Prevention of Information Leakages in a Web Browser by Monitoring System Calls

Harshad Wadkar
Computer Engineering Dept.
Defence Institute of Advanced Technology
Pune, India
wadkarharshads@rediffmail.com

Dr. Arun Mishra
Computer Engineering Dept.
Defence Institute of Advanced Technology
Pune, India
arunmishra@diat.ac.in

Dr. Arati Dixit
Computer Engineering Dept.
PVPIT
Pune, India
adixit98@gmail.com

Abstract— **The web browser has become one of most accessed process/applications in recent years. The latest website security statistics report about 30% of vulnerability attacks happen due to the information leakage by browser application and its use by hackers to exploit privacy of an individual. This leaked information is one of the main sources for hackers to attack individual's PC or to make the PC a part of botnet. A software controller is proposed to track system calls invoked by the browser process. The designed prototype deals with the systems calls which perform operations related to read, write, access personal and/or system information. The objective of the controller is to confine the leakage of information by a browser process.**

Keywords— browser security; confinement; information leakage.

I. Introduction

With the ease of access and good speed, popularity of the internet is growing. Online banking, gaming, digital media deliveries have become increasingly popular over the last few years. The web browser provides the interface to access these services. Modern web browsers have not been designed and developed to guarantee personal web privacy. There is a plethora of tools available online that can tell what kind of information may be leaked while surfing on net. The information leakage occurs even during trivial activities like search using search-engines, or participating on social networking sites.

Information Leakage is largely a catch-all term used to describe vulnerability where a website reveals sensitive data like:

1. Technical details of the web application/environment,
2. Browser specific Information
3. User specific data

This sensitive data may seem to be completely irrelevant for a typical visitor but it may be used by an attacker to exploit the users, system, and the respective hosting network. User's email addresses and passwords can be used in masquerading. The hacker or salesperson can use the user's email address and telephone number to send spam mails containing advertises and probably adwares. By reading credit card numbers, pin numbers and bank account numbers, the hacker can fake financial transactions. These discussed attacks are the clear and direct attacks caused by information leakage. But there are also indirect ways of information leakage such as read transaction history, read user's system information to identify user's internet usage pattern, the weaknesses in the user's system using which user's system can be compromised and can also be used to attack other systems.

The private information can be leaked using seemingly harmless information such as *interests* on social networking sites and augmenting it with semantic knowledge [16]. The four classes of activities defined harmful for both daily life and online privacy of individuals are [18]:

1. Information Collection which includes all activities related to surveillance
2. Information Processing which involves the way information is stored, aggregated, linked and used
3. Information Dissemination which involves the way information is distributed
4. Invasion with intrusions into people's private affairs.

According to Mayer [10], the identity information can signal not just *"what you're doing right now, but ... what you've done in the past and what Web browsing activity you may have in the future"*.

Fig. 1 shows the results cited from Website Security Statistic Report, May 2013 [8]. As per the report, the information leakage vulnerability is the most vulnerable class of attacks with attack percentage ranges from 21% to 35%.

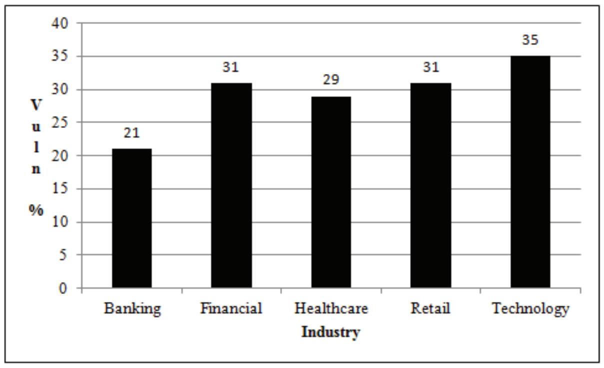

Fig. 1. Industry wise information leakage vulnerability % [8]

