BLADE: Slashing The Invisible Channel of Drive-by Download Malware

Long Lu[1], Vinod Yegneswaran[2], Phillip Porras[2], and Wenke Lee[1]

[1] School of Computer Science, Georgia Tech, Atlanta, GA 30332 USA
[2] SRI International, Menlo Park, CA, 94025 USA
{long, wenke}@cc.gatech.edu, {vinod, porras}@csl.sri.com

Abstract. Drive-by downloads, which result in the unauthorized installation of code through the browser and into the victim host, have become one of the dominant means through which mass infections now occur. We present BLADE (Block All Drive-by download Exploits), a browser-independent system that seeks to eliminate the drive-by threat. BLADE prudently assumes that the legitimate download of any executable must result from explicit user consent. BLADE transparently redirects every browser download into a non-executable *safe zone* on disk, unless it is associated with a programmatically inferred *user-consent event*. BLADE thwarts the necessary underlying transaction on which all drive-by downloads rely, therefore it requires no prior knowledge of the exploit methods, and is not subject to circumvention by obfuscations or zero-day threats.

1 The BLADE System

Unlike push-based approaches adopted by Internet scanning worms and viruses, contemporary malware publishers rely on drive-by exploits for silent dissemination of spyware, trojans, and bots [3]. As a countermeasure, BLADE is a kernel-based monitor designed to block all malware delivered without user knowledge via browsers and overcomes the challenges described in [2].

BLADE's design is motivated by the fundamental observation that all browser downloads fall into either of two basic categories: supported file types (e.g., html, jpeg) or unsupported file types (e.g., exe, zip). While browsers silently fetch and render all supported file types, they must prompt the user when an unsupported-type is encountered. The objective of client-side download exploits is to deliver malicious (*unsupported*) content through the browser using methods that essentially bypass the standard unsupported-type user prompt interactions. BLADE's approach is to intercept and impose "execution prevention" of all downloaded content that has not been directly consented to by user-to-browser interaction. To achieve this, BLADE introduces two key OS-level capabilities:

(1) User-Interaction Tracking: A novel aspect of BLADE is the introduction of user-interaction tracking as a means to discern *transparent* browser downloads from those that involve direct user authorization. Operating from the kernel space, BLADE introduces a browser-independent *supervisor*, which infers user consent-to-download events, by reconstructing on-screen user interfaces (UI) from

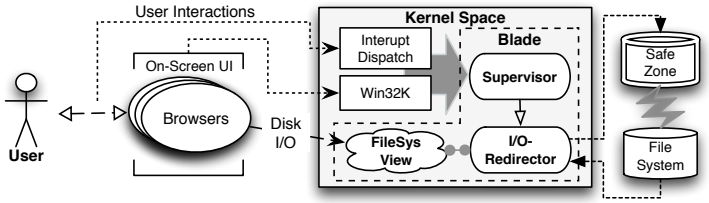

Fig 1. BLADE's Architecture

kernel memory and monitoring user interactions in the form of hardware interrupts. Specifically, it retrieves semantic UI information from the kernel objects maintained by the windowing subsystem (Win32K), discovers interested UI elements and their status changes (e.g., download confirmation dialogs), and listens to hardware-interaction events (e.g., mouse clicks) targeted at any interested UI element. Once a download consent event is inferred, the supervisor records it as an authorization along with the information parsed from UI elements (e.g., file names and URLs).

(2) Disk I/O Redirection: BLADE's *I/O-Redirector* transparently redirects all hard disk write operations to a *safe zone*. This safe zone, created and managed by BLADE, represents offshore storage inaccessible from the local file system. Being addressable only through BLADE ensures that files in the safe zone can never be loaded or executed, even by the OS. Upon finishing each file write operation, the *I/O-Redirector* queries the *supervisor* and maps the current file to the local file system if a stored authorization correlates with it. To maintain functional consistency, the supervised processes are also provided a modified file system view, which renders the impression that all disk writes are carried out in their respective locations, while actual disk I/O to these files are forwarded by BLADE to the safe zone. A prototype of BLADE is now under development as a kernel driver for Windows platforms, which will be tested with multiple versions of Firefox, Internet Explorer and Chrome.

Threat Model: We assume that the OS, the underlying hardware and network infrastructure are uncompromised. The attacker's goal is to perform a forced upload and execution of malicious binary content on the victim machine. Upon successfully hijacking control of a browser process, an attacker may pursue either of two possible paths to bypass BLADE and install a malware binary: (a) evading I/O redirection, or (b) executing the malware stored in the safe zone. As a kernel driver only dealing with trusted OS components and unforgeable hardware events (e.g., mouse clicks), BLADE is not subject to code injection or data manipulation attacks, and not deceived by fake UI messages which makes (a) difficult. Likewise, attempts to launch the malware from outside the browser process are naturally prevented as the the malware is only addressable through BLADE.

References

1. Drive-by downloads: The web under siege. Kaspersky Lab, `http://www.viruslist.com/en/analysis?pubid=204792056`, April 2009.
2. M. Egele, E. Kirda, and C. Kruegel. Mitigating drive-by download attacks: Challenges and open problems. In *iNetSec 2009*, Zurich, Switzerland, April 2009.

3. N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iframes point to us. In *Proceedings of the 17th USENIX Security Symposium*, 2008.