A Safety-Oriented Platform for Web Applications

IEEE Symposium on Security and Privacy, 2006

Richard S. Cox, Jacob Gorm Hansen,
Steve D. Gribble, Henry M. Levy
University of Washington

By Kiran Kumar Gollu
Tuesday, October 17, 2006
ECE 1776

Agenda

- Motivation
- Design Goals
- Architecture
- Results/Evaluation
- Related Work
- Conclusions
- Pros and Cons

Motivation

- Browsers run lot of active untrusted code
- Web applications interfere with other applications and with browser itself
- Exposes users and web services to a lot of risk
  - Drive by download attacks
  - Cross-site scripting attacks
  - Content based attacks and Phishing attacks

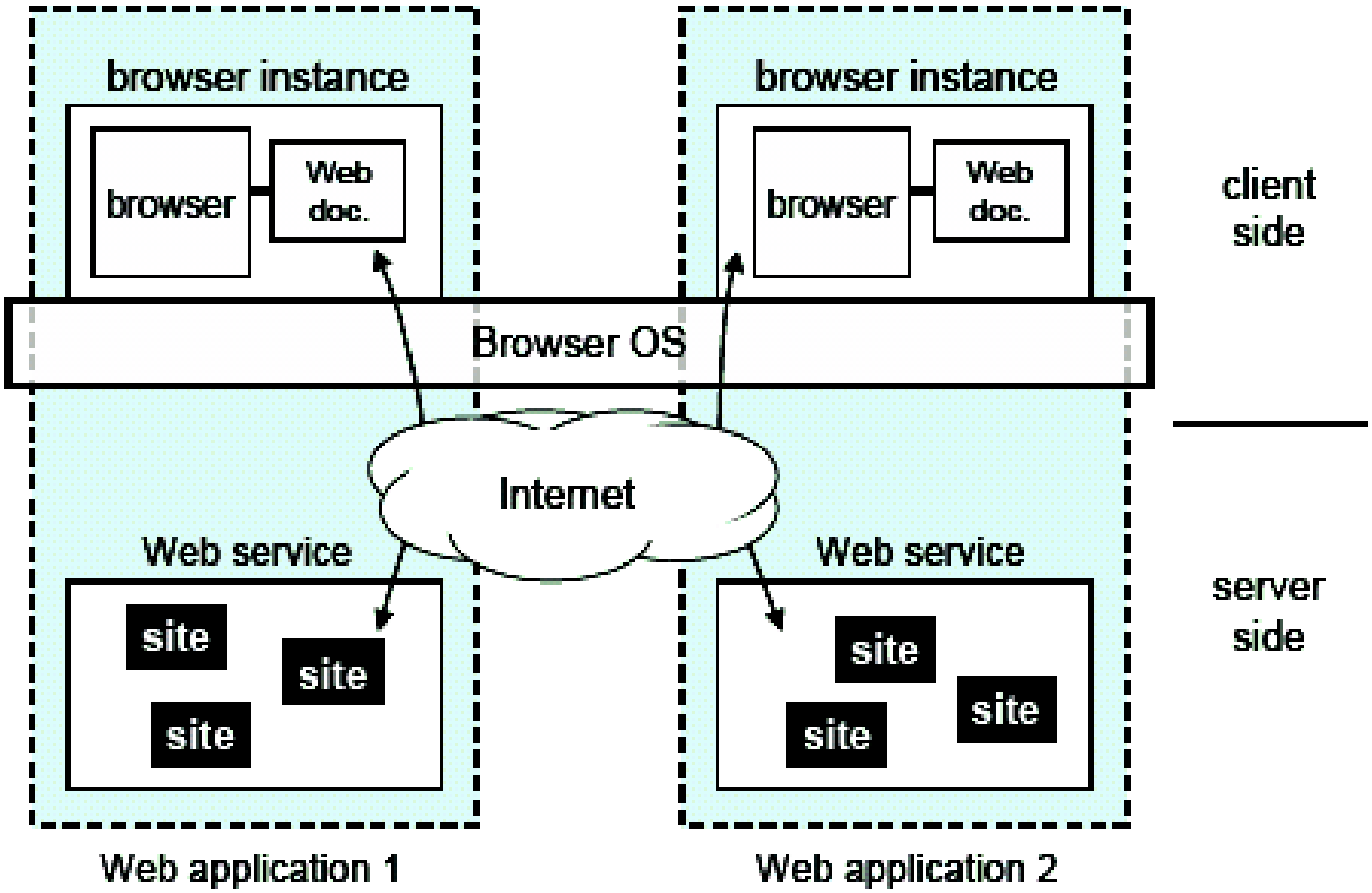
Design Goal

- Develop a new browser platform to improve safety and security for users and web services
  - New trusted layer on which web browsers execute

Tahoma Architecture

- Web applications should not be trusted
  - Web Application = Browser Instance + Web Service
  - Contain each browser instance in VM sandbox
- Web browsers should not be trusted
  - Isolate browsers from rest of the system
- Increase visibility and control over downloaded web applications
  - Web applications should be visible to users like desktop applications

