Divide and conquer: the threats posed by hybrid apps and HTML 5

Steve Mansfield-Devine

Steve Mansfield-Devine

The distinction between online and offline is becoming increasingly blurred. From Google Docs to iPhone apps, our interaction with the web has grown from simple browsing to complex interactions. There is a breed of hybrid applications that reside partly on your device – whether it's a PC or a smartphone – and partly on the net. Both application code and data may be split. But this raises some concerns over how you manage security in this schizophrenic environment.

This trend is only likely to accelerate with the advent of devices like the Apple iPad, which moves away from the concept of a computer being a general-purpose device to being geared more towards specific applications, many of which have an online component. And underlying this movement to hybrid applications is the development of HTML 5.

More than just tags

HTML 5 isn't just a bunch of new tags. It provides new mechanisms for exchanging and storing data, as well as presenting video, audio, animations, typefaces and complex layouts.

We've had technologies like this before. Google Gears and Adobe Air both address the current limitations of web technology by creating environments in which code is run on the client. And both have raised security concerns. It's likely that Gears will fade away, given Google's dedication to using HTML 5. The fate of Air, and even Flash, is anybody's guess. It's notable, however, that Apple chose not to support Flash natively on the iPad, and also seems to be getting behind HTML 5.

As ever, the concern isn't the technology itself, but how people use it. It will be some years before we see a fully ratified specification for HTML 5. Current estimates are 2012 for the W3C Candidate Recommendation stage, and some pundits add several more years before all the kinks are worked out.

However, the first working draft was published on 22 January 2008, with work having started as far back as 2004.[1] Consequently, elements of HTML 5 are already supported in the major browsers and some organisations – Google especially – are making extensive use of them.

The worry, then, is that developers will rush to exploit these great new features without fully understanding, let alone addressing, the security implications. In its *2010 Threat Predictions report*, McAfee claims that as a result of the technological advances of HTML5, "the web will undergo a dramatic upgrade that will change the way web application developers and hackers are able to interact with their 'target market'."

HTML 5 security

That's the bad news. The good news is that HTML 5 boasts a number of new security features. Many of these are geared to tackling the problems that arise when a web page includes elements from other sources. A classic example, and one that has been exploited many times in the past, is the iframe. An iframe allows the embedding of content from a source different to that of the main page. Many advertisement services serve content this way. Alas, miscreants have used iframes to inject malicious code, including the kind used for drive-by infections.

"The good news is that HTML 5 boasts a number of new security features. Many of these are geared to tackling the problems that arise when a web page includes elements from other sources"

With HTML 5 comes a new mime-type – text/html-sandboxed – to be used with a sandboxed version of the iframe. This can be used to prevent the iframe:

Accessing the DOM of the parent page.

- The iframe has no access because it has a different origin.
- Executing scripts.
- Embedding forms, or manipulating other forms on the page.
- Reading or writing to cookies, local storage devices or SQL databases.

The sandboxed iframe does not inherit the full privileges of the parent page. What's more (and this already works in many browsers), content called within a sandboxed iframe must be served with the text/html-sandboxed mime type; but the browser will not render such content by itself, as a standalone page. It is only rendered as part of the parent page (exact details of how this is done varies from browser to browser). The upshot of this is that it is more difficult for malicious sites to lure users to navigate to untrusted content.

Even with this kind of capability, some commentators still regret the continued existence of the iframe in HTML 5. The <frame> tag has been dropped, which may slightly reduce incidents of clickjacking. But <iframe>, many believe, will remain a source of exploits.

