allocate little data on the heap, and have few events.

Zorn displayed graphs showing the different heap behavior of several Web apps, including ones from Google, Amazon, eBay, The Economist, and Microsoft. JSMeter makes it easy to see different programming styles, such as how the heaps grow continuously with Gmail, while eBay wipes the heap clean with each new page. JSMeter also clearly shows that Google caches functions across reloads to speed performance. In Bing, you stay on the same page and the heap grows a lot over time, where Google focuses on using less memory.

Zorn went on to point out more shortcomings of V8 and SunSpider that make them poor benchmarks (and one that makes IE look much slower than Firefox and Chrome). He suggested that people visit their Web site and read their tech report: http://research.microsoft.com/en-us/projects/jsmeter/.

Does JSMeter deal with anonymous functions? JSMeter does monitor them, uses hashes to identify functions, and counts each invocation. Have they have noticed any changes in the way JavaScript is being used? JavaScript use is indeed a moving target and they expect its use will keep changing over time. Armando Fox mentioned that there are a number of code frameworks that generate JavaScript and wondered what framework writers could do to make the world better. Zorn said that the amount of code downloaded affected performance a lot, so why download code that is never going to be executed? He suggested staging code and lazily downloading code as needed. Someone else asked about how Web apps could better use browsers, and Zorn suggested that better tools need to be available. While JSMeter was an internship project in IE8, IE9 will provide tools that will be available to developers.

JUNE 23, 1:30 P.M.–3:00 P.M.

Summarized by Aiman Erbad (aerbad@cs.ubc.ca)

- ***JSZap: Compressing JavaScript Code***
  Martin Burtscher, University of Texas at Austin; Benjamin Livshits and Benjamin G. Zorn, Microsoft Research; Gaurav Sinha, IIT Kanpur

Ben Livshits from Microsoft Research introduced JSZap, a technique to efficiently compress JavaScript code based on Abstract Syntax Tree (AST) representation. Livshits claimed that 85% of downloads for rich Web applications is JavaScript code. An application like Bing Maps has more than

70,000 lines of code, which translates to more than 1MB of network traffic. This trend adheres to the conventional wisdom of moving more code to the client for responsiveness. However, execution cannot start without the code, so their approach aims to make the wire format lighter.

The main idea in JSZap is to send JavaScript code using an AST-based representation instead of sending the raw code and generating the AST at the client. JSZap uses Gzip as second-level compressor, following the wisdom that says if you can't beat them, join them. JSZap splits the JavaScript compression into three streams: productions, identifiers, and literals. JSZap proposes a number of techniques for compressing productions, such as frequency-based production renaming, differential encoding, predictable production elimination, and tree-based prediction by partial match. To compress identifiers, JSZap used a symbol table instead of a flat stream. JSZap achieved an average 10% improvement across the board. Preliminary results were verified using benchmarks with different sizes, application types, and a mix of machine-generated and hand-crafted code. Livshits claimed that AST-based representations can have broader applicability in caching incremental updates, unblocking HTML parser, correctness, and security checks.

Dan Peek from Facebook asked why the AST is the right abstraction level. Livshits explained that the language is standardized while lower-level representations (byte codes) are specific to browsers, and it is a small step to get the AST. Wanchun Li from Georgia Tech asked about performance trade-off and how much time it takes to parse the AST. Livshits explained that is future work and that they are not concerned about time on the server side. Jon Howell from MSR asked about other places where Gzip fails and the JSZap algorithm succeeds. Livshits mentioned tree-based predictive matching where structure matters, and places where the window of compression is small in Gzip. In a follow-up, Dan Peek asked about the ability to tweak Gzip settings. Livshits said they used the same settings in the browser, because some powerful compression algorithms are not supported by browser implementations.

