Nine years of bugs & coordinated vulnerability disclosure: Trends, observations & recommendations for the future

Prepared by:
Matt Lewis, Research Director

Table of contents

1. Background

As part of our vulnerability research work at NCC Group we find many vulnerabilities (bugs) in commercial products and systems. In addition, our consultants engage in vulnerability research in their own time when not working on a client project, looking at the security of products or systems of interest to them. In March 2009, in order to keep track of vulnerabilities that consultants were finding in commercial products and systems, we set up an internal system to log these details and to assist with the workflow of responsible disclosure with affected vendors. In June 2016 we then moved to a new vulnerability tracker based on Bugzilla [1] and in April 2018 we managed to merge the old and new systems to provide us with data on nine years of vulnerabilities found across all manner of commercial products and systems. This paper provides some analysis of the data that we've captured over the past nine years in terms of types of bug found, their risk ratings, whether there are any trends in specific vulnerability classes and whether there are any observations around the overall responsible disclosure process.

To clarify, the vulnerability data that we're presenting here is not that captured from our daily penetration testing engagements across our client base – the data we present here encompasses 1108 logged vulnerabilities that were found by one of:

- Consultants researching commercial products in their own time.
- Consultants finding vulnerabilities in commercial products as part of a client engagement (the commercial product itself was not typically the focus of the scope, but a vulnerability was identified as part of testing and so logged in our tracking system).
- Consultants finding vulnerabilities from internal research projects looking at commercial products and systems.
- Product Attack Challenges (PACs), where several times a year we run a short time-bound crowd-sourced internal bug hunt against a commercial product or system. We do this for general training as well as for fun, but the outcome might be revelation of a number of bugs in the target.

"As part of our vulnerability research work at NCC Group we find many vulnerabilities (bugs) in commercial products and systems."

The 1108 vulnerabilities relate to a number of different technologies (across 354 unique vendors), including operating systems (both closed and open source), commercial off-the-shelf (COTS) applications, hardware and networking appliances (both domestic and enterprise, including IoT devices, routers and switches) and enterprise cloud or web services.

Now that we have formally established our ever-growing