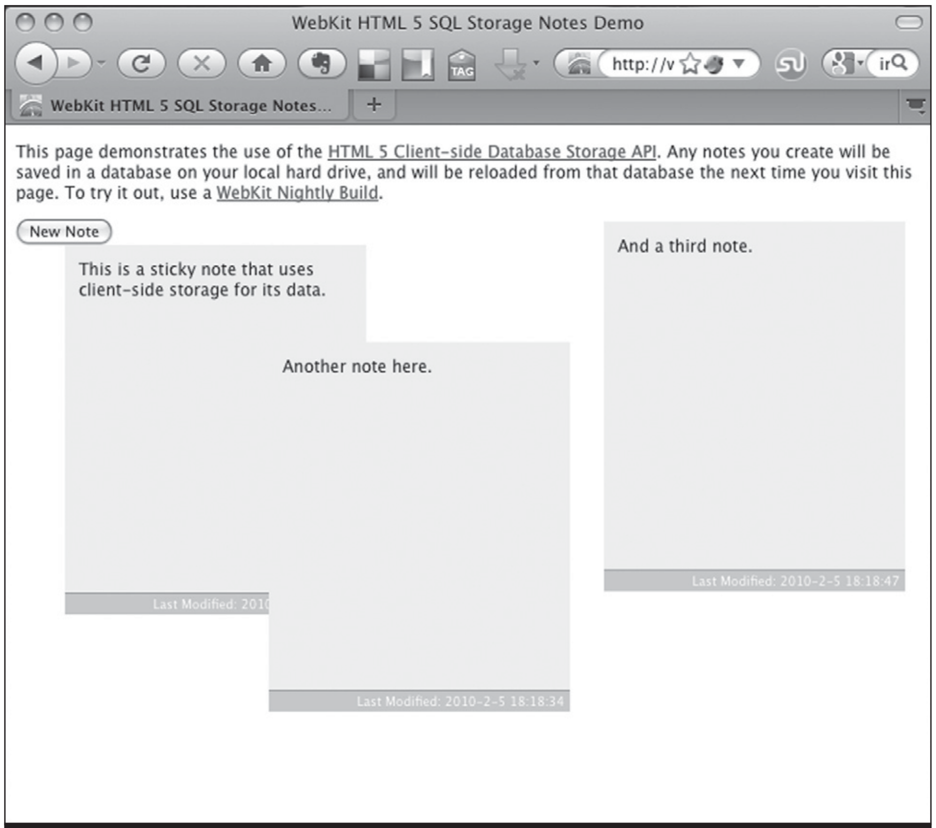
Client-side storage

HTML 5 also provides mechanisms for the client-side storage of information that go far beyond the humble cookie.[2] While cookies are restricted to 4K, or more lately up to 10K, of data, the new HTML 5 features are capable of handling megabytes, some of it stored in SQL database tables.

Client-side storage isn't new. Microsoft added the userData function to Internet Explorer back in version 5.5[3]. The Adobe Flash Player 6 plug-in introduced the Local Shared Object. And Google Gears also provides for local storage. With HTML 5, however, there will be a more consistent, homogeneous and widespread system – just the kind of environment malware writers like because it increases their chances of success.

With HTML 5, this DOM Storage functionality is provided by the Storage Interface.[4] This includes objects such as sessionStorage, which persists as long as the browser is open but is erased when the session ends, and localStorage, which remains on the disk between browser sessions.[5] As well as providing for larger amounts of data than a cookie can manage, these mechanisms do not require the data to be sent with every page load or refresh.

The immediate concern that springs to mind is cross-site scripting (XSS). Domain restrictions and sandboxing are used as defensive mechanisms but to what degree of effectiveness is yet to be tested. With data held in SQL databases, there's also the potential for SQL injection attacks on the client side. There may also be opportunities for cross-directory attacks, as HTML 5 storage lacks the directory restrictions enjoyed by cookies.


WebKit HTML 5 SQL Storage Notes Demo
http://v
WebKit HTML 5 SQL Storage Notes...
New Note

This page demonstrates the use of the HTML 5 Client-side Database Storage API. Any notes you create will be saved in a database on your local hard drive, and will be reloaded from that database the next time you visit this page. To try it out, use a WebKit Nightly Build.

This is a sticky note that uses client-side storage for its data.

Last Modified: 2010

Another note here.

Last Modified: 2010-2-5 18:18:34

And a third note.

Last Modified: 2010-2-5 18:18:47

Figure 1: WebKit's sticky notes demonstration using client-side storage.

For a demonstration of how WebKit-based browsers can use offline storage, try WebKit's demonstration of 'sticky' notes that survive between browser sessions.[6] It's worth noting that my version of Firefox (3.5.7) popped up a warning, "This website (webkit.org) is asking to store data on your computer for offline use" and allowed me to choose whether to approve this. However, the warning isn't very scary, and we know how users ignore warnings.

Cross origins

The concerns don't end there. Amichai Shulman, CTO at Imperva, believes that easy control scripting of input and output devices might open up new fraud opportunities. He also points to HTML 5's support for 'cross origin' communications capabilities between elements from different domains.

"So far, the 'same origin' policy of the browser was the one thing that protected sensitive applications from being ripped apart by HTML coming from other, less trusted servers accessed by the user," he says. "In HTML 5, this barrier can be crossed by design. Using 'message' events on which a page is listening, an interaction between two pages coming from different domains is possible. If not carefully configured and implemented, this set of features would allow attackers to have their code access and manipulate sensitive applications and their sensitive data very easily."

Bad programming

The real worry is that, so far, different browsers have implemented HTML 5's capabilities in slightly different ways. For all that this is meant to be a standard, browser programmers and website developers seem to be in a rush to use the new features in the field. That's fine so long as they are implemented properly, but that isn't always the case.