- ***Leveraging Cognitive Factors in Securing WWW with CAPTCHA***
  Amalia Rusu and Rebecca Docimo, Fairfield University; Adrian Rusu, Rowan University

Amalia Rusu explained how they mapped their knowledge in the field of handwriting recognition and image analysis to improve Web security and CAPTCHA generation algorithms. CAPTCHA challenges are generated automatically on the fly without dependence on a database and are public, so the algorithm/code to build CAPTCHAs should be publicly accessible. They are used widely to differentiate between humans and machines and secure Web sites. CAPTCHAs are made harder for machines but this sometimes comes at the expense of human usability. The authors' work can improve the user experience of CAPTCHA while still maintaining or even improving the security, which Rusu refers to as usable security. The main contribution is to introduce structures and transformations in CAPTCHA challenges based on cognitive factors so that only machine recognition is hindered.

Some basic ideas which this work revolves around are: handwritten CAPTCHAs have unique characters that increase readability while making it harder for machine recognition, and tree-based structures are the basic structure for information visualizations so trees are familiar to people. They also leverage principles of cognitive psychology (such as symmetry, proximity, similarity, etc.) and geon theory in pattern recognition to transform handwriting. These could also be extended to the tree structure as well. So the CAPTCHA they envision is a tree structure with each node representing a handwritten word. To solve the challenge, the human/machine will answer a question such as what are the words connected with an edge, or give me all the pairs in this tree. So in order to pass the CAPTCHA you need to recognize the words, segment the information, and interpret it, which is where machines fail. In a preliminary evaluation, they tested with both machines and users with three state-of-the-art recognizers. Machines had a low recognition rate of 1–5%. In usability testing, the recognition rate for humans is more than 80%, and users gave these CAPTCHAs a difficulty level of 2 (5 is most difficult). These results show the feasibility of the handwritten tree-based CAPTCHAs.

James Mickens from MSR asked how you would attack this system. Rusu explained that the pre-processing phase before segmentation is used to attack CAPTCHAs. They created some custom attacks based on techniques such as occlusion. These attacks can make recognition for other transformations worse, so we cannot combine attacks for different kind of transformations. Dan Peek from Facebook asked if these techniques can enhance recognition neutral to the language. Rusu explained that, for their approach to work, they need to generate everything on the fly. So if they have a tool to generate the words based on IP addresses using the language common in the specific region, this might improve the CAPTCHA generation and make it language-neutral.

- ***Gulfstream: Staged Static Analysis for Streaming JavaScript Applications***
  Salvatore Guarnieri, University of Washington; Benjamin Livshits, Microsoft Research

Salvatore Guarnieri introduced Gulfstream, which helps with safe inclusion of third-party source code using staged static analysis. Two ways to have safe inclusion are run-time enforcement, which detects the behavior of code at run-time and prevents the damage, and static analysis, which analyzes the code before it is sent and, if it detects that something bad could possibly happen, prevents the page from being sent to the user. Static techniques are the focus here, and they usually require full source code. But JavaScript is usually streamed as you interact with the page. Updates are relatively small compared to the static page you are getting from the server initially. Gulfstream performs offline static analysis on the static page in the server and online analysis of the small updates in the client.

To understand the behavior of the program they use queries. For example, what does variable f refer to or is alert() ever called? To get the information, they use points-to analysis, which tells them what memory location f points to. They have two techniques for the points-to analysis: (1) datalog with an engine based on binary decision diagrams, which is fast for large programs and highly tuned but has a large start-up cost and is difficult to integrate in browsers; (2) graph-based flow analysis, which has a small start time but does not scale very well. This trade-off is reasonable when the updates are small. Gulfstream starts by normalizing the JavaScript, building the flow graph, and serializing the graph. Then Gulfstream performs points-to analysis and uses the results along with the flow graph to answer the queries. In evaluation they asked if Gulfstream is faster than non-staged (static) analysis using a representative benchmark. They simulated analysis on a diverse set of devices (PC and mobile) with different CPUs and network speeds. The main results are that slow devices benefit the most from Gulfstream because the analysis is CPU-intensive and is harder to perform in slow devices. A slow network can negate the benefits of the staged analysis if the CPU is fast enough to finish the full analysis before the staged analysis results finish transferring over the network. Large page updates don't benefit from Gulfstream due to analysis overhead (> 60kb).