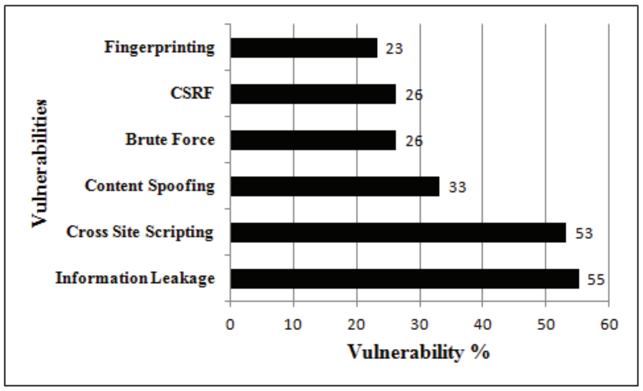

Fig. 2. Top vulnerability classes [8]

The other major vulnerability classes are cross-site scripting, content spoofing, Cross site request forgery (CSRF), Fingerprinting etc. Fig. 2 shows the vulnerability classes along with their percentage chances of appearing atleast one serious vulnerability in a website.

To collect any personal / system information, any application is required to make use of underlying system calls. Therefore it is of great significance to monitor and control system call execution to thwart information leakages which is a prime objective of this work.

The paper is organized as follows. The section 2 highlights the related work. The section 3 discusses different ways that help in information leakages and the leakages observed using some available online tools. The system components and functioning of the proposed system are described in section 4. We conclude in section 5 along with listing future work.

II. RELATED WORK

The idea of developing a controlled environment in which a possibly untrustworthy program can be run safely was proposed by Butler Lampson in his research paper "A Note on the confinement problem" [4]. The paper [26] presented four aspects of privacy leak. Leak content is the data that leaked. Leak source refers the storage form of privacy data. System call sequence and final destination constitute the leak procedure. The leak destination reveals the location where the leaked data is passed.

To prevent information leakage on the internet *Personal Information Transfer Control Model* has been proposed. The model inspects every network packet transferred from user's PC to a server to check whether the packet contains any user's personal information and based upon predefined user control policy a decision is made to forward or discard [12]. A technique [17] has been proposed to quantify information leak capacity in network traffic. The measurement algorithm is focused on analyzing leaks in HTTP traffic.

Smetters and Stewart defined a term "mismatch problem" where the user sends sensitive data to an unintended user. They prototyped a model which combines protected link or secure bookmark that authenticated their targets. The model also helps the user to bookmark legitimate sites by performing automatic bookmark integrity verification and whitelisting of allowed bookmarks [13]. A website can learn what urls user has visited, this information can be used to guess surfing behavior of the user. In 2010, software patches have been developed to limit the ability of Web pages to style pages based on whether links are visited. This still enables to differentiate visited from unvisited links [15]. Many research articles have been published which discuss the JavaScript's ability to inspect how a webpage is rendered and leaks information of user's browsing history, list of visited urls [14]. Some browsers interpret JavaScript code, which allow the JavaScript program to monitor visited documents, bookmarks, form data, critical information stored in browser cookies.

The architecture of secure browser based on process specific protection mechanism has been proposed to handle attacks by incoming malicious objects [5]. Krishnamurthy et al. [27] explored the information leakages by first-party websites to third-party websites with or without using third-party cookies. Without using cookies or any other standard tracking methods, browser provides some system configurations like version, operating system, IP address etc. This information is sufficient to track user or his/her behavior pattern [19]. A user mode implementation [3] is demonstrated to intercept and filter dangerous system calls. This Solaris process trace based method reduces the risk of a security breach by confining the program's access to the operating system. A software controller [2] has been proposed to monitor all programs that execute within the browser for possible information leakage attempts through IPC. Tahoma, the Browser Operating System (BOS) is a trusted software layer on which web browsers execute [1]. This layer separates web services and user's local resources. The techniques such as runtime reconfiguration and feature modeling have been explored to introduce noise into the actual browser fingerprint, this limits capacity of distinctive identification of user or his/her web surfing activities [25].

The process to make the browser secure from local malware threats protects data from keyloggers, screen scraping and cache raiders. Encrypting and deleting data written from the browser to the local cache, preventing the cut, copy, paste, print and screen capture features and delivering this secure web browser protection as part of the application closes many of the current security gaps in meeting Payment Card Industry Data Security Standard (PCI DSS) requirements [9].

The extensive comparison of privacy protection tools like NoTrace, AdBlock Plus etc. is discussed in [6]. The paper also discusses categorization of information leakage into Low, Medium and High based upon the degree of sensitivity and identifiability.

