Memory Corruption...

(And Why We Dislike That Term)

Carsten Eiram
Chief Research Officer

About Risk Based Security

Mission

To equip clients with the technology and customized risk-based consulting solutions to turn security data into information and information into a competitive advantage.

Background

Risk Based Security, Inc., incorporated in 2011, was established to better support the users/contributors to the Open Security Foundation, OSF, with the technology to turn security data into a competitive advantage.

The OSF's wealth of historical data, combined with the interactive dashboards and analytics offered by Risk Based Security provide a first of its kind risk identification and security management tool.

Risk Based Security further complements the data analytics with risk-focused consulting services to address industry specific information security and compliance challenges.

Discriminators

Risk Based Security offers a full set of analytics and user-friendly dashboards designed specifically to identify security risks by industry.

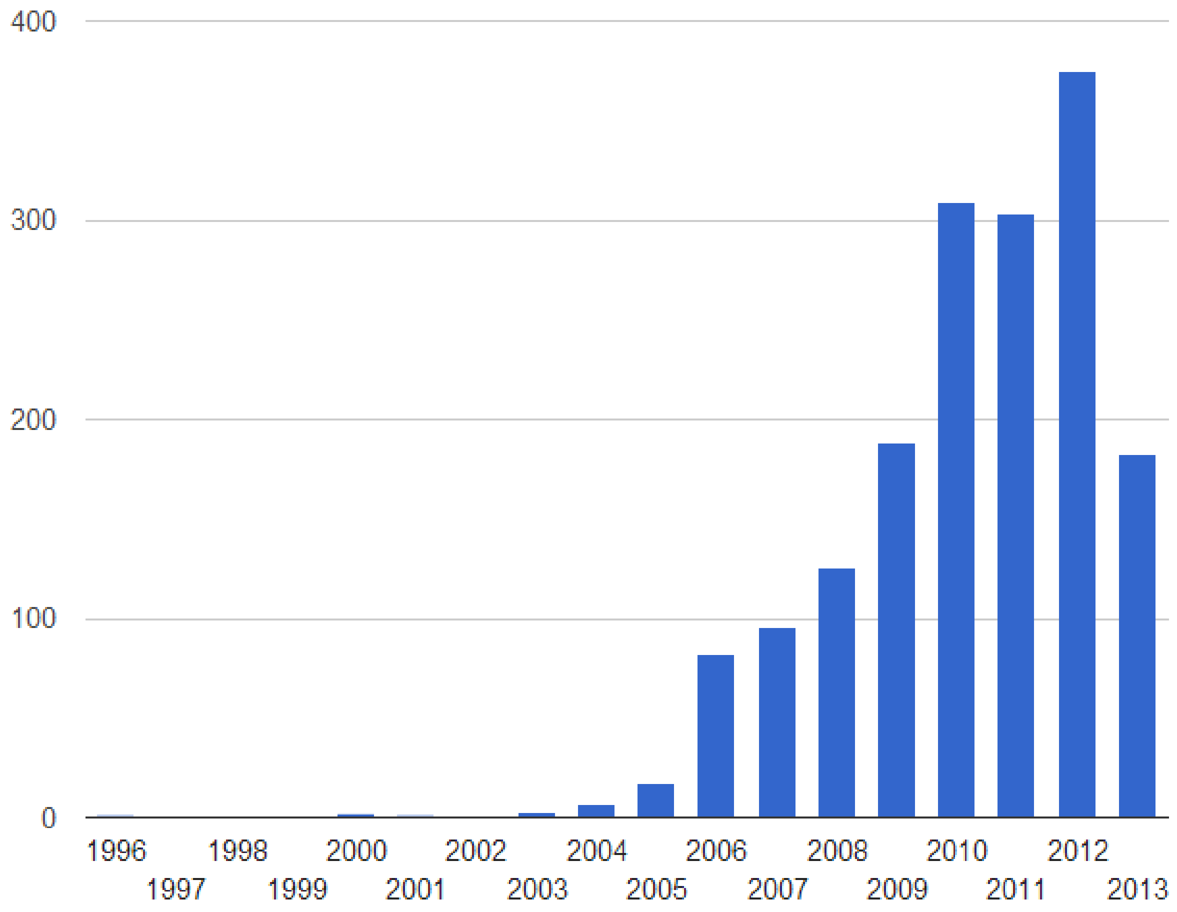
Risk Based Security is the only company that offers its clients a fully integrated solution – real time information, analytical tools and purpose-based consulting.