Jon Howell from MSR asked why answering queries is slow—taking tens of seconds for large devices (which means minutes for small devices)—for both techniques. Guarnieri mentioned that they need to optimize the approach to yield timely results. Godmar Back from Virginia Tech asked whether the fact that all these applications were written to be modular can be exploited in the analysis. Guarnieri said that optimizations based on modularity might have malicious applications, so they were not considered.

JUNE 23, 3:30 P.M.–5:00 P.M.: WORK-IN-PROGRESS REPORTS (WIPS) AND POSTER PROMOS

Summarized by Pradeep Teregowda (pbt105@psu.edu)

- ***Detecting User-Visible Failures in AJAX Web Applications by Analyzing Users' Interaction Data***
  Wanchun Li, Georgia Institute of Technology

Wanchun Li pointed out that developers and administrators are not aware of user failures and issues with interfaces, which include AJAX and input elements in Web applications. He presented a method for detecting such failures of interfaces by identifying interactions such as repeated steps or missteps from user interaction data. He presented initial results for an AJAX application.

- ***Doha: Real-Time Support for Event-driven Web Applications***
  Aiman Erbad and Charles Krasic, University of British Columbia

Aiman Erbad presented Doha in the context of HTML5. The proposed HTML5 standards provide powerful tools for execution on the client side. Doha allows developers to effectively leverage workers on the client side. It does this by managing resources and coordination of Web workers so that event-driven Web applications can be supported in real time.

- ***MyEbay Research Web Service: An Application to Practice the Web Service***
  Shaun-inn Wu and Jian Huang, California State University San Marcos

Shaun-inn Wu presented an approach for involving students in building Web applications. MyEbay was built for students involved in undergraduate coursework. This service allows users to explore the depth of the Web application features and interfaces. MyEbay has been successful in involving students in Web application projects.

- ***A Seamless Online Application and Advising System for a Degree Program***
  Shaun-inn Wu, California State University San Marcos

Scheduling university courses presents several challenges, especially coordinating student choices and courses. Shaun-inn Wu and his group are building an online advice system for degree programs offered by the university. In scheduling courses, students are advised to contact other students following the same course pattern.

- ***Fine-Grained Isolation in Web Browsers Using Script Spaces***
  Amarjyoti Deka and Godmar Back, Virginia Tech

Godmar Back points out that the current state of browser isolation cannot support complex applications even with Google Chrome. Pages contain widgets executing third-party code; client-side extensions running content scripts on Web pages are particular examples. Script Spaces developed by his team provide an isolated execution environment for all parts of the Web page, separate namespaces, and access to CPU and memory. Script Spaces allows for fail-safe loading of pages without lockups. A prototype based on Firefox 3.0 has been developed.

- ***SaaS and Cloud Computing in Undergraduate Software Education***
  Armando Fox, University of California, Berkeley

Armando Fox discussed the use of cloud computing for a SaaS course. Adopting cloud computing saved resources and made it easier to finish assignments. The focus on SaaS using agile development also resulted in high-quality working prototypes by the end of the semester. Dedicated server resources would have consumed entire data centers and could now be used only when needed. The lifetime of projects also lasted more than the length of the course.

- ***xHunter: Tracking XSS Attacks on the Net***
  Elias Athanasopoulos, FORTH-ICS

Elias Athanasopoulos presented xHunter, a tool which allows researchers to track and analyze patterns of XSS attacks. Cross-site scripting (XSS) attacks are a significant security issue for Web applications. xHunter processes URLs, trying to build syntax trees using the provided URL. It marks those as suspicious that generate high-depth JavaScript syntax trees. Elias requested that users submit URLs to xHunter.

