**Social engineering** is the act of [manipulating](http://en.wikipedia.org/wiki/Psychological_manipulation) people into performing actions or divulging confidential information, rather than by breaking in or using technical cracking techniques.[[1]](http://en.wikipedia.org/wiki/Social_engineering_%28security%29#cite_note-0) While similar to a [confidence trick](http://en.wikipedia.org/wiki/Confidence_trick) or simple [fraud](http://en.wikipedia.org/wiki/Fraud), the term typically applies to trickery or deception for the purpose of information gathering, fraud, or computer system access; in most cases the [attacker](http://en.wikipedia.org/wiki/Attack_%28computer%29) never comes face-to-face with the victim.

60% are drive by downloads says google

Download without authorization

Properties:

Browser agonistic

Check – sandboxing

Goal of Drive by

The goal of the

drive-by exploit is to take effective, temporary control of the client

web browser for the purpose of forcing it to fetch, store, and then

execute a binary application

Shellcode injection phase:

Shellcode execution phase

Covert binary install phase:

Although scenarios where

the assumed attacker can remotely exploit a kernel vulnerability

via a browser exist, which are out of the scope of this model, we

argue that they are extremely rare and could be addressed by integrating

orthogonal OS integrity protection technologies, such as

hypervisor-based protections,

<http://www.youtube.com/watch?v=9emHejh8hWE>

Drive by download

Browser agonistic

Zero- day attack

Code obfuscation

Identity theft

Denial of service attack

Secure zone

The sandbox typically provides a tightly-controlled set of resources for guest programs to run in, such as [scratch space](http://en.wikipedia.org/wiki/Scratch_space) on disk and memory. Network access, the ability to inspect the host system or read from input devices are usually disallowed or heavily restricted. Ex: Adobe Flash, Silverlight (small window, restricted user inputs)

How automatic update of browsers is not affected?

Browsers use a technique called “Native client” to install its own update codes. In that case the update code relies on its own preinstalled browser as a client rather relying on Operating system to install a piece of code. This applies for the installation of Plug-ins too.

Not mentioned in paper:

What if a download that tries to spoof the identity and attempts to gain access at the same time there has been a download with the same name that has authorized by the user?

Advantage:

Zero false positive

3% Worst case performance cost.

Surreptitious delivery exploits, which are the focus of this paper,

represent a particularly insidious form of exploit, whereby the

mere connection to a web server can result in the installation of

malware on the client machine.

Rather, the malicious web server silently passes malicious

shellcode to the victim browser, which then forces the browser to

download, store, and silently execute a malicious application

Blade Objectives and Challenges:

User authorization capture

Correlation between authorization and download content

Enforcement on Execution prevention

Browser agonistic: just because new updates are potential Zero day vulnerable.

Evasion Independence: Immune to all kind of evasion techniques such as Code Obfuscations, Zero day attacks

BLADE Architecture:

Screen Parser

Supervisor

Hardware event tracer

Correlator

IO redirector

(URL, Path)

Secure zone Blocking all memory-section synchronizations

By default secure zone: blocking all memory section syncs. – Only the meta data is done so. Not the original data. Done in constant time.

Screen Parser:

Worst way of doing: Direct hook into windowing event handlers. Significant performance degradation in drawing the window results in IU delay.

Good approach: UI Signatures. There is only a handful of UI Signatures among browsers.

Keystrokes are obtained as associated attributes.

Fake user consent dialogue may obtain a signature match but it cannot evade from correlator.

Notifies status changes.

Supervisor:

Monitors all browser processes.

Authorizing downloads. (Not done until hardware event tracer finds a user event)

Super visor adds process p to the list when:

Newly created browser process

Remote thread is created within the process by supervised process

This one is critical to prevent I/O redirection evations.

Correlator:

Establishes 1-1 mapping download authorizations and downloaded files.

Associates each file with each TCP transactions. Recal (URL, Path) tuple.

URLs are solved into IP addresses by DNS lookup.

Stream log is searched for particular (candidate file, IP address) pair

**I/O Redirector:**