Tahoma Architecture



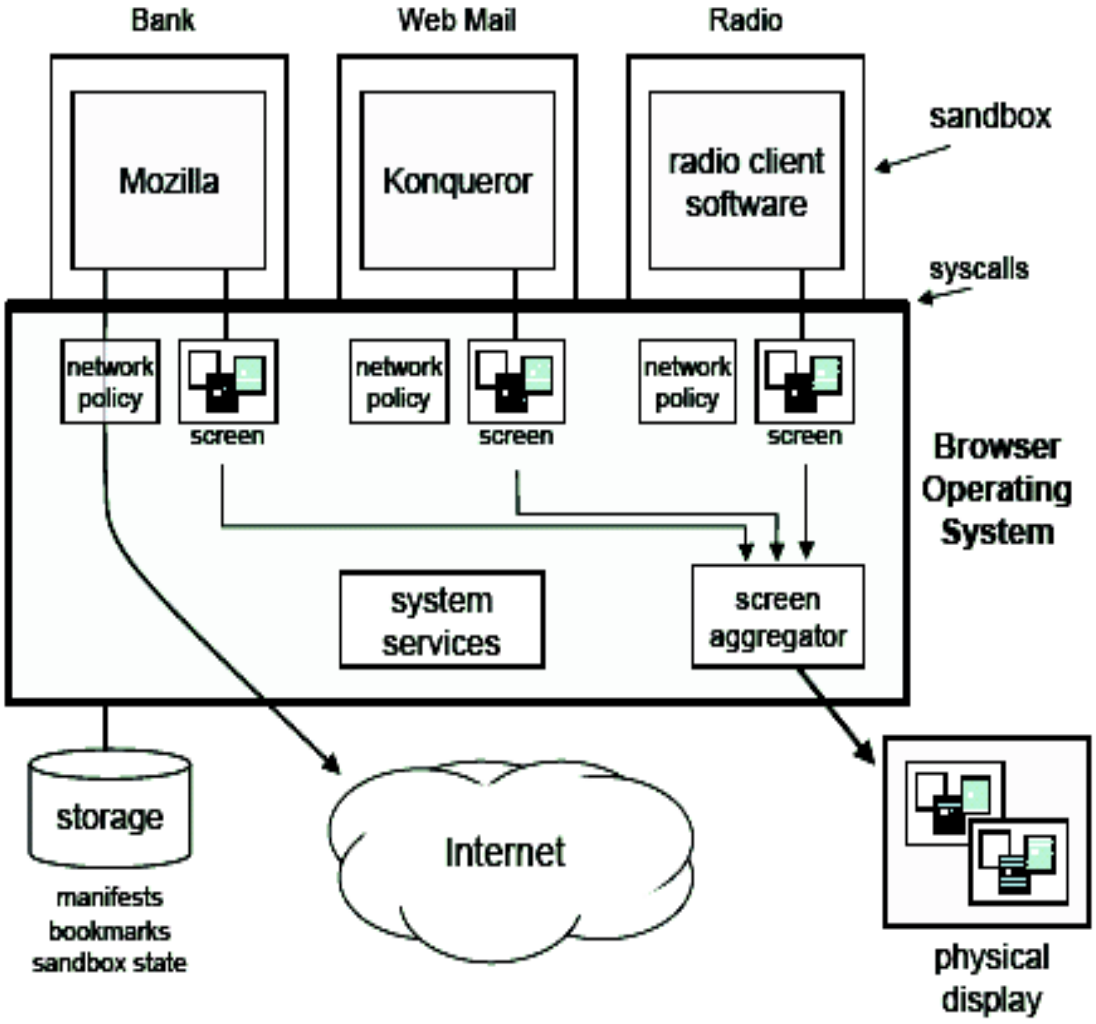
Web Application

- Tahoma 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

Browser Operating System(BOS)

- Trusted computing base for Tahoma browsing system
- Multiplexes the virtual screens of each browser instance into physical display
- Enforce network policies for each instance
- Store state for associated browser instances, bookmarks and manifests
- Also, stores pre-forked browser instances that can be cloned easily when installing web application