"It's not so much the language you use but the way the application is written that will introduce the problems that might be exploited," says Matt Watchinski, a member of the Sourcefire Vulnerability Research Team in the US. "The concern would be, of course, that anything that shares space on your machine with space on the internet would be a higher priority target since it might yield richer crops for the attacker."

For example, input tags in HTML 5 have validation capabilities based on regular expressions. That alleviates the need for reams of Javascript when, for example, checking that a field isn't left empty, or that an email address is in the right format. However, it's crucial that web programmers also carry out server-side data validation, as client-side validation is so easily subverted. That's always been the case: yet we know from experience that lazy, rushed or incompetent programming can lead to gaping security holes, and this new feature of input tags – invaluable as it is to those who know what they're doing – may encourage poor practices among the less-gifted members of the programming community. And it adds an extra twist: regular expressions are notorious for giving programmers headaches. It's easy to make a mistake with a regex, with unpredictable consequences.

"So far, the 'same origin' policy of the browser was the one thing that protected sensitive applications from being ripped apart by HTML coming from other, less trusted servers accessed by the user"

Similarly, the protections provided for client-side storage, such as domain restrictions, might be easily eroded by poor programming.

Taking the shine off Chrome

Arguably the most extensive use of HTML 5 is by Google. Its Chrome browser supports many elements of the language and it will feature large in the company's forthcoming Chrome OS. Google's approach with this OS is that the browser *is* the operating system. Chrome OS (or Chromium OS in its currently available, open-sourced guise) is essentially the Chrome browser running on a Linux kernel. Most apps will be hybrid or web-based.

Both browser and OS, then, use a similar approach to security, with an emphasis on sandboxing. Process sandboxing features include:

- Mandatory access control, limiting resource, process and kernel interactions.
- Control group device filtering and resource abuse constraint.
- Chrooting and process namespacing for reducing resource and cross-process attack surfaces.
- Media device interposition to reduce direct kernel interface access from Chromium browser and plug-in processes.

Other features involve 'toolchain hardening', which places limits on how reliably exploits can work, kernel hardening and a number of file system restrictions, such as a read-only root partition.

The fact is that HTML 5, the Chrome browser and Chrome OS are packed with security features. And yet, the introduction of new capabilities will always increase the attack surface. When Gears was introduced, it was criticised for providing both XSS and (thanks to client-side storage) SQL injection vulnerabilities. Google Wave is built on top on the Extensible Messaging and Presence Protocol (XMPP), which gives Wave its data federating capabilities. XMPP is hardly new – it originated with Jabber – but Google's 'enhancement' of it with Wave is being seen by some as the next-generation technology for decentralised botnet command and control servers. This would give such a CnC server immense resilience.

McAfee's 2010 Threat Predictions said: "This arrangement will give cybercriminals another weapon to create redundancy in their networks and make them even more difficult to disrupt. With Twitter also serving as a control vehicle for botnets, the trend of botnets riding on top of commonly used applications and protocols will continue to make botnet communication traffic more difficult to detect and prevent."

The potential problem with HTML 5 and other technologies that underlie hybrid and web apps is not that they lack security features or that no consideration has been given to security implications. It's that many of these features will be in use – not necessarily by people skilled in creating secure applications – before we fully understand how they might be misused.

References

1 W3C HTML 5 Working Draft, W3C, January 2008 <http://www.w3.org/TR/2008/WD-html5-20080122/>

2 HTML 5 client-side storage, W3C, February 2010 <http://dev.w3.org/html5/webstorage>

3 Microsoft Internet Explorer userData, Microsoft, Accessed March 2010 <http://msdn.microsoft.com/en-us/library/ms531424(VS.85).aspx>

4 HTML 5 client-side Storage object, in W3C HTML 5 Working Draft, W3C, january 2008 <http://dev.w3.org/html5/webstorage/#storage-0>

5 DOM Storage – Mozilla Developer Center, Mozilla, Accessed March 2010 <https://developer.mozilla.org/En/DOM:Storage>

6 WebKit sticky notes demonstration of client-side storage, Webkit, Accessed March 2010:< http://webkit.org/demos/sticky-notes/index.html>

Resources

1 Microsoft Internet Explorer 8 Introduction to DOM Storage, Microsoft, Accessed March 2010 <http://msdn.microsoft.com/en-us/library/cc197062(VS.85).aspx>

2 W3C HTML5 A vocabulary and associated APIs for HTML and XHTML, W3C, March 2010 <http://dev.w3.org/html5/spec/Overview.html>