III. BROWSER INFORMATION LEAKAGE ANALYSIS

A "vulnerability" is a specific problem in the code having a security impact while an "attack vector" is a way of triggering/reaching the vulnerability. [28]

A. *Browser information leakage analysis vector (BILAV)*

The analysis of literature survey helped in preparing the information leakage vector. The BILAV is listed in Table 1

Table 1 : BILAV

1.	The hacker may steal personal information of a registered user of a website by attacking the server on which the website is hosted.
2.	Using phishing technique, the hacker can send a url of faked website which looks legitimate to human eyes and unknowingly the user inputs his credentials or personal information.
3.	The attacker may steal user's personal information by using some spyware or capturing keystrokes using some keylogger.
4.	Performing packet sniffing of network packets transferred between the user's PC and application server.
5.	Reading logs or information kept by the application server on user's PC such as cookies, visited urls.
6.	Reading system specific information that can be leaked by the browser running on user's PC which can identify user and / or his/her PC uniquely on the internet.
7.	User's information can also be leaked by reading clipboard contents, accessing system registry and installed plug-ins.
8.	Web browsers interpreting JavaScript and VBScript code snippets.
9.	Browser fingerprinting to find version, system configuration information etc.

B. Experimetal analysis

Using tools / functionalities available at different websites, we tried to find what information is dripped and what kind of information can be deduced from the leaked data.

The test machine used for experimentation run on Ubuntu 12.04 LTS with Mozilla 20.0 Firefox browser.

In this experimentation, we have accessed the sites – browerleaks.com, browserspy.dk, using the browser running on the test machine. We identified that the browser leaks the information about browser and system configuration.

1) Browser Information leaks

The Table 2 & 3 show some information leaked by the browser running on test machine while accessing tools on different websites

a) http://www.browserleaks.com/

The features accessed are IP address, Do Not Track.

Table 2 : Extraction of information leakage ingredients (browserleaks.com)

HTTP Request Headers	
Accept	text/html, application/xhtml+xml, application/xml;q=0.9,*/*;q=0.8
Connection	keep-alive
Referer	http://www.browserleaks.com/
User-Agent	Mozilla/5.0 (X11; Ubuntu; Linux x86_64;rv:20.0) Gecko/20100101 Firefox/20.0
Navigator Object	
Build ID	20130329030832
OS CPU	Linux x86_64
Do Not Track	Unspecified
Gelocation API	[object GeoGeolocation]
Cookies Enabled	True
HTTP Header method to detect a DO Not Track preference	
DNT	× Disabled
JavaScript methods to detect a Do Not Track preference	
navigator.doNotTrack	× Disabled
navigator.msDoNotTrack	× Disabled

From the above output, the information that can be realized [20, 21, 22] is :

CPU	:	Intel 64 bit processor
OS	:	Linux, Ubuntu distribution
Browser	:	Mozilla with version 5, Gecko engine with version 20.0, Firefox version 20
Do Not Track (DNT)	:	DNT is disabled means visited websites can learn about user's browsing behavior

b) Using Tools on http://browserspy.dk/ [24]

Table 3 : Extraction of information leakage ingredients (browserspy.dk)

Using proxy	:	No
Max connections per hostname	:	3
Max connections	:	3
Cookies enabled?	:	Yes
JavaScript Cookies supported?	:	Yes
Server Cookies supported?	:	Yes
HTTP Only Cookies supported?	:	Yes
Can JavaScript read HTTP Only Cookies?	:	No
Personal Security Manager version	:	2.4

From the above test result, the information that can be realized [24] is :

- Connection information tells how many maximum connections browser opens for a single hostname.

- Whether browser running on user's machine supports cookies? Whether the cookies are enabled?
- Personal Security Manager (PSM) provides cryptographic operations for setting up an SSL connection, object signing and signature verification, certificate management etc.

2) Browser uniqueness identification

The experiment carried to find uniqueness of the browser running on test machine. The testing is done using a functionality provided by https://panopticlick.eff.org/ [23].

The result is as follows : *Your browser fingerprint appears to be unique among the 3,530,701 tested so far. Currently, we estimate that your browser has a fingerprint that conveys at least 21.75 bits of identifying information.*

In our opinion, by embedding web browser security testing tools to his website, the attacker can have access to user's personal identity data which is leaked by the user's browser.

