Empowering Browser Security for Mobile Devices Using Smart CDNs

Benjamin Livshits and David Molnar
Microsoft Research

Abstract

There has been a great deal of attention on browser security in recent years. However, the majority of projects in this space have focused on security of *desktop browsers*, while it is likely that it is the *mobile browsers* that will be targets of security attacks in the coming years. In this paper we propose the use of "smart CDNs" to quickly drive security innovations into the mobile browser space.

1 Motivation

The mobile Web is here to stay, but the mobile Web browser must change dramatically. The rise of "mobile-only" Web users, the limitations of today's wireless technology to provide peak bandwidth, pressure on battery life, and the decreasing cost of storage point to a future where browsing is increasingly split between the mobile client and a location-aware "smart content distribution network." Such a network would have has a low-latency connection with the mobile client and would be able to perform some computation, with the heavy lifting left to be done by the data center. One of the main benefits is pushing data towards the end user, reducing latency due to the speed of light and working around wireless bandwidth limitations. This split raises new security issues and opens a golden opportunity for researchers to drive security innovations in mobile Web browsers.

The iPhone is the leading edge of these trends. Today's entry-level iPhone comes with 8 GB of storage, has both a 3G and a 802.11 radio, and may be periodically synced with a PC that has close to a terabyte of storage. iPhone usage has driven 3G data traffic to unprecedented levels; AT&T reported that traffic in the San Francisco Bay Area increased by $2,000$ percent, leading to many dropped calls and a temporary suspension of iPhone sales in the region. Subsequently AT&T announced a $65 million upgrade to its infrastructure to accommodate these users [1], with other mobile carriers following not far behind. Much of this data usage is mobile Web *browsing*, as opposed to other data use, while an increasing amount is video [2]. Much of this content is already served by CDNs, but without tight integration with the client: the content distribution network Akamai pushes a peak of 3.45 Tbps of web traffic, including bilions of requests for video streams [13].

Security Research Trends. Recent mobile worms affecting the Nokia/Symbian platform amply demonstrate that mobile devices are far from impervious when it comes to security compromises [6]. Recent trends in security research towards merging the browser and the operating system address such problems. Chrome's process isolation and secure extension architecture is one example, as of course is ChromeOS. In addition to this, security researchers have been hard at work improving the security of traditional desktop browsers. Some examples of security-related technologies include XSS filters, phishing filters, and other end-host protection technologies recently proposed in research literature such as Nozzle [16], Ripley [17], Gatekeeper [8], AdSafe [4], Caja [15], ConScript [14], or URL reputation services.

Despite this rapid innovation, mobile devices rarely benefit. The devices themselves have serious limitations compared to desktop PCs. Even if "fat" devices such as the iPad or Nexus One become commonly available, there will still be a large number of less expensive devices at lower price points with lesser capabilities. Furthermore, software is difficult to upgrade on many mobile devices,

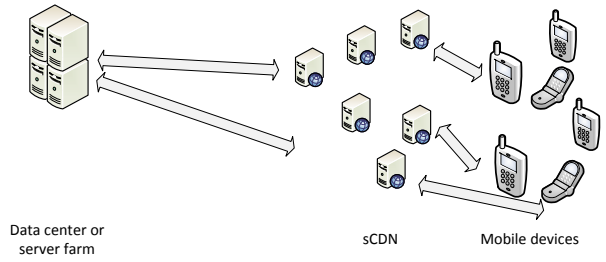

Figure 1: Mobile devices fill their pre-caches from sCDN middle tier nodes, including laptops, 802.11 access points, boxes located with the mobile carrier's network, or other devices. The data center is the canonical source of a page.

even relatively painless ones such as the iPhone. A few shipping systems, such as the SkyFire browser or BlackBerry, attack these limitations by splitting the browser between a thin client and a "cloud" that performs much of the work. For example, SkyFire can display Flash content by pre-rendering bitmaps on its servers. The Achilles' heel of these approaches is typically the latency incurred due to network traffic: BlackBerry combats this with a focus on email and enterprise-local deployment of BlackBerry servers, but SkyFire suffers compared to unsplit browsers. Opera Mini is another shipping system that uses a proxy to pre-render web pages for mobile devices.

Proposal. Our proposal is to push security into a smart content distribution network (sCDN) that runs a powerful — and secure — desktop browser, while leaving a thin client on the mobile device. Recent publicly announced numbers show that Microsoft's sCDN serves 40% of all content, which is estimated to rise to 60% by 2010 [3]. The specific details of the split will depend on the device, carrier, and use cases: in this position paper we focus on security enhancements enabled by the split.

1.1 Motivating Examples

Below we discuss two motivating examples of sCDN use.