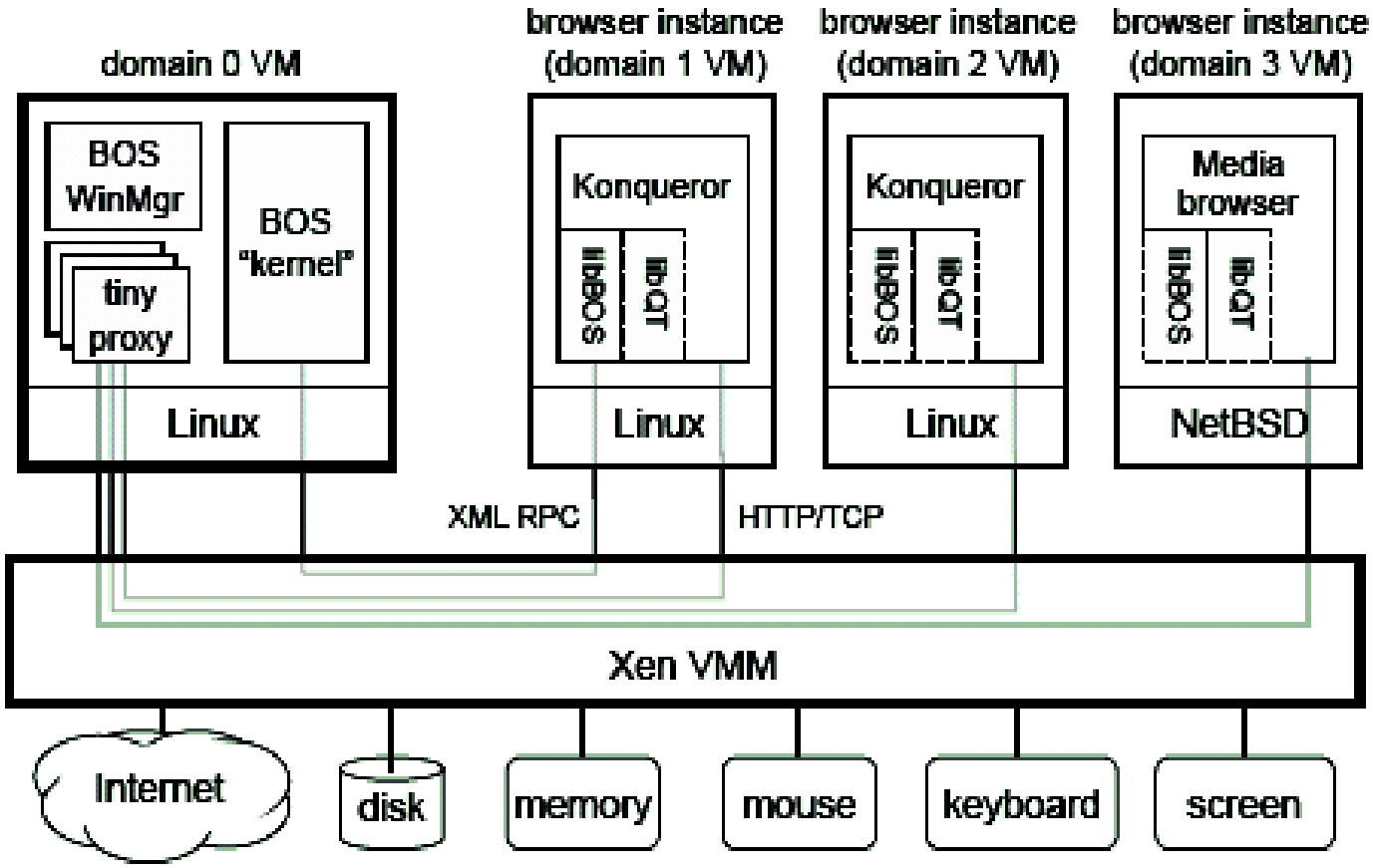
Tahoma Implementation



Implementation

- Tahoma prototype implemented on Xen virtual machine monitor in Linux
- BOS, BOS Kernel and tiny proxy implemented as domain0 VM
- User mode applications run on guest OS at lowest privilege level (domain-1 through domain-N)
- Manifests
- Window Manager aggregates virtual screens on the physical screen
- Browser instance is run on Xen Virtual MAchine

Tahoma Implementation on Xen



Tahoma Evaluation

- Safety and Effectiveness
  - Tahoma contained 95 out of 109 vulnerabilities in Mozilla (87% of the attacks)
- Performance overhead
  - Works well if pre-forked browser instance already exists
  - Cost of launching browser is high if no pre-forked instance

operation		average latency
Tahoma fork()	specialize a pre-forked browser instance	1.06 seconds
	clone a new VM, boot guest OS, launch browser program	9.26 seconds
native Konqueror open URL	load URL in running Konqueror	0.84 seconds
	warm-start Konqueror	1.32 seconds
	cold-start Konqueror	5.74 seconds

Related Work

- GreenBorder:
  - provides sandbox environment for IE and outlook
  - Virtualizes access to windows resources and redirect modifications to virtualized copies.
- Collective Project
  - Collections of applications within VMWare virtual machines
  - Ships above compute appliances over the network

Pros

- Novel approach – isolating web applications not only with user OS but also with other web applications
  - Better control for web services in defining policies through manifests
  - Prevents cross-site scripting attacks
- VM contain security vulnerabilities to one single browser instance
  - Even if the browser is compromised, it limits damage to one browser instance
  - E.g. Attack on SSL certificate management scheme

Pros

- Network policies protect web applications from compromised browsers
  - Protection against security holes in the browser
