Anatomy of an exploit - six, in fact - as Google reveals details of Pwnium hack against Chrome

Blog Post from Paul Ducklin, Sophos

Google'sChromiumblog - which offers "news and developments from the open source browser project" - tells a fascinating tale of how the Chrome browser was penetrated and pwned atCanSecWest2012.Breaking into a browser isn't as easy as it was a few years ago.
Back then, many browsers ran as a single monolithic process which could be taken over by finding and exploiting a single vulnerability.
That's unusual these days, so this is an hacking story of considerable derring-do.
For those of you not up to speed on modern hackerspeak, the verbto pwnis a needlessly deliberate misspelling ofto own. Pwning a system means penetrating it with an exploit or hack to the point that you acquire sufficient privilege to do pretty much what you want. It's pronouncedpone, and rhymes withhone,drone,cloneand, of course,own.
In 2012, Google withdrew from the regularpwn2owncompetition operated by CanSecWest as part of its conference. Instead, it ran its own contest,pwnium, with generous prizes and different rules.

Google chose to spurn pwn2own because it was inconsistent with Google's ownresponsible disclosure doctrine- pwn2own didn't require entrants, or even winners, to reveal information about the exploits they used to the vendors whose code they penetrated.
In the end, there were two pwnium winners, and the Chromium blog's current article is the story of the first winner, Pinky Pie.
(She gets her name from aHasbro toy. A pony. Geddit?)
I won't go into too many details here - read theChromium blog articlefor those - but the level of determination shown in this attack is worth remembering.
Pinkie Pie exploited six independent bugs in a carefully-orchestrated sequence, each building on the pwnership offered by the last:
1.Use a bug in Chrome's pre-renderer process, which boosts HTML rendering speed by pre-processing web content, toforce a Native Code module to load. This escapes the limitations of the pre-rendering process.
2.Use the Native Code module totrigger a memory overflowin Chrome's graphics-handling process.
3.Use the memory overflow toinject executable codeinto the graphics-handling process. This escapes the limitations of the Native Code module.
4.Use a timing bug to make the shellcoderun with higher privilegesborrowed from the HTML renderer. This escapes the limitations of the graphics handler.
5.Use aprivilege protection bugto boost the HTML renderer's privilege to allow it to load extensions. This escapes some of the limitations of the HTML renderer.
6.Use anotherprivilege protection bugto allow an extension to be loaded using full browser privileges. This escapes further limitations of the HTML renderer.
7.Use anotification bypass bugto install and load a Native Code module silently. This avoids the "you probably don't want to do this" dialog which the user is supposed to see.
Installing and running native code with the privilege of the browser itself, simply by visiting a web page, and without any warning, gives you adrive-by install.
That's undeniable pwnership - you can now do everything the current user could do, and anything he wouldn't.
In two words, "Game over".
(There are seven steps above, but six bugs. I divided the memory overflow bug into two steps: triggering the overflow, and actually controlling the overflow to turn the bug into an actively-exploitable vulnerability. Finding a memory overflow is one thing. Reliably exploiting it is quite another.)
In summary, whether you are writing your own code or deploying other people's: expect the best, but prepare for the worst.
Even powerful and much-vaunted technologies such as Address Space Layout Randomisation (ASLR), Data Execution Prevention (DEP), sandboxing and privilege separation can be bypassed, albeit one small step at a time. And hackers are prepared to take the time to devise and combine those small steps to achieve a grand-final-winning result.
After all, there's good money in finding exploits these days. Pinkie Pie took home a cool US$60,000 for her work.

PS.Watch this space. The Chromium bloggers have yet to write up the details of the second pwnium winner. He chained togethertenbugs, not merely six.