Heap Spraying Defense. Heap spraying is a commonly used technique in exploits for platforms that run JavaScript, such as Web browsers, including Internet Explorer, Firefox, and Safari. The highly publicized attacks on Google and other companies in China allegedly involved malware that used a heap spraying attack [10]. Nozzle is an end-host technology for detecting and preventing these attacks [16]. While Nozzle has been successfully applied to IE, Firefox, and Acrobat Reader, it has not been applied to mobile browsers. Worse, mobile platforms do not today have OS-level protections such as address space layout randomization that would resist heap spray attacks. While ASLR for Android has been demonstrated, deploying it requires a software install.

Imagine a mobile browser going to `http://evil.com`, a page containing a heap spraying attack. In our scenario, we would be running a Nozzle-protected version of the browser on the sCDN (possibly, enclosed in a VM that is periodically refreshed). Both the mobile browser and the sCDN-based browser will start rendering the URL in question. Once the sCDN-based browser detects a spraying attempt, it will signal to the mobile device to stop rendering the page. Alternatively, it could reboot the mobile device or restart its browser. If we opt for a model in which the sCDN browser is in charge of rendering the contents and sending bitmaps over to the mobile client, the chance for exploitation on the mobile device is greatly reduced.

In the example above, one can imagine a variety of other detection technologies such as behavior-based intrusion detection techniques being substituted in place of Nozzle. New technologies can be deployed merely by upgrading the sCDN.

Request Prediction and Bandwidth Shifting. The key metric for a carrier is the amount of peak bandwidth required, because adding infrastructure to support higher peak use incurs substantial capital costs (e.g. adding new cell towers or new cables). The sCDN can provide a *pre-cache* of content that is pushed opportunistically to the mobile device based on predictions of the user's future requests. Each successful prediction results in shifting bytes from peak bandwidth to off-peak bandwidth, thereby smoothing the load on carrier infrastructure and creating value for the carrier. Even better, a successful prediction reduces latency for the user's requests. Figure 1 shows an architecture diagram for this scenario.

2 Proposal

We propose a *smart content distribution network* or *sCDN* to improve mobile Web browsing. A sCDN is a network of servers, which may include "local" infrastructure

such as home PCs or 802.11 access points, that perform pre-rendering and pre-caching of data for a mobile Web browser. The server-side piece of the sCDN can run a headless variant of a desktop Web browser specifically instrumented for sCDN use.

The sCDN can also pre-render the site for a particular phone. These pre-rendered versions may also be shared across the users (consider `nytimes.com` or `google.com`). Because the sCDN knows about the user, including browsing history, favorite sites, and potentially even passwords and deep Web material, the sCDN can then perform predictive fetch and pre-rendering of user requests. Mobile device state can be backed up on the sCDN to enable seamless migration from one device to another.

Pre-caching can also reduce radio transmission requirements. The large memory on the device can simply be updated when the device is in WiFi range, during off-peak hours, or when plugged into a fast wired connection. The device may not need to send requests except for rare cases of requesting a new URL that was not predicted and pushed to the device. The following issues present themselves.

- How do we manage the privacy impact of the sCDN collaborating with the browser?
- How do we faithfully replicate a range of mobile Web browsing environments in a distributed (virtualized) environment? Working with the browser here allows the sCDN to avoid today's unsavory hacks such as sniffing User-Agent strings.
- The infrastructure itself could fall a victim to a worm that propagates through sCDN and all connected mobile devices or a cache poisoning attack that affects the common cache.
- The shared caching infrastructure can be used to avoid paying for clicks. This emphasizes how desirable it is to co-design the sCDN and the mobile browser.
- How do we cache and authenticate dynamic content? While static content can be easily fingerprinted and signed by the origin, dynamic content, especially when generated for each user, presents significant challenges not yet overcome.
- How do we handle time-sensitive data, such as news, that should not stay long in a cache?

While there are many left to work out, we believe the sCDN architecture offers substantial benefits. Users choose the sCDN for its other benefits, but receive security "bundled in."

The sCDN benefits from security mechanisms because they lower support costs arising from user security incidents, or because they reduce risk to the sCDN from compromised middle tier nodes. Solving the security issues involved with partially trusted sCDN nodes might even reduce the capital cost to build a sCDN by allowing the re-use of existing equipment (e.g. home PCs).

2.1 sCDNs and Security Innovations

We see two broad classes of security innovations that particularly benefit from sCDNs. First, in-browser enforcement mechanisms, such as Nozzle, StackGuard, or RandomHeap, usually come with a performance cost. This translates directly to reduced battery life on a mobile device. With a sCDN, the cost of such mechanisms is shifted to the server instead.

The second class of mechanisms are intrusion detection mechanisms. The key is shared data gathering: an attack on one client of the sCDN can potentially alert all clients of the sCDN to the danger. Examples of such mechanisms include blacklists, phishing filters, URL reputation schemes, XSS filters, and real-time intelligence on attack propagation. An example of how this kind of collaborative intelligence is helpful is stopping worm propagation across a social networking site such as MySpace: as has been suggested in the Spectator project [12], the ability to observe traffic to and from a range of users will enable correlating related requests and responses and stopping a worm quickly.

