ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-5, 1543 – 1548

SECURE PLATFORM FOR WEB USERS BASED ON BROWSER OPERATING SYSTEM ARCHITECTURE

Satish Kumar Ray[1], Mr. Toran Verma[2], Dr. (Mrs.) Sipi Dubey[3]

[1]*M.Tech. (C.T.) Scholar, Dept. of C.S.E., RCET Bhilai/C.S.V.T.U. Bhilai, C.G. India,* ***satishray15@gmail.com***
[2]*Reader Dept. of C.S.E., RCET Bhilai, C.G. India,* ***vermatoran24@gmail.com***
[3]*Dean (R&D), RCET Bhilai, C.G. India,* ***drsipidubey@gmail.com***

Abstract

This paper will demonstrate that a secure, user friendly platform can be implemented using on Browser Operating System Architecture. This platform protects host machine from various attack like phishing attack, drive by download attack, cross site scripting attack. The BOS architecture treats Web applications as first class objects that users explicitly install and manage, giving them explicit knowledge about and control over downloaded content and code. The BOS runs the client-side component of each Web application (e.g., on-line banking, Web mail) in its own virtual machine, provides strong isolation between Web services and the user's local resources. The BOS lets Web publishers limit the scope of their Web applications by specifying which URLs and other resources their browsers are allowed to access. This limits the harm that can be caused by a compromised browser.

Keywords: *Browser Operating System Architecture, Java Virtual Machine, Operating System, Secure Platform.*

----------***----------

1. INTRODUCTION

The Web browser has become the dominant interface to a broad range of applications, including online banking, Web-based email, digital media delivery, gaming, and ecommerce services. Early Web browsers provided simple access to static hypertext documents. Modern browsers serve as de facto operating systems that must manage dynamic and potentially malicious applications. Unfortunately, browsers have not properly adapted to their new role. As a consequence, they fail to provide adequate isolation across applications, exposing both users and Web services to attack [2].

Many popular programs, such as Netscape, use un-trusted helper applications to process data from the network. Unfortunately, the unauthenticated network data they interpret could well have been created by an adversary, and the helper applications are usually too complex to be bug-free. This raises significant security concerns. Therefore, it is desirable to create a secure environment to contain untrusted helper applications [1]. This paper proposes to reduce the risk of a security breach by restricting the program's access to the operating system[5]. In particular, we intercept and monitor dangerous system calls via the Java process tracing facility. This enabled us to build a simple, clean, user-mode implementation of a secure environment for untrusted helper applications [1]. This implementation has negligible performance impact, and can protect pre-existing applications.

2. MOTIVATION

The Browser Operating System (BOS), a new trusted software layer on which Web browsers execute. The benefits of this architecture are threefold. First, the BOS runs the client-side component of each Web application (e.g., on-line banking, Web mail) in its own virtual machine. This provides strong isolation between Web services and the user's local resources. Second, this project lets Web publishers limit the scope of their Web applications by specifying which URLs and other resources their browsers are allowed to access. This limits the harm that can be caused by a compromised browser. Third, this project treats Web applications as first-class objects that users explicitly install and manage, giving them explicit knowledge about and control over downloaded content and code [2].

2.1 Browser instance

The browser instance associates with a single well circumscribed web application. Web services specify manifests.
Manifest contains:
– Digital signature authenticating web service.

– Browser policy: code to run in the browser instance.
– Network policy: Internet access policy to be enforced by reverse firewall.
User need to approve web application when web application is run for the first time.
Trusted computing base for Secure Browsing System Multiplexes the virtual screens of each browser instance into physical display.

2.2 Network policies

Enforce network policies for each instance. It store state for associated browser instances, bookmarks and manifests. Also, stores pre-forked browser instances that can be cloned easily when installing web application Problem Identification. Browsers run lot of active untrusted code [6]. Web applications interfere with other applications and with browser itself[1]. Exposes users and web services to a lot of risk
– Drive by download attacks
– Cross-site scripting attacks
– Content based attacks and Phishing attacks [4]

3. METHODOLOGY

This project implements a prototype of BOS. This project's security evaluation will show that this platform can prevent vulnerabilities that have been identified in the widely used web browser. In addition, project measurements of latency, throughput, and responsiveness demonstrate that users need not sacrifice performance for the benefits of stronger isolation and safety.