IV. PROPOSED SYSTEM

The proposed software controller provides an intermediate layer between browser and operating system. The controller inspects system calls invoked during the execution of browser process. The system call is the fundamental interface between an application and the Linux kernel. System calls are generally not invoked directly, but rather via wrapper functions in glibc. The code belonging to the system call with number __NR_xxx defined in /usr/include/asm/unistd.h can be found in the kernel source in the routine sys_xxx() [11]. The present prototype monitors 72 system calls as listed in Table 4.

Table 4 : System calls used for normal web browsing [7]

Type	System calls	Number
Reading/Writing	writev, read, recv, write, send, sendto, recvfrom, recvmsg	08
I/O Wait	poll, select	02
Network query/control	connect, getpeername, bind, socket, getsockname, getsockopt	06
File query/control	close, stat64, open, getdents64, fstat64, fcntl64, ioctl, _llseek, unlink, lseek, access, rename, dup2, getdents, lstat64	15
Memory management	mmap2, munmap, brk, mprotect, mlock	05
Shared memory/IPC	futex, shmctl, pipe, set_robust_list, semop, semget, semctl, shmat, shmdt, shmget	10
Signal management	rt_sigreturn, sigreturn, rt_sigaction, rt_sigprocmask	04
Process/thread management	clone, waitpid, execve, getpid, getppid, getpgrp, getrlimit, getuid32, getgid32, geteuid32, getegid32, getresuid32, getresgid32, set_thread_area, set_tid_address, sched_getaffinity	16
Time	gettimeofday, time, clock_gettime	03
Miscellaneous	shutdown, restart_syscall, uname	03

The controller contains one daemon process and one or more child processes. The task of daemon process is to detect browser process instantiation and generation of child process. Each child process monitors only one browser process assigned to it by the daemon process. Fig. 3 shows the how the processes and kernel interact with each other to carry out the task of restricted information passing to the browser process.

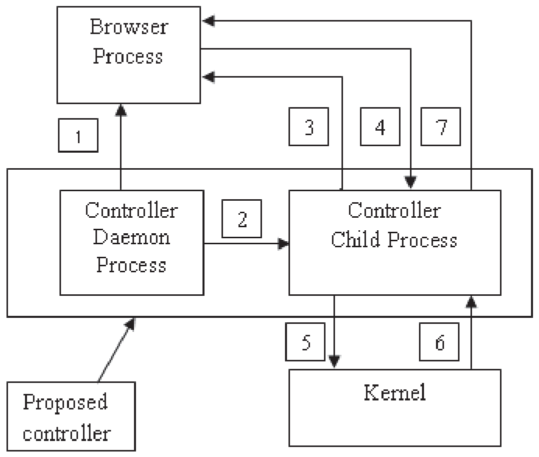

Fig. 3: System components and their interactions

The interaction among the system components as shown in Fig. 3 with labels 1,2, … 7. is as follows

1. Controller daemon process detects browser process.
2. Daemon process creates child process using fork system call.
3. Controller child process (CCP) starts monitoring the browser process.
4. System call made by browser process detected by CCP.
5. CCP passes the system call to kernel.
6. The Kernel executes the system call and returns the result to CCP.
7. The CCP checks whether the result contains personal and / or system information. The information is passed to the browser process if it is not violating security policy.

The child process monitors the system calls invoked by the browser process and makes decision about execution of system call and returning results to the browser process. The flowchart depicting working of controller child process is represented in Fig. 4.