3 Threats to the sCDN Scenario

We now describe several potential problems that could prevent sCDNs from becoming a reality, or if they do become common, could negatively affect security integration.

Capital cost. To effectively cover current 3G service areas would require an investment in infrastructure of the same order of magnitude as that required to construct cell towers and network infrastructure. Less obvious but no less important is the cost to solve the technical problems associated with rendering active content,

frequently changing content, and pushing content to the nearest sCDN node for a user. While proposals exist in the research community, if security mechanisms increase this capital cost substantially, either by complicating the technical solutions or by requiring more infrastructure, they may not be deployed.

Infrastructure dependency. This architecture creates a dependency on the sCDN for support, maintenance, availability, and upgrades. The sCDN may also go bankrupt, taking the user's data with it.

Privacy. The sCDN sees all URLs queried by a user, all search terms, and possibly all cookies and passwords. How can we overcome this privacy issue?

Disruptive Technological Change. Finally, our assumptions may fail. A breakthrough in wireless technology or battery life would remove the constraints that push towards sCDNs and make the threats we have identified more serious.

4 Related Work

The Infostations project showed how devices with intermittent access to fat, cheap connectivity could use prefetched caches to support mobile operation [7]. Predicting user requests to save bandwidth also appears in the TIERStore and DiSc projects aimed at developing regions with poor WAN connectivity [5, 9]. The SONGO work shows that search queries can be successfully predicted and cached [11]. Traditional content distribution networks such as Akamai [13] focuses on static content, while the SkyFire browser works with active content such as Flash.

5 Conclusions

Smart content distribution networks (sCDNs) are an architecture direction that enables rapid innovation in security mechanisms and fits with today's trends in mobile communication. We believe that security researchers should seriously consider the role of sCDNs to impact the security of the mobile Web.

References

[1] AT&T. AT&T invests nearly $65 million through 2009 to strengthen 3g wireless coverage in SF bay area, 2009. http://www.att.com/gen/press-room?pid=4800&cdvn=news&newsarticleid=27561.

[2] Cisco, Inc. Cisco visual networking index: Global mobile data traffic forecast update, 2009-2014. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html, Feb. 2010.

[3] J. Cohen. Meeting microsofts content delivery needs. http://www.streamingmedia.com/east2009/presentations/CDNSummit09-Keynote-Microsoft.pdf, 2009.

[4] D. Crockford. ADSafe. adsafe.org.

[5] M. Demmer, B. Dui, and E. Brewer. Tierstore: A distributed filesystem for challenged networks in developing regions. In *USENIX Technical Conference*, 2008.

[6] Fortinet, Inc. Worm:symbos/yxe. http://www.fortiguard.com/encyclopedia/virus/symbos_yxes.a!worm.html, Feb. 2009.

[7] R. Frenkiel, B. Badrinath, J. Borras, and R. Yates. The infostations challenge: Balancing cost and ubiquity in delivering wireless data. *IEEE Personal Communications*, 7(2):66–71, 2000.

[8] S. Guarnieri and B. Livshits. Gatekeeper: Mostly static enforcement of security and reliability policies for JavaScript code. In *Proceedings of the Usenix Security Symposium*, Aug. 2009.

[9] R. Honicky and E. L. Miller. Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS)*, 2004.

[10] J. Kelley. How Google was pwned: In-depth look into the aurora attacks. http://www.slideshare.net/NEOISF/how-google-was-pwned-indepth-look-into-the-aurora-attacks, 2010.

[11] E. Koukoumidis, D. Lymberopoulos, J. Liue, and D. Burger. Improving mobile search experience with SONGO, 2010. MSR TR 2010-15.

[12] B. Livshits and W. Cui. Spectator: Detection and containment of JavaScript worms. In *Proceedings of the Usenix Annual Technical Conference*, July 2008.

[13] O. Malik. Akamais network now pushes terabits of data every second, 2009. http://gigaom.com/2010/04/11/akamai-3-4-terabits/.

[14] L. Meyerovich and B. Livshits. ConScript: Specifying and enforcing fine-grained security policies for Javascript in the browser. In *IEEE Symposium on Security and Privacy*, May 2010.

[15] M. S. Miller, M. Samuel, B. Laurie, I. Awad, and M. Stay. Caja - safe active content in sanitized JavaScript, October 2007. http://google-caja.googlecode.com/files/caja-spec-2007-10-11.pdf.

[16] P. Ratanaworabhan, B. Livshits, and B. Zorn. Nozzle: A defense against heap-spraying code injection attacks. In *Proceedings of the Usenix Security Symposium*, Aug. 2009.

[17] K. Vikram, A. Prateek, and B. Livshits. Ripley: Automatically securing distributed Web applications through replicated execution. In *Conference on Computer and Communications Security*, Oct. 2009.