Unlike other security information providers, Risk Based Security offers companies comprehensive insight into data security threats and vulnerabilities most relevant to their industry.

"Memory corruption"...

In recent years, we've seen more and more security advisories released from vendors and researchers (both new and accomplished) misusing the term, “memory corruption”. While the term is becoming more used, we VDB people have a strong dislike for it, as it just vaguely says: "Something messed up memory”. It provides no real details about the root cause, and from an exploitation standpoint it’s often assumed to possibly, potentially, perhaps, maybe, theoretically allow code execution.

To illustrate the adoption of the term, I pulled some quick stats from the OSVDB[1] database for entries titled: "Memory Corruption". The graph below shows the number of entries per year from 1996 to August 1st 2013.



[1] http://osvdb.org/

As illustrated, a significant shift in its use occurred around 2006 and then another in 2010. While the total number of entries with "Memory Corruption" in the title as of August 15th 2013, 1706 (~1.8%), is by no means staggering, it's should be noted that we've in recent years seen an uptake in the use of this term to around 4% of all reported issues.

February 15th 1996 is currently the first time OSVDB records[2] the term being used. This was in the AlphaX 6.2a2 changelog[3] ("*Memory corruption problem fixed.*").

July 4th 2004 appears to be the first time OSVDB recorded[4] a researcher misusing the term. The information was initially posted on securiteam.com[5], but on July 23rd 2004 also to the full-disclosure mailing list[6] and the company's[7] own website.

The researcher seemingly didn't understand the root cause of a crash, which ultimately was a heap-based buffer overflow due to an error when calculating the size of a comment. This size was used by memcpy() when copying the data, as later explained[8] by another researcher. The vulnerability was eventually fixed by Microsoft in MS04-038[9] and later exploited[10] (not confirmed).

One major software vendor that is a fan of the term and uses it frequently to tell us absolutely nothing about a vulnerability is Adobe[11], though they at least seem to try to use more appropriate terms for some issues. Microsoft is another culprit - did anyone read MS13-055[12] and find it informative or useful?

The Sad Case Of Microsoft's Security Bulletins Improving, But Then Devolving

Microsoft is an especially saddening case: For a while they were actually doing a good job by providing a fair amount of details in their IE (Internet Explorer) security bulletins. Back in 2011, Microsoft was heavily using the "memory corruption" term and their descriptions would state (using MS11-050[13] as example): "*A remote code execution vulnerability exists in the way that Internet Explorer accesses an object that has not been correctly initialized or has been deleted*",

[2] http://osvdb.org/show/osvdb/90021
[3] http://alphatcl.cvs.sourceforge.net/viewvc/alphatcl/Help/Changes%20-%20Alpha?view=markup
[4] http://osvdb.org/show/osvdb/7607
[5] http://www.securiteam.com/windowsntfocus/5XP051FDFM.html
[6] http://archives.neohapsis.com/archives/fulldisclosure/2004-07/0960.html
[7] http://web.archive.org/web/20040724054719/http://www.ecqurity.com/adv/IEstyle.html
[8] http://archives.neohapsis.com/archives/fulldisclosure/2004-07/1125.html
[9] http://technet.microsoft.com/en-us/security/bulletin/MS04-038
[10] http://www.securiteam.com/exploits/5NP042KF5A.html
[11] http://www.adobe.com/support/security/bulletins/apsb13-14.html
[12] http://technet.microsoft.com/en-us/security/bulletin/ms13-055
[13] http://technet.microsoft.com/en-us/security/bulletin/ms11-050

indicating that it was either an issue related to use of uninitialized memory or a use-after-free.

Towards the end of 2011 (e.g. MS11-081[14]) and in 2012 (e.g. MS12-010[15]), Microsoft went away from the “memory corruption” term and starting just using “remote code execution”, which is also vague w.r.t. root cause, but their descriptions improved greatly by stating: “*A remote code execution vulnerability exists in the way that Internet Explorer accesses an object that has been deleted.*”, now making it clear (at least to researchers and VDBs) that it was actually a use-after-free. Similarly, the titles would give at least a hint to what functionality the issue was related to e.g. “*SelectAll Remote Code Execution Vulnerability - CVE-2012-0171*” from MS12-023[16]. This allowed VDBs to provide a somewhat unique title and description for the issue and also made it easier later on to match additional details to the right entry and CVE.