- ***A Case for PIQL: Performance Insightful Query Language***
  Michael Armbrust, Nick Lanham, Stephen Tu, Armando Fox, Michael Franklin, and David Patterson, University of California, Berkeley

Stephen Tu et al. found that many applications are moving away from traditional relational databases to key/value pair stores for scalability and performance guarantees. They propose PIQL, an expressive language for key/value stores. An aspect of PIQL adoption was challenging developers to think beyond the standard database design, forcing them to handle performance issues at design time. Applications developed with PIQL were demonstrated.

- ***A Performance-Monitoring Framework for Multi-Tier Web Applications***
  Chris McCoy, Northeastern University and Smarter Travel Media; Ryan Miller, Smarter Travel Media

Chris McCoy and Ryan Miller found it was a challenge monitoring profile information across multiple tiers, especially when they consist of heterogeneous systems and applications. The tool they developed allows administrators to monitor performance across the cluster with multiple tiers supporting better granularity than those provided by current tools. The data collected from the systems is displayed in a user-friendly interface.

- ***A Combined Autonomic and On-Demand Approach to Configuring Virtualized Development Environments***
  Ryan Miller, Smarter Travel Media; Matt Warren, Northeastern University and Smarter Travel Media

Ryan Miller said that developers in virtualized environments can save time and improve efficiency by adopting the proposed autonomic and on-demand configuration system. Such an adoption would reduce development time by enabling administrators to quickly deploy the virtualized environment without being constrained by complex configuration steps.

- ***Taking Control Away from Users . . . in a Good Way***
  Jon Howell, Microsoft Research

Jon Howell raised the provocative question of how much control should be vested with users. The contention was that, while simple choices are easy for the user to understand and answer, questions which are more involved tend to confuse the user. Such a model already exists in data centers and can be extended to desktops and Web browsers. Architectural changes and user interface issues were discussed.

JUNE 24, 9:00 A.M.–10:00 A.M.: INVITED TALK

- ***Gmail: Past, Present, and Future***
  Adam de Boor, Staff Software Engineer, Google

Summarized by Pradeep Chalise (pradeepchalise@gmail.com)

Adam de Boor started by outlining his talk: where Gmail came from, where it is now, what the Gmail team learned, and their plans for the future. Today, Gmail has expanded beyond mail to include video chat, voice chat, Gmail labs, and lots more. For Gmail to arrive at this stage, it had to undergo a lot of technical modifications since its launch back in 2004. At that time, Gmail was a slick Webmail application using AJAX, and it has continued to develop ever since. De Boor said that to fulfill Gmail's promise to its users, it has evolved into a single complex application supporting a lot of diverse functionalities.

De Boor wanted to provide some higher-level concepts regarding the macro architecture of the communication between the Gmail client and server, how the business logic is invoked and is used to reply to the client for chat, mail, etc. He explained some details of implementation of the Gmail client module and the Gmail server. Then he provided an introduction to mods, which are named code segments enabled on a per-user basis. He then combined the concept of modules and mods to give us the clear idea of what is possible and what is served. He also compared Gmail with Microsoft Windows features like video, chat, messaging, etc.

Gmail's future plans include dealing with service-oriented architecture, trying to make users feel that Gmail is like a desktop application. Gmail developers are now planning to use HTML5, which is exciting because it reduces the DOM by 30% and initial load time by 12%.

Finally, de Boor informed the audience about the lessons learned by the Gmail team through its experience: testing is vital, type-checking is important, it's beneficial to instrument everything and codify lessons learned in sanity tests.

Since Google heavily depends on JavaScript, are there any plans on the horizon for using any other language? Lots of programmers understand JavaScript and do simple things with it, so they will continue using it. There is no single application that doesn't use JavaScript right now.