  - Prevents web application from sending information to a untrusted site (drive by download attacks – spyware infections)
- Secure sharing interface between web applications
  - Limit browser calls to Fork, BinStore, BinFetch
- Provides language independent safe execution environment for browser instances(VM)

Pros

- BOS kernel stores a set of VM checkpoints of freshly created stock browsers
  - Reduces the overhead in creating new browser instance
- Prevents phishing attacks to some extent using labelled borders

Cons of Implementation

- Increases complexity of web applications – More maintenance and required
- Manifest Issues:
  - Every web application will need a manifest created for them. Who will maintain them?
  - No mention of how manifests will be adopted
  - If manifest is incorrectly specified, web application will not work
  - Updating of manifest file may be problematic – Not dynamic

Cons of Implementation (contd)

- Does not prevent attacks from hi-jacked web applications:
  - Permissible browser unavailable, BOS relies on web service to supply URL of VM image
  - This VM image downloaded & executed- *This may be a hi-jacked phishing web application!*
- BinStore & BinFetch browser calls may be susceptible to format string vulnerabilities & buffer overflow attacks

Cons of Paper Content and Format

- Limited mention of the “trusted Tahoma tool” for transferring objects between the holding bin and the host OS
- Contradiction: “Most Apps will run on Tahoma with little or no modification. However, three kinds of modifications may be necessary …”

- Increases complexity of web applications – More maintenance and required

Cons of Paper Content and Format

Manifest Issues:

- Every web application will need a manifest created for them. Who will maintain them?
- No mention of how manifests will be adopted
- If manifest is incorrectly specified, web application will not work
- Updating of manifest file may be problematic – Not dynamic

Q & A