Security Reading Group:
Detecting In-Flight Page Changes with Web Tripwires

Nabil Schear
June 16, 2008

We’ve all heard about in-flight modifications, but now we know they are a reality. So who would care about this problem and how much? Well, you should because it can affect your security and privacy even if it’s done for "good intention" reasons.

Some brief stats from the paper that were noteworthy:

- ~1% modifications from 50,000 distinct IPs
- 70% were caused by client side proxies
- 46 were ISP added
- 125 were vulnerable
- 3 were malware (they didn’t properly escape characters in the URL comment, pretty common web vulnerability)

1 Pros

They were the first to bring up and study this. We found that the study part of this paper alone was an interesting enough contribution for a publication (though perhaps it would have been more suitable for another venue). Though they are not the first to do this, they did a good job of getting lots of testing by posting to slashdot and digg. It was also good that they evaluated different tripwire implementation strategies and discuss their shortcomings. We found the detailed categorization of modifications very enlightening. They were responsible enough to report all the vulnerabilities they found to the vendors well before the paper went out. They had a good structure for the evaluation section that helped understanding. Finally, we appreciated the direct applicability of this work to anyone’s day-to-day security.
2 CONS

The most glaring problem we found, especially from a security researcher’s point of view, is that the integrity guarantees they are using are weak. While they have gone to some length to prevent modification to the tripwire itself, an attacker can remove the tripwire (though it is unclear how easy or practical this is). The adversary can still do whole-page replacement.

In figure 3, the time for rendering is much longer for the tripwire version than for HTTPS. If you can amortize the cost of SSL setup over many pages (keep-alive HTTP), then encryption might be a faster option.

We discussed a bit about who wants modifications and who controls the web experience. It can be bad for web publishers because they may be losing ad revenue, and bad for users who don’t want extra stuff inserted into their traffic. Then again, content providers and ISPs may benefit from this because they can do better ad placement and have more revenue. The question we discussed is that providers and users have competing goals. For whom should the web experience be optimized? In the end, users will demand the service and ISPs will find a way to deliver it even if that means messing with the user’s experience. We also discussed whether users can realistically fight back against ISPs in any meaningful way to prevent this type advertising? Also, should ISPs punish users who use ad blocking proxies?

3 Discussion Points

Do we need stronger mechanisms for preventing malicious modifications? In this vein, we discussed a bit about why not to use HTTPS. The rendering time is smaller and as we discussed in the CONS, the start-up cost may be absorbed over many connections. If the content providers really care about integrity, then HTTPS is available and standardized. Is there a middle-ground where providers care a enough to provide some mild protection but don’t care enough to invest in HTTPS? We think that only time and the market will really tell. Certainly there has been a steady increase in the number of services that use HTTPS full time.

We also discussed whether there could be an extension to the SSL protocol that would not actually encrypt but simply do MAC. This might offer a middle ground for performance, though start-up costs are still there to key the MAC code.

Could this service be used to detection and monitor modifications made by a censoring state (like China?)

From a security standpoint this paper is interesting because it doesn’t have an explicit threat model. The threat model comes from the measurement. This may mean that there are other threats that they didn’t measure (targeting or more advanced attacks perhaps). This may make the approach have less security appeal.

Should ISPs be required to notify you when they do this? ISP are going to
slap this into their EULA that no one will read. Users can push back if they are paying for it. However as ISPs reduce their rates due to it, users are going to have a hard time pushing back because they are getting a free lunch.

Who’s responsibility is it to deal with this or have control over it? publishers, users, ISPs, anyone? This is another Internet gray area. In the end ISPs (within reason) still command much of the control because most users are oblivious. For example, most people have no idea what java script is doing. I would even include myself in that (most of the time).

Would you use this on your website? None of us are running a large site with much content that anyone cares about, but many said they would because it is cool.

Finally, everyone voted weak accept but me. I voted strong accept.