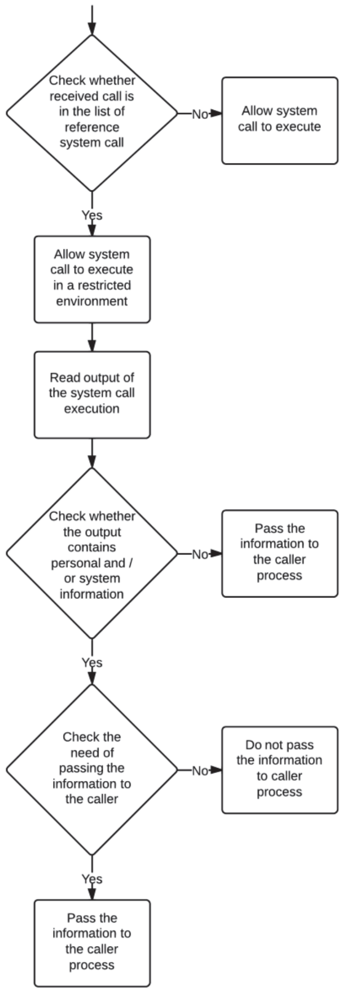
Check whether received call is in the list of reference system call
No
Allow system call to execute
Yes
Allow system call to execute in a restricted environment
Read output of the system call execution
Check whether the output contains personal and / or system information
No
Pass the information to the caller process
Yes
Check the need of passing the information to the caller
No
Do not pass the information to caller process
Yes
Pass the information to the caller process

Fig. 4 Working of controller child process

V. CONCLUSION & FUTURE WORK

By examining user's system, environment or personal data which is leaked by application (browser), an attacker can plan targeted attacks. These attacks give rise to serious problems such as privacy violation, masquerading and financial fraud.

The proposed software controller examines system call invoked by browser process to prevent information leakage done by a browser running on client machine.

The present design handles 72 system calls. As per the list of linux-headers-3.2.0-49 system calls there are 1079 system calls. It is a challenge to analyze the number of system calls out of 1079 which are invoked by web browser. The present controller design is fairly scalable to accommodate more number of system calls.

A portion of BILAV is implemented in the proposed controller which protects user from variety of information leakage avenues. The 100% implementation for BILAV will produce a secured and robust web browsing experience.

ACKNOWLEDGMENT

We would like to thank Deepti Vidyarthi, Anita Thite and the anonymous reviewers for their helpful comments and advice.

REFERENCES

[1] Richard Cox, Jacob Hansen, Steven Gribble, Henry Levy, “A Safety-Oriented platform for Web Applications,” Proceedings of the 2006 IEEE Symposium on Security and Privacy, pp 350 – 364.

[2] Nemisha Shrama, Swati Kare, Sanket Chichni, Vidhi Naredi, Jyoti Nandimath, Arun Mishra, Arati Dixit, “Monitoring Information Leakage in a Web Browser,” Security in Computing and Communications, Communications in Computer and Information Science, Springer Berlin Heidelberg, Volume 377, 2013, pp 322-329.

[3] Ian Goldberg, David Wagner, Randi Thomas, Eric Brewer, “A Secure Environment for Untrusted Helper Applications,” Proceedings of the 6th conference on USENIX Security Symposium, July 1996.

[4] Butler Lampson, “A Note on the Confinement Problem”, Communications of the ACM, Volume 16 Issue 10, Oct. 1973, pp 613-615.

[5] Sotiris Ioannidis, Steven Bellovin, “Building a Secure Web Browser,” Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference, Pages 127-134.

[6] Delfina Malandrino, Andrea Petta, Vittorio Scarano, Luigi Serra, Raffaele Spinelli, Balachander Krishnamurthy, “Privacy Awareness about Information Leakage: Who knows what about me?,” 2013 Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, - November 4, 2013, [Online]. Available: http://www.di.unisa.it/~delmal/papers/UNISA-ISIS-082913TR.pdf. [Accessed: Oct. 29, 2013]

[7] Joel Galenson, Christopher Grant, Jones Jason Lo, “Toward a Browser OS Designing a next-gen mobile OS with web technologies,” [Online]. Available: http://www.cs.berkeley.edu/~bodik/research/cs262-abstraction-tax.pdf . [Accessed: Oct. 29, 2013]

[8] Website Security Statistic Report, May 2013, [Online]. Available: https://www.whitehatsec.com/assets/WPstatsReport_052013.pdf. [Accessed: Oct. 29, 2013]

[9] Web Browsers: Your Weak Link in Achieving PCI Compliance, [Online]. Available: http://www.quarri.com/files/Quarri_PCI_Brief.pdf. [Accessed: Oct. 29, 2013]

[10] Sid Kirchheimer, “Many Websites Leak Personal Data,” AARP Bulletin, October 13, 2011, [Online]. Available: http://www.aarp.org/technology/privacy-security/info-10-2011/many-websites-leak-personal-data.html. [Accessed: Oct. 29, 2013]

[11] Linux System calls, [Online]. Available: http://manpages.ubuntu.com/manpages/hardy/man2/syscalls.2.html. [Accessed: Oct. 29, 2013]

[12] Daeseon Choi, Seunghun Jin, Hyunsoo Yoon, “A Personal Information Leakage Prevention Method on the Internet,” IEEE Tenth International Symposium on Consumer Electronics, 2006, pp-1-5.