JUNE 24, 10:30 A.M.–NOON

Summarized by Pradeep Chalise (pradeepchalise@gmail.com)

- ***Managing State for Ajax-Driven Web Components***
  John Ousterhout and Eric Stratmann, Stanford University

John Ousterhout started his speech by giving an introduction to the basics of Ajax and why and how Ajax-driven components cause problems. To deal with the problem that Ajax complicates Web applications is the idea of using reusable Ajax components that hide complexity. But this solution creates the problem of maintaining the state of the browser across Ajax requests. To solve this problem, his team proposes two possible solutions: using *reminders*, which store state on the browser, and using *page properties,* which store state on the server. He also emphasized that neither of these solutions is perfect.

Reminders are the collection of name-value pairs similar to the View State mechanism in ASP.net, except more granular and embedded in the page by server-side components. Reminders, however, have security implications since they store internal server state (potentially sensitive), requiring the use of encryption.

Page properties, which are name-value pairs specific to a page, are stored in session, are created during the initial page rendering, and are accessible/modifiable during Ajax requests. Page properties have no security issues but include extra overhead for saving properties. The biggest problem with Properties is garbage collection.

To demonstrate the work his team has done, Ousterhout provided trace-driven simulations showing a graph with broken pages per one thousand views in the Y axis and the Least Recently Used list length (per-user) in the X axis. He concluded that managing Web application state is hard, and neither reminders nor page properties are ideal but that garbage collection problems are less serious than security problems and that, overall, page properties are better.

- ***SVC: Selector-based View Composition for Web Frameworks***
  William P. Zeller and Edward W. Felten, Princeton University

William Zeller started his talk by giving an introduction to Selector-based View Composition (SVC) as a new programming style for Web application development. Developing a framework like SVC targeted for supporting both JavaScript and non-JavaScript browsers is worthwhile because there are still some browsers without JavaScript. He showed us how requests travel in a Model View Controller (MVC) architecture pattern from client to the controller and browser first without and then with SVC. Further, he went into details of the SVC-server side API. APIs are used to compose views together, and (CSS) selectors are used to identify the point of composition.

SVC allows developers to write/view/update code only once. He also said that the reasons for using selectors are that they are familiar to developers, are more common in JS frameworks, and are used in the HTML5 API. To give an idea of what SVC looks like, he provided some actual code samples, showing that they implemented nine actions with the possibility of easy extension. He also made clear how non-DOM actions are added to SVC.

Finally, he talked about the alternatives to SVC such as GWT, Cappuccino, and RIS, but none of them supports non-Ajax browsers, which are supported by the SVC model. He also said that extension of SVC could be done in terms of additional language support (Python instead of PHP) and additional client-side libraries. He concluded that SVC provides automatic progressive enhancement and compatibility with older browsers.

Further information about this paper can be found at http://svc.from.bz.

- ***Silo: Exploiting JavaScript and DOM Storage for Faster Page Loads***
  James Mickens, Microsoft Research

James Mickens started by discussing the process of interaction between a client and a server and how large round-trip times (RTT) are harmful and increase page load time. There are two general approaches to decreasing the RTT: (1) reducing Cascading Style Sheets and JavaScript in a page, but nobody will do that because it would decrease the page's fanciness; (2) inlining JavaScript and CSS, but doing so would make the browser cache useless.

As a solution to this problem, he introduced Silo, a system that leverages JavaScript and DOM storage to reduce both the number of HTTP requests and the bandwidth required to construct a page. Mickens evaluated the Silo protocol as having the advantages of being able to fetch an arbitrary number of JSS/CSS in two RTTs, with caching being restored and working on unmodified browsers. Silo exploits four key features: read/write, key+value, asynchronously fetching Web data, and overwriting a page's data.

Mickens said that slow pages cause anger and depression. To avoid this, we either have to reduce the number of objects or inline everything. Both options have their own problems. But Silo is an appropriate solution since it uses JavaScript and DOM storage to aggressively inline HTML, JS, and CSS and uses Cache with DOM storage instead of regular browser cache.

Someone asked if Silo is helpful for all kinds of Web sites, or are there any situations where you lose some functionality by using Silo? Wickens responded that delays may vary depending on how the Web site is designed.