The BOS runs the client-side component of each Web application (e.g., on-line banking, Web mail) in its own virtual machine. The BOS lets Web publishers limit the scope of their Web applications by specifying which URLs and other resources their browsers are allowed to access.
The BOS treats Web applications as first class objects that users explicitly install and manage.

It defines a new trusted system layer, the browser operating system (BOS), on top of which browser implementations (such as Netscape or IE) can run.

It provides explicit support for Web applications. A Web application consists of a browser instance, which includes a client-side browser executing dynamic Web content and code, and a Web service, which is a collection of Web sites with which the browser instance is permitted to communicate [2].
It enforces isolation between Web applications, prohibiting one application from spying on or interfering with other applications or host resources. Each Web application has an associated browser instance that is sandboxed within a virtual machine.

It enforces policies defined by the Web service to control the execution of its browser instances, e.g., to restrict the set of Web sites with which a browser instance can interact. A Web service provides the BOS with a manifest – an object that defines its policies and other Web application characteristics.

It supports an enhanced window interface for browser instances. The BOS multiplexes windows from multiple instances onto the physical screen and authenticates the Web application for users.

It provides resource support to browser instances, including window management, network communication, bookmark management, and the execution of new Web applications [2].

4. THE SECURE WEB BROWSING SYSTEM (SWBS)

Secure Web Browsing System architecture has six key features:
1. It defines a new trusted system layer, the browser operating system (BOS), on top of which browser implementations (such as Netscape or IE) can run.
2. It provides explicit support for Web applications. A Web application consists of a browser instance, which includes a client-side browser executing dynamic Web content and code, and a Web service, which is a collection of Web sites with which the browser instance is permitted to communicate.
3. It enforces isolation between Web applications, prohibiting one application from spying on or interfering with other applications or host resources. Each Web application has an associated browser instance that is sandboxed within a virtual machine.
4. It enforces policies defined by the Web service to control the execution of its browser instances, e.g., to restrict the set of Web sites with which a browser instance can interact. A Web service provides the BOS with a manifest – an object that defines its policies and other Web application characteristics.
5. It supports an enhanced window interface for browser instances. The BOS multiplexes windows from multiple instances onto the physical screen and authenticates the Web application for users.
6. It provides resource support to browser instances, including window management, network communication, bookmark management, and the execution or "forking" of new Web applications.

4.1 Browser Operating System Architecture (BOS):

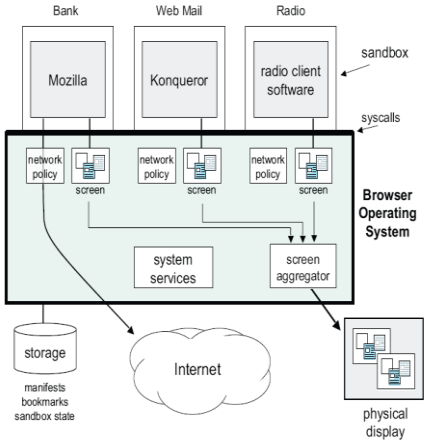

Fig 4.1: Browser Operating System Architecture (BOS)

Figure 4.1 shows a simplified, high-level view of the Secure Web Browsing System architecture. It identifies two Web applications, each consisting of a client-side browser instance and a remote Web service. Each browser instance comprises a browser and the Web documents that it fetches, executes, and displays. The browser operating system, shown below the browser instances, isolates Web applications from each other, preventing resource sharing or communication between browser instances. The BOS also manages communication between a browser instance and the Internet, permitting access to those sites (and only those sites) in the associated Web service.