Towards the end of 2012, it became even better: Titles included some details and even clearly stated the root cause as “use-after-free” e.g. “*cloneNode Use After Free Vulnerability - CVE-2012-2557*” from MS12-063[17]. Kudos to whoever at Microsoft made this decision to uniquely title the issues and specifying the root cause.

Beginning of 2013, Microsoft changed the format again with MS13-009[18]. Instead of listing separate entries, they were all merged into a single entry, but that didn’t matter, as each vulnerability still received a unique, descriptive title and clearly mentioned the root cause.

Then from the April release (MS13-028[19]) in 2013, things started to devolve fast. Instead of unique titles, each issue would just be titled: “*Internet Explorer Use After Free Vulnerability*”, but at least still specify the root cause. However, that changed with the June release (MS13-047[20]) in 2013, which just titled almost every single issue as: “*Internet Explorer Memory Corruption Vulnerability*” and described the issues as: “*Remote code execution vulnerabilities exist when Internet Explorer improperly accesses an object in memory*”. Very disappointing...

As Microsoft is sometimes used as a reference point, we'd hate to see other vendors adopt and similarly misuse the "memory corruption" term and provide such vague descriptions. We also hope that Microsoft recognizes this useless manner in which to list vulnerabilities in their security bulletins and go back to the late 2012 policy.

Limiting information in this manner does not make things harder for people wanting to exploit this, as binary diffing the patches quickly allows them to obtain the necessary details within a few

[14] http://technet.microsoft.com/en-us/security/bulletin/ms11-081
[15] http://technet.microsoft.com/en-us/security/bulletin/ms12-010
[16] http://technet.microsoft.com/en-us/security/bulletin/ms12-023
[17] http://technet.microsoft.com/en-us/security/bulletin/ms12-063
[18] http://technet.microsoft.com/en-us/security/bulletin/ms13-009
[19] http://technet.microsoft.com/en-us/security/bulletin/ms13-028
[20] http://technet.microsoft.com/en-us/security/bulletin/ms13-047

hours to a couple of days of the release; only customers and VDBs are negatively impacted by this lack of detail.

Reasons For Misusing “Memory Corruption”

This leads to the point that some vendors are fond of using this term as some sort of catch-all for everything from buffer overflows to use-after-frees, even when more appropriate terms exist. Why? Three immediate hypotheses:

1) It’s a way for them to divulge absolutely zero information to their customers and the public. Many vendors are still “afraid” of disclosing too much information about vulnerability fixes.

2) The vendor just can’t be bothered spending the time and resources to provide proper advisories, instead copping out and just flagging everything possible as “memory corruption”.

3) The vendor doesn’t really understand the root cause, but confirmed that something went wrong and corrupted memory, and that it was somehow addressed. This should be a major cause for concern; if the vendor didn’t properly understand the root cause, then what are the odds it was properly fixed? Having done binary diffing of patches from various vendors, I’ve unfortunately seen quite a few examples of this.

The same, unfortunately, goes for some researchers. Cases where an application crashes, but the researcher doesn't really understand the crash or root cause or just can’t be bothered figuring it out after a fuzzing run, just get tagged with "memory corruption" regardless of that even being the case or not. Instead of guesswork, it would be refreshing to see researchers just make it clear to what extent they actually researched a vulnerability and admit, if they haven't researched the root cause or just don't understand it.

Throwing menhir in Obelix style, glass houses, and all that then let me be the first to admit to also being guilty. Though I have absolute disdain for the term, I have a few times misused it to not divulge details, but also in some cases where it was just the most fitting when only providing a very short paragraph as description of a vulnerability.

Yes, "memory corruption" does have its merit in some cases, but both researchers and vendors should stop misusing this very generic term and instead strive to use more appropriate terms, provide more accurate descriptions of the root cause, add a proper CWE, or alternatively just admit that they don't know what the problem is. That will make our lives as VDB people so much easier!