THE 2010 HOEFER PRIZES
FOR EXCELLENCE IN UNDERGRADUATE WRITING

In Recognition of Writing Achievement in the Undergraduate Field of Study

STANFORD UNIVERSITY
MAY 19, 2010
Getting Out of Band: Internet Explorer and the XML Data Island Vulnerability

ROBERT HOLLEY

Computer Science 181

Computers, Ethics, and Public Policy

MAGGIE JOHNSON
COMPUTER SCIENCE
Getting Out Of Band:
Internet Explorer and the XML Data Island Vulnerability

The second Tuesday of each month is a day of reckoning in most IT departments. Schedules are cleared and espresso is brewed as technology shepherds prepare their flocks for the arrival of a great danger. Though the atmosphere suggests otherwise, this event is nothing more than Microsoft’s monthly release of software updates for its products. To be sure, the widespread loathing of “Patch Tuesday” is far from baseless. Change is disruptive to the software ecosystem, and even well-intended fixes frequently have unintended consequences. As such, the arrival of software updates forces a grueling testing process upon system administrators, since even the smallest patches regularly cause previously functioning systems to fail mysteriously. This is, in fact, precisely the motivation for the Tuesday ritual. In years past, Microsoft would send out automatic updates as soon as they were ready in an attempt to get fixes, especially security-related ones, in the hands of users as quickly as possible. Unfortunately, this turned the task of managing large fleets of Microsoft-equipped machines into an unpredictable nightmare. Disaster could strike at any moment, and administrators were always putting out fires that seemed to erupt at the most inopportune times. Thus, a bargain was struck: updates would arrive once per month at a regular and convenient time, even though it meant prolonging the tenure of broken code (Lemos, 2003). Microsoft made the shrewd decision to keep high-volume customers happy; as a result, important security fixes are routinely withheld for up to five weeks.

Microsoft’s strict adherence to this approach makes it all the more interesting to examine the occasional patch that appears on a non-Tuesday. These rare “out of band”
patches indicate problems so dire that the software giant would rather deliver a nasty surprise to its entire customer base than allow the issue to persist.

**A Treacherous Island**

On December 10, 2008, the day after Patch Tuesday, a Chinese security research team called “knownsec” published details on a critical flaw in Internet Explorer, mistakenly believing that the flaw had been fixed in the previous day’s updates. While this was the first public disclosure, the vulnerability had surfaced several weeks earlier in the Chinese underground. The specifics were traded around for various sums of money (up to fifteen thousand US dollars) before eventually landing in the hands of a group seeking to steal online gaming credentials (Krebs, 2008).

These hackers then developed and launched their attack. When leveraged correctly, the flaw allowed an intruder to gain complete control over a user’s computer, so long as the user visited a website that was designed to take advantage of this security hole (Microsoft Corp, 2008). Ordinarily, the threat would be mitigated by the fact that regular users do not generally browse the web sites of Chinese hackers. However, the browser attack was launched simultaneously with one targeting popular websites. By exploiting a common database programming error, the hackers managed to subtly alter thousands of popular websites, such as abit.com, causing them to appear the same but priming them to deliver a malicious payload to vulnerable visitors (Ullrich, 2008).

The payload exploited an antediluvian feature in Internet Explorer known as “XML Data Islands.” In the early years of the web, there was a constant struggle between Microsoft and various standards bodies for control over emerging technologies. While
organizations like the World Wide Web Consortium (W3C) sought to forge an industry consensus on new protocols, Microsoft routinely jumped the gun and published its own standards unilaterally, hoping to leverage its influence and the first-to-market advantage to spark adoption of its technology. Such was the case when XML, currently the gold standard for data exchange, was first gaining acceptance. While the W3C was still hashing out the details of the “XML DOM” standard, Microsoft published MSXML, which included a special way of embedding XML data in HTML documents called “XML Data Islands” (Microsoft Corp, n.d.). The W3C’s standard eventually emerged victorious, and proprietary MSXML features like data islands fell into disuse (W3Schools, n.d.)\(^1\). Nevertheless, Microsoft left the code in future versions of its browser to appease those customers who had jumped on the wrong bandwagon (Microsoft Corp, n.d.). Enabled but forgotten, the code languished in the heart of Internet Explorer for over nine years until it was rediscovered and exploited by hackers.

According to Microsoft, the attack was an example of two common hacking techniques known as “heap-spraying” and “use-after-free.” Programs store data in a vast region of computer memory known as a heap. When the program needs to use a chunk of memory for storage, it uses various bookkeeping subroutines to mark that chunk as reserved so that it is not used to store something else. When the contents are no longer needed, the program informs the bookkeeping system that the chunk has become “free” and may be used for other purposes. The XML data island attack exploits this paradigm in the following way (Microsoft Corp, 2008):

\(^1\) The term “Data Islands” arises from the fact that Microsoft’s approach involved cramming chunks of data into HTML documents, making them virtual islands of data in a sea of markup. The W3C opposed this, arguing that documents should have a single purpose and that XML data should reside in separate XML files.
1. The attacker manipulates the program to allocate a chunk of memory from the heap and fill it with benign code, marking it as safe to execute.

2. The attacker triggers a miscommunication (the bug) whereby the bookkeeping routines think that the chunk has been freed while the rest of the program does not.

3. The attacker repeatedly requests chunks of unused memory, filling each one with the same sequence of malicious instructions. Eventually, the disputed chunk surfaces, and is filled with the attacker’s payload.