4.2 Web Applications

The execution environment of a browser instance :

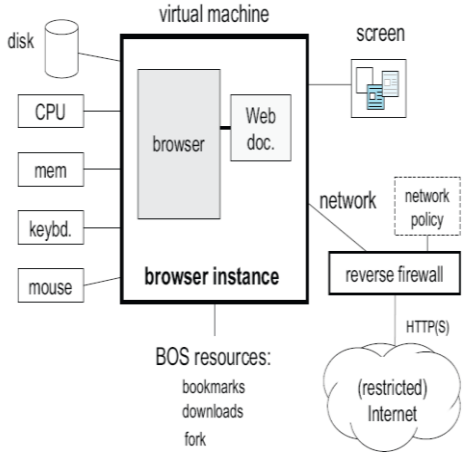

Fig 4.2: The execution environment of a browser instance

Figure 4.2 shows the execution environment as viewed by a client-side browser instance. Each browser instance executes in a virtual machine (VM) that has its own private virtual disk, CPU, memory, input devices, and screen. The VM also supports a virtual network, through which the browser instance interacts with remote Web sites. Unlike conventional browsers, which can browse and display multiple Web sites simultaneously, each browser instance is associated with a single, well-circumscribed Web application, e.g., an application that provides online access to the user's bank. Thus, system users have a unique browser instance associated with each running Web application.

The VM environment provided for the browser instance has several advantages. First, aWeb application is safe from interference by other applications, and it is free to modify any machine state, transient or persistent, without endangering other applications or the user's host OS. Second, the user can easily remove all local effects of a Web application by simply deleting its VM. Finally, the VM environment increases flexibility for the programming of Web applications. For example, a service could provide a customized browser rather than using a commodity browser.

Network policies protect the Web application from compromised browsers. Browsers are easily compromised by malicious plug-ins or through active Web content that exploits security holes in the browser or its extensions [10]. A compromised browser could capture confidential data flowing between the browser instance and its Web service and send that information to an untrusted Internet site, or it could use the browser instance as a base to attack other Internet hosts. The network policy and reverse firewall aim to prevent these attacks by restricting communication from the browser instance to legitimate sites within the Web service.

Users accessing a Web application for the first time must approve its installation. Only then will system create a new virtual machine, install within it the browser code and data, and execute the new browser instance. The BOS caches approvals, so the user need not re-approve a Web application on subsequent executions.

5. THE BROWSER OPERATING SYSTEM

The browser operating system is the trusted computing base for the secure browsing system. It instantiates and manages the collection of browser instances executing on the client. To do this, it must multiplex the virtual screens of each browser instance onto the client's physical display, enforce the network policies of each instance, and durably store state associated with browser instances, bookmarks, and manifests.

Figure 4.1 shows a detailed architectural view of the browser operating system. The figure contains three Web applications and their isolated browser instances. Two of the browser instances contain conventional browsers, while the third is executing a custom radio application instead of a conventional Web browser.

The BOS provides the highest level user interface, letting users manipulate the virtual screens of each browser instance. In addition, it wraps each virtual screen with a border that the browser instance cannot occlude. In the border, the BOS provides trusted information to the user, such as the name and credentials of the Web application with which the screen is associated. The BOS routes input events to the appropriate browser instance, similar to the way conventional window systems operate.

The BOS also provides users with control panels and bookmark management tools. These let the user install, execute, and uninstall Web applications, or create bookmarks that point to documents within a Web application. A bookmark has a familiar meaning in the context of a conventional Web browser. However, a Web application that provides its own custom browser instance may co-opt bookmarks for its own purposes. For example, a streaming radio service could use bookmarks to implement radio channels.

The BOS mediates all network interactions between a browser instance and remote Web sites. To access the Web, a browser instance invokes a BOS system call that fetches Web documents over HTTP. The BOS will service the connection only if the document falls within the network policy specified in the instance's manifest. If not, the BOS refuses the request. If the document is allowed by the manifest of a different Web application, the BOS gives the user the option of loading it into that Web application's browser instance.

Web applications have durable state that the BOS must manage. Sandboxes provide private virtual disks to browser instances, and the BOS maintains the state of these disks between invocations of the Web application. It also stores a set of "stock" browser instances (e.g., Mozilla) that can be cloned when installing a Web application. Finally, the BOS stores manifests and bookmarks associated with Web applications. It treats all long-term storage as a soft-state cache.

Accordingly, durable state can be evicted, but at the cost of having to re-download manifests or re-install browser instances when the user next accesses a Web application.

The proposed architecture is driven by the principles: distrust of Web browsers and applications, and the empowerment of users. The resulting architecture isolates Web applications, protecting other applications and client resources from malicious downloaded code [9]. In addition, it permits Web services to build safer, more powerful Web applications. Overall, our goal is to accept the enhanced role of modern browsers in managing the client-side components of complex, non-trusted, distributed applications.