JUNE 24, 1:30 P.M.–3:00 P.M.

Summarized by Thomas Moyer (tmmoyer@cse.psu.edu)

- ***Pixaxe: A Declarative, Client-Focused Web Application Framework***
  Rob King, TippingPoint DVLabs

Rob King presented Pixaxe, a client-focused Web application framework, supporting the model-view-controller (MVC) design pattern. The framework is designed for building Web interfaces with legacy code in mind. Pixaxe is built from several components, with each component being stand-alone. King presented the general structure of an example Pixaxe application and described each of the underlying components and how they work together to form the overall framework.

One of the main features of Pixaxe is that all of the processing is done on the client. The only operation the server is required to support is the ability to serve static files. The views developed for Pixaxe are valid XHTML, meaning the application developer can leverage existing knowledge of XSL transformations and XSLT macros. The logic in the views is written in the form of Jenner expressions. Jenner is one of the three components for Pixaxe. Furthermore, the Jenner component is a superset of the ECMAScript expression language (Esel), the second component of the framework. The final component of the framework is the parser/combinator library, named Kouprey. Kouprey is written in ECMAScript, and runs in all major browsers supporting JavaScript. King finished by highlighting the availability of the code at http://www.deadpixi.com and taking questions.

To illustrate the simplicity of the framework, King presented a complex example that pulled log files from a server and presented them to the client. The client was responsible for all of the rendering and processing, with the server only providing the code and data. The client stores all of the data in the model, which is then used by the view to generate the final output seen by the client.

One audience member wondered how complex an application had to become before Pixaxe was no longer a good framework to choose. King responded that the best target for Pixaxe was a single model with a single view and that more complex applications would probably be suited to other frameworks.

- ***Featherweight Firefox: Formalizing the Core of a Web Browser***
  Aaron Bohannon and Benjamin C. Pierce, University of Pennsylvania

Aaron Bohannon began the talk with a series of questions related to the behavior of browsers. The first question related to properties of the DOM, specifically the ownerDocument property and how the value is set. The second question related to altering the DOM to force script re-execution, noting that it is not possible to re-execute a script once it has been run. The final question discussed how event handlers can obtain access to the window that received the event, noting that the asynchronous nature of event handlers made the answer to this question unclear. Bohannon used these three questions to drive home a simple point: we don't fully understand all of the complex components of today's browsers and how they interact.

Featherweight Firefox is a model of the core functionality of the browser. Bohannon described the larger goal of this body of work to be a formal understanding of the security policies of browsers and how the enforcement happens. A formal model of the core functionality is required before formal security proofs can be presented. The model presented in the talk does not include any security features of the browser, such that the model can serve as a basis for modeling both current and proposed security features.

Bohannon highlighted several of the key features of the model, including the modeling of multiple windows and pages, mutable DOM trees, user inputs, event handlers, cookies, and network operations. Currently, the model does not have support for history, HTTP error codes and redirects, timeouts, or JavaScript and file URL handlers. Some of these features would require modeling the current security features of the browser, and are left out as a result.

The first questioner asked who the target audience of the model was. Bohannon sees the target audience as browser developers and researchers. Browser developers can use the model to gain insight into the browser, and researchers can use the model to develop new security mechanisms for browsers. What larger lessons were learned in doing this modeling? That browser behavior was found to be highly non-deterministic and standardizing the behavior would make things simpler. Why did the authors use small-step semantics? Small-step was chosen since it provides the most detailed information about the browser's internal workings.

- ***DBTaint: Cross-Application Information Flow Tracking via Databases***
  Benjamin Davis and Hao Chen, University of California, Davis

Benjamin Davis presented DBTaint, a system for providing taint tracking in Web applications. Davis began by describing current solutions for taint tracking and how they are insufficient. For example, systemwide tracking typically works at the process level, leading to marking all Web application processes as tainted as soon as they process any user input. The next approach was to examine solutions that perform taint tracking within a single process, such as Perl's taint mode. Davis argued that such systems don't allow for tracking information flow between the various applications that comprise a Web application.

