Measurement Research to the Web Calamity's Rescue

Gregory BLANC

Internet Engineering Laboratory
Nara Institute of Science Technology
WIDE member

3rd CAIDA-WIDE-CASFI Measurement Workshop
April 24-25, 2010, Osaka

What measurement does?

- CAIDA: malicious activity analysis, traffic classification, data sharing
- CASFI: performance measurement, traffic analysis, data sharing
- WIDE-mawi: DNS behavior analysis, traffic measurement, data sharing
- overall, deploying probes at the network layer and measuring traffic characteristics

What measurement does? (from the leaders)

- Kenjiro CHO ~ “AJAX generates a lot of traffic”
- Brad HUFFAKER ~ “HTTP is king”
- Sue MOON ~ “The Web admin left”

What measurement can do?

- distinguishing application won’t help
- we need to look deeper in the application layer
- draw statistics of what is actually flowing
- collect samples of what interests us

Common Issues in Web Security Research

- we often encounter issues when evaluating proposals (systems):
  - lack of datasets: nothing to play with
  - homogeneous datasets: too much of the same thing
  - outdated datasets: remember the KDD Cup 1999?
  - unbalanced datasets: might not be representing the reality

Existing methods to collect JS samples (1): crawling

- merits
  - automated
  - can collect loads of data
- demerits
  - do not understand AJAX
  - can not mimic accurately the user
  - target site should be wisely chosen
    - JS may represent a small percentage
  - solution: targeting blacklisted websites
    - user contribution
  - Example:
    - crawler.archive.org

Existing methods to collect JS samples (2): analysis website

- merits
  - only malicious JS
  - often deobfuscated
  - available online
- demerits
  - size depends on user contribution
  - dataset is not enough varied
  - data is not always available

- solution: to encourage sharing
  - but it will be limited to what users would want to contribute
- Example
  - wepawet.cs.ucsb.edu
  - jsunpack.jeek.org

No solution in the wild (1)

- we do not capture malicious JS because it is volatile in nature:
  - volatileness
  - obfuscation
  - transience
  - duplication
  - redirection
  - application layer
  - silent bidirectional communication

No solution in the wild (2)

- no efficient crawlers
- no attractive sharing platforms
- small user contribution
- new ways to get samples in the wild:
  - network probes with deep packet inspection -> overhead
  - browser monitoring -> privacy
  - logs

JS measurement

- what to measure? is it measurable?
  - degree of obfuscation of benign Web 2.0 traffic: obfuscation does not indicate maliciousness
  - spread of JS malware: Samy was fast but noisy
  - JS malware code collection: overall lack of reliable datasets

Web 2.0

- not only a buzzword
- paradigm shift:
  - shift in the development
  - shift in the usage

Development Shift

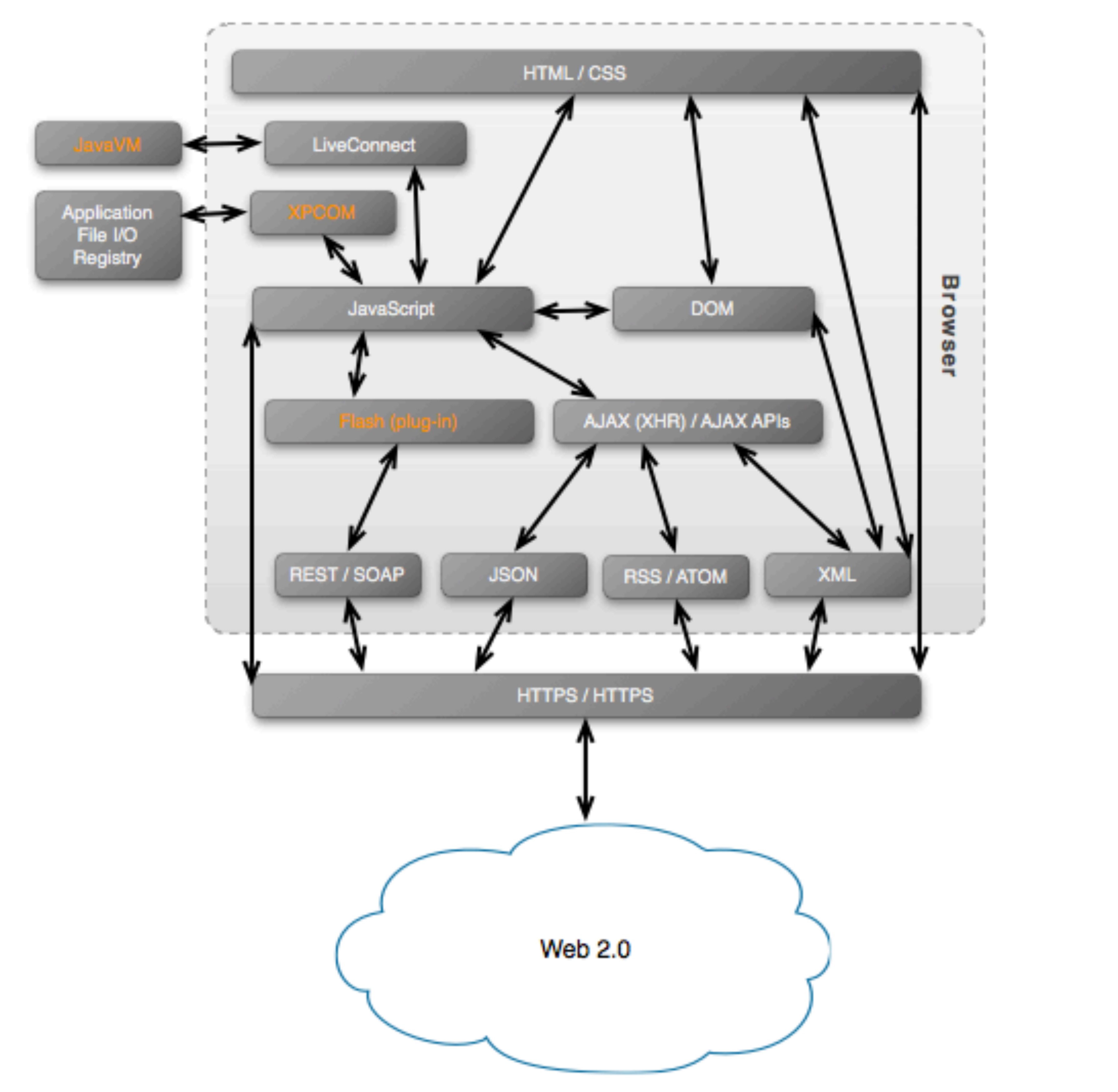
- Rich Internet Applications (desktop)
- Asynchronous Communication
- Cross-domain Interaction
- Web Services