[13] D.K. Smetters and Paul Stewart, “Breaking out of the Browser to Defend Against Phishing Attacks,” Fifth Conference on Email and Anti-Spam (CEAS 2008); 2008 August 21-22; Mountain View, CA

[14] Zachary Weinberg, Eric Y. Chen, Pavithra Ramesh Jayaraman, Collin Jackson, "I Still Know What You Visited Last Summer - Leaking browsing history via user interaction and side channel attacks," 2011 IEEE Symposium on Security and Privacy, pp 147-161.

[15] L. David Baron, "Preventing attacks on a user's history through CSS : visited selectors," 2010. [Online]. Available: http://dbaron.org/mozilla/visited-privacy [Accessed: Oct. 29, 2013]

[16] Abdelberi Chaabane, Gergely Acs, Mohamed Ali Kaafar, "You Are What You Like! Information Leakage Through Users' Interests," In Proceedings of the 19th Annual Network & Distributed System Security Symposium (February 2012). [Online]. Available: ww.crysys.hu/~acs/publications/ChaabaneAK11ndss.pdf .]Accessed :Oct .29 ,2013[

[17] Kevin Borders, Atul Prakash, "Quantifying Information Leaks in Outbound Web Traffic," Proceedings of the IEEE Symposium on Security and Privacy, May 2009.

[18] Daniel J. Solove, "A Taxonomy of Privacy," University of Pennsylvania Law Review, Vol. 154, No. 3, p. 477-560, January 2006 GWU Law School Public Law Research Paper No. 129.

[19] Peter Eckersley, "How Unique Is Your Web Browser?," Proceedings of the 10th international conference on Privacy enhancing technologies , Springer-Verlag Berlin, Heidelberg, 2010, pp 1-18. [Online]. Available: https://panopticlick.eff.org/browser-uniqueness.pdf. [Accessed: Oct. 29, 2013]

[20] Gecko user agent string reference. [Online]. Available: https://developer.mozilla.org/en-US/docs/Gecko_user_agent_string_reference. [Accessed: Oct. 29, 2013]

[21] User agent string explained. [Online]. Available: http://www.useragentstring.com/Firefox20.0_id_19621.php [Accessed: Oct. 29, 2013]

[22] How do I turn on the Do-not-track feature. [Online]. Available: https://support.mozilla.org/en-US/kb/how-do-i-turn-do-not-track-feature. [Accessed: Oct. 29, 2013]

[23] How unique and trackable is your browser? [Online]. Available: https://panopticlick.eff.org/ [Accessed: Oct. 29, 2013]

[24] http://browserspy.dk/ [Accessed: Oct. 29, 2013]

[25] Benoit Baudry, Master internship: Browser fingerprint obfuscation through random reconfiguration, http://people.rennes.inria.fr/Benoit.Baudry/fingerprint-reconfiguration/ [Accessed: Oct. 29, 2013]

[26] Lejun Fan, Yuanzhuo Wang, Xiaolong Jin, Jingyuan Li, Xueqi Cheng, Shuyuan Jin, "Comprehensive Quantitative Analysis on Privacy Leak Behavior," www.plosone.org, Vol 8, Issue 9, September 2013.

[27] B. Krishnamurthy, K. Naryshkin, and C. E. Wills, "Privacy leakage vs. protection measures: the growing disconnect," In Web 2.0 Security and Privacy Workshop, 2011.
[Online]. Available: http://w2spconf.com/2011/papers/privacyVsProtection.pdf. [Accessed: Oct. 29, 2013]

[28] Carsten Eiram, "Vulnerabilities vs. attack vectors," [Online]. Available: http://secunia.com/blog/vulnerabilities-vs-attack-vectors-97. [Accessed: Nov. 13, 2013]