DBTaint is a system that provides information flow tracking between the Web application and the database storing the data. DBTaint relies on the taint tracking systems present in languages like Perl and Ruby, as well as work being done to add taint tracking to Java and PHP. These systems allow the Web application to mark and track information as it enters the application and as it is processed. The database schema for an application is modified slightly to track the taint value of each value in the database. The prototype relies on the composite datatypes provided by PostgreSQL. The database interface is modified to work with these composite datatypes. The current prototype provides support for parametrized queries as well as queries constructed on the fly. The evaluation showed that the overhead was roughly 10%.

Does the system work with queries that are built on-the-fly, and not as parametrized queries? The current system does handle this; Davis provided a quick example showing that the taint tracking would require knowing which parts of the SQL statement were tainted. The second question related to handling implicit flows in the application. Davis responded that the current system only handles explicit flows. Had DBTaint led to the discovery of any flaws in the applications used in the evaluation? They did not discover any new flaws, but discussed several possible uses of DBTaint beyond information flow tracking, such as regression testing.

JUNE 24, 3:30 P.M.–4:30 P.M.

Summarized by Thomas Moyer (tmmoyer@cse.psu.edu)

- ***xJS: Practical XSS Prevention for Web Application Development***
  Elias Athanasopoulos, Vasilis Pappas, Antonis Krithinakis, Spyros Ligouras, and Evangelos P. Markatos, Institute of Computer Science, Foundation for Research and Technology—Hellas; Thomas Karagiannis, Microsoft Research, Cambridge

Elias Athanasopoulos opened the talk with justification for another anti-cross-site scripting (XSS) framework. He argued that each of today's frameworks failed in certain ways, including not being developer-friendly, unacceptable overhead, not being backwards-compatible, and not being DOM-independent. Their solution, xJS, addresses each of these issues and can defeat most XSS attacks. Athanasopoulos then introduced a new type of XSS attack called "return-to-JavaScript" attacks. He gave some examples of these attacks and showed how current techniques failed to prevent them.

xJS is a solution based on instruction-set randomization. The legitimate JavaScript in each page is passed through an isolation operator on the server that encodes the JavaScript. In order to decode the JavaScript on the client, the key is passed to the client. In their implementation, the isolation operator was the XOR operation followed by Base64 encoding. This encoded blob is then inserted where the script code would normally be and the client must run the isolation operator before being able to execute the script. Any injected code will be injected as plaintext and become muddled and not be understood by the JavaScript interpreter.

What JavaScript was protected in their implementation? The current implementation only encoded stand-alone JavaScript files, script tags in static files, and event handler code (e.g., onclick, onload, etc.) in static files. Handling dynamically generated files is future work.

- ***SeerSuite: Developing a Scalable and Reliable Application Framework for Building Digital Libraries by Crawling the Web***
  Pradeep B. Teregowda, Pennsylvania State University; Isaac G. Councill, Google; Juan Pablo Fernández R., Madian Kasbha, Shuyi Zheng, and C. Lee Giles, Pennsylvania State University

Pradeep Teregowda began with a description of SeerSuite and what services currently utilized SeerSuite. The SeerSuite framework is used to build digital libraries in an automated fashion. SeerSuite, unlike other digital libraries, does not rely solely on user submissions. Sites like CiteSeerX and ChemXSeer are currently available as examples of digital libraries built by SeerSuite. Scalability and reliability are two of the main design goals of the SeerSuite framework.

The architecture of SeerSuite was described, with a focus on how the framework automatically crawls the Web and builds a digital library. The Web crawler begins with a set of seed sites that it scans for links and documents. When documents are found, another component within the framework converts the document to plaintext and extracts the necessary information. The document metadata that was extracted is inserted into the database, adding the content to the digital library.

One audience member asked about semantic information provided by third parties. Teregowda responded that federation of services allows third parties to provide services and access available data.