Usage Shift

- Software Consumption
- Collaboration/Participation
- Content Sharing
- Syndication/Aggregation
- Social Networking

Browser Model Shift

- To cope with the Web 2.0 offer, the browser model has also changed:
  - plugins (Flash)
  - APIs (Ajax, custom, etc.)
  - interconnection (ActiveX, JavaVM)


HTML / CSS
JavaVM
LiveConnect
Application
File I/O
Registry
XPCOM
JavaScript
DOM
Browser
Flash (plug-in)
AJAX (XHR) / AJAX APIs
REST / SOAP
JSON
RSS / ATOM
XML
HTTPS / HTTPS
Web 2.0

User is the new victim

This new browser model provides a better user experience but provides the attacker with a wider attack space

- server side: too many websites with too many inputs to validate or control
- client side: the user is left defenseless even against deemed benign popular sites

Attackers prefer to concentrate on the most vulnerable, the end-user: phishing, drive-by attacks,etc.

JS malware (1)

- JS is a dynamic prototype-oriented event-drivent scripting language
  - a good tool to program automated elaborated script that can do massive harm
  - JS malwre: observed and defined by some security researchers (Brian Hoffman, Jeremiah Grossman, Martin Johns, etc.)

JS malware (2)

- propagates like conventional malware
- wide category regrouping JS-based malicious code
- PoC: XSS tunnel/proxy/botnet
  - in-the-wild examples: BeEF, BrowserRider, XSS-proxy, Samy worm, Yamanner

Strengths of JS Malware

- 1) stealth: property of going unnoticed by the user and the server
    - use of the XHR object
- 2) polymorphism: ability of changing its form dynamically to evade signature
    - use of prototype hijacking
- 3) obfuscation

JavaScript Analysis

- dynamic execution [Moshchuk'07]
- static/dynamic tainting [Vogt'07]
- control flow graph [Guha'09]
- semantics [Hou'08]
- machine-learning based [Choi'09, Hou'10, Likarish'09]

JavaScript Deobfuscation

- manual deobfuscation
- semi-automated (Malzilla)
- anti-analysis tricks:
  - recursive obfuscation
  - anti-crawling traps
  - argument.callee

Conclusion

- Our research area suffers a great lack of reliable and representative data
- We have the methods and tools to carry out analysis but no data
- Measurement research has made progress not only on collection but also on efficiency
- It is time to cooperate!

Overture

- JavaScript is not the only matter of concern
  - VBScript, ActionScript (Flash)
  - new media of propagation (SNS)
  - distribution websites structure

Questions / Discussion

- Thank you for your attention

- Let’s start a cooperation: gregory@is.naist.jp

References

- [Moshchuk'07]: SpyProxy: Execution-based Detection of Malicious Web Content, USENIX Security'07
- [Vogt'07]: Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis, NDSS'07
- [Hou'08]: Malicious Webpage Detection by Semantics-Aware Reasoning, ISDA'08
- [Choi'09]: Automatic Detection for JavaScript Attacks in Web Pages through String Pattern Analysis, FGIT'09
- [Guha'09]: Using Static Analysis for Ajax Intrusion Detection, WWW'09
- [Likarish'09]: Malicious Javascript Detection Using Classification Techniques, MALWARE'09
- [Hou'10]: Malicious Web Content Detection by Machine Learning, Expert Systems with Applications #37