4. Finally, the attacker triggers the execution of the original chunk. Since the program believes that the chunk still contains benign code, it happily obliges, turning over control to the foreign instructions.

Though textbook, the technique is deadly when executed correctly. News of the vulnerability set off waves of alarm in the security world, with one firm director warning his customers: “The [XML Data Island vulnerability] is really nasty. No patch. Mitigation options are not good; some are draconian. Dig in folks, this could be a rough ride” (iDefense, 2008). The danger and consternation was apparently enough to compel Microsoft to break with its monthly patch schedule: a fix was developed and deployed to all customers the next week.

**Laying Blame**

From a naive perspective, the blame for any computer attack lies with the attackers. Certainly they are the only ones in the situation with malicious intent, and the mere opportunity to commit a crime does not excuse carrying it out. That answer is, however, unsatisfying. The Internet connects billions of people from all corners of the globe, and represents them to each other as numbers. This has three important consequences. First, the anonymous and borderless nature of the Internet makes it notoriously difficult to track down cyber-criminals, so the threat of punishment is not a viable deterrent. Second, the Internet provides each of its participants with direct access
to everyone else, so a lone unscrupulous individual with the appropriate expertise is a threat to everyone everywhere (that is to say, the Internet is a weapon of mass disruption). Finally, the faceless nature of the victims enables those too squeamish for traditional robbery to virtually break and enter from the comfort of their homes. As such, it is more productive to view cyber-attacks as akin to natural disasters. It would be convenient to blame the hurricane, but assigning responsibility to the builders of flimsy levees is a more effective approach to preventing future harm. Adopting this standpoint, the lens of culpability sweeps across the Pacific from China to Redmond.

“Blaming the engineer” is a well-known scapegoating technique in technical industries. Most problems could, in retrospect, have been solved by more careful, more ingenious, more prescient engineering, so it is tempting to blame any flaw in a system on those who built it. Unfortunately, this myopic perspective glosses over deeper managerial missteps that spawn technical errors and amplify their effects. To be sure, some poor programmer in 1998 made a small mistake when he or she wrote the heap-allocation routines for the XML Data Island feature, but all software has bugs. The more interesting questions are why the bug was never caught during development, and why it remained in the code for almost a decade. Corporate privacy makes this question impossible to answer with certainty. Nevertheless, the interesting history of the feature engenders speculation that Microsoft’s business strategy was the true culprit. In its effort to conquer the market by blitzkrieg, Microsoft likely rushed the initial development of MSXML to maximize its advantage over the W3C. In times of hurried development, careful review can fall by the wayside, rationalized away by the notion that the software can always be reviewed and updated later. When data islands failed to gain significant adoption, the feature remained,
but Microsoft almost certainly did not allocate any resources towards maintaining and improving it. Thus, unpolished code from the 90’s most likely lay dormant as the rest of the browser modernized, a ticking time bomb waiting to be discovered by evildoers.

**Assessing the Damage**

For all the fuss, the initial impact of the vulnerability appears to have been relatively small. Tens of thousands of credentials to Chinese online gaming sites were stolen, undoubtedly causing a fair amount of disruption and monetary loss. But there are no reports from that time by major security researchers of the vulnerability being exploited to steal financial information, compromise trade secrets, or form remote-control armies for future attacks. Granted, those who remained un-patched likely fell victim to later copycat exploits, but the data island vulnerability is hardly unique in that regard: an un-patched system is vulnerability to every documented security issue fixed since the initial release. From a security and public safety standpoint, the damage was smaller than anyone at the time would have dreamed.

The vulnerability had another interesting, though indirect, effect. Beginning on December 16th, news of the vulnerability trickled out of special-interest media and into the mainstream press. Reports of the danger surfaced on sources like *The Washington Post* (Krebs, 2008) and *BBC News*, with headlines such as "Serious security flaw found in IE - Users of Microsoft's Internet Explorer are being urged by experts to switch to a rival until a serious security flaw has been fixed" (BBC, 2008). This time in the limelight had a profound effect. The Mozilla Metrics Team, which tracks download statistics for Firefox (a rival browser), reports a gigantic spike in downloads of Firefox by Internet
Explorer users (suggesting the intent to switch) coinciding exactly with the moment the news went mainstream (Fig 1).

Fig 1 – Firefox Download Data

Thus, it seems plausible that Microsoft itself emerged from the debacle as the biggest loser. This certainly has an element of poetic justice – the software giant paid for its aggressive market share grab by losing millions of customers ten years later. Nevertheless, it is worth noting that this outcome is the exception rather than the rule. The 2003 MSBlaster worm caused up to ten billion dollars in damage (Garza, n.d.), and the most recent Internet Explorer hole facilitated a massive attack on Google China that prompted the company to pull out of the country completely (Drummond, 2010). Like hurricanes, security vulnerabilities sometimes fail to deliver the storm that was predicted. Even so, one should not underestimate their potential for destruction.
Works Cited

http://news.bbc.co.uk/2/hi/technology/7784908.stm


Garza, George. Top 10 worst computer viruses. Retrieved from

iDefense. (2008). Exploitation for Unpatched Internet Explorer 7 Vulnerability in the
Wild. Retrieved from

Washington Post. Retrieved from
http://voices.washingtonpost.com/securityfix/2008/12/exploit_for_unpatched_internet.html