6. SAFETY AND EFFECTIVENESS

A critical measure of this architecture's value is whether it successfully prevents or contains threats that occur in practice. Isolation should provide significant safety benefits.

As an example, security vulnerabilities can arise due to Secure Web Browsing System (SWBS) dependence on external systems, such as DNS. Attackers that subvert DNS can subvert system's network filtering policies by changing legitimate bindings to point to IP addresses outside of the intended domain. Secure Web Browsing System cannot defend itself from these attacks [7]. Another example is a malicious browser instance, which could use a sharing interface provided by system to attack another browser instance or Web application. While system greatly reduces the number of sharing channels, these channels still exist. Consider a browser that contains a buffer-overflow vulnerability in its URL parsing code. A malicious browser instance could use the fork browser-call to pass an attack string to a second browser instance, potentially subverting it. Any channel that permits sharing between mutually distrusting Web applications is susceptible to attack.

6.1 Sandbox weakness:

Browsers use language and runtime mechanisms to sandbox scripts, applets, and other active Web content, but these language- and type-specific sandboxes are often flawed. In contrast, the secure web browsing system uses virtual machines as a language independent sandbox for the entire browser instance.

6.2 Vulnerable sharing interface:

Browsers contain many programmatic interfaces (e.g., DOM access) and user interfaces (e.g., file upload dialog boxes) for sharing data across security domains. These interfaces can often be subverted. In secure web browsing system, we limit sharing across Web applications to a small set of browser-calls and holding bin manipulation interfaces.

6.3 Improper labeling:

Browsers assign Web objects to security domains using a complicated set of heuristics. Incorrectly labeling an object as belonging to a domain can enable attacks such as drive-by downloads. In secure web browsing system, Web services explicitly declare the scope of their Web application through manifests.

6.4 Interface spoofing:

Browsers are susceptible to spoofing attacks, in which a malicious site attempts to occlude or replicate browser UI elements or the "look and feel" of victim sites. In secure web browsing system, the system window manager decorates browser instances with labeled borders that cannot be accessed or occluded.

Other: Some vulnerability could not easily be classified; this category is a "catch-all" for these.

7. PERFORMANCE

Our analysis of Mozilla vulnerabilities demonstrates that secure web browsing system can increase safety and security for Web browsing. However, there is typically a tradeoff between safety and performance. Given secure web browsing system's use of VMs for isolation, what is the cost of virtualization to the user and to the Web application?

To answer this question, we ran several benchmarks to quantify the performance of common Web-browsing operations and the overhead of secure web browsing system's browser virtualization.

8. OUTCOMES

Over the last decade, the Web has evolved from a repository of interconnected, static content to a delivery system for complex, distributed applications and active content. As a result, modern browsers now serve as de facto operating systems that must manage dynamic and potentially malicious applications.

Output Window – I

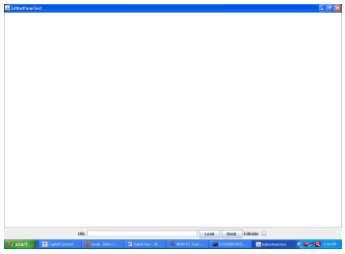

Fig. 8.1 Secure Web Browser Interface-I

Output Window – II

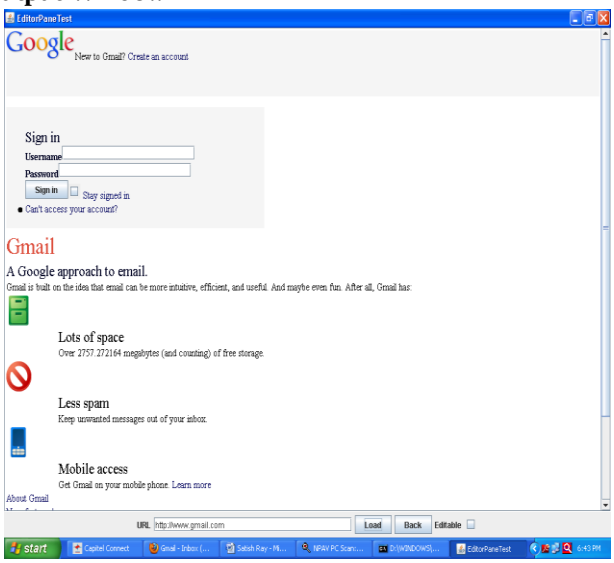

Fig. 8.2 Secure Web Browser Interface-II

CONCLUSION

The Secure Web Browsing System can prevent vulnerabilities that have been identified in the widely used web browser. In addition, its performance evaluation will demonstrate that users need not sacrifice performance for the benefits of stronger isolation and safety. To limit damage from hijacked browsers, the Browser Operating System restricts the set of sites with which each Web application can communicate. The Browser Operating System gives users increased visibility and control over downloaded Web applications.

The resulting architecture will isolate Web applications, protecting other applications and client resources from malicious downloaded code. In addition, it permits Web services to build safer, more powerful Web applications. Overall, goal is to accept the enhanced role of modern browsers in managing the client-side components of complex, non-trusted, distributed applications.

REFERENCES

[1] Ian Goldberg, David Wagner, Randi Thomas, and Eric Brewer, "A Secure Environment for Untrusted Helper Applications (Confining the Wily Hacker)", Proceedings of the Sixth USENIX UNIX Security Symposium, July 1996.

[2] Cox, R.S., Hansen, J.G. Gribble, S.D., and Levy, H.M. "A safety-oriented platform for Web applications", IEEE Symposium on Security and Privacy, Print ISBN: 0-7695-2574-1, 15 pp. – 364, May 2006.

[3] Grier, C., Shuo Tang, and King, S.T. "Secure Web Browsing with the OP Web Browser", IEEE Symposium on

Security and Privacy, Print ISBN: 978-0-7695-3168-7, page(s): 402 – 416, May 2008.
[4] Vinod Anupam and Alain Mayer, "Security of Web Browser Scripting Languages: Vulnerabilities, Attacks, and Remedies", Proceedings of the 7th USENIX Security Symposium, January 1998.
[5] D. Dean, E. W. Felten, D. S. Wallach, and D. Balfanz, Java security: Web browsers and beyond. Chapter 7 of "Internet besieged: Countering cyberspace scofflaws", ACM Press, 1997.
[6] A. Acharya and M. Raje. MAPbox, "Using parameterized behavior classes to confine untrusted applications", Proceedings of the Ninth USENIX Security Symposium, August 2000.
[7] D. Balfanz and D. R. Simon. Windowbox, "A simple security model for the connected desktop" Proceedings of the Fourth USENIX Windows Systems Symposium, August 2000.
[8] S. Ioannidis and S. M. Bellovin, "Building a secure Web browser", Proceedings of the FREENIX track of the 2001 USENIX Annual Technical Conference, June 2001.
[9] T. Jaeger, A. D. Rubin, and A. Prakash, "Building systems that flexibly control downloaded executable content", Proceedings of the Sixth USENIX Security Symposium, July 1996.
[10] D. S. Wallach, D. Balfanz, D. Dean, and E. W. Felten. "Extensible security architectures for Java", Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles (SOSP '97), October 1997.

ACKNOWLEDGEMENT

I would like to thank my guide **Mr.Toran Verma** for his immense support and enlightened guidance for my paper which I had prepared as a M.Tech. student. I am very grateful for the inspiring discussions with all my faculties and department heads that led to this paper, whose valuable help and path-guiding supervision helped me to prepare this paper. Also I am thankful to Dr. (Mrs.) Sipi Dubey for her kind cooperation and suggestions.
I owe the greatest debt and special respectful thanks to Mr. Sourabh Rungta, Director (Tech.), Mr. Sonal Rungta, Director (F&A.), and Dr. S. M. Prasanna Kumar, Principal, Rungta College of Engg. & Tech., Bhilai for their inspiration, and constant encouragement that enabled me to present my work in this form.

BIOGRAPHIES

Satish Kumar Ray, has completed Bachelor of Engineering from Bhilai Institute of Technology Durg. Currently pursuing M.Tech.(Computer Technology) from RCET Bhilai/ CSVTU Bhilai C.G. India.

Mr. Toran Verma, Reader, Department of Computer Science and Engineering, RCET Bhilai-490024, C.G. India.

Dr. (Mrs.) Sipi Dubey, Dean(R&D), RCET Bhilai-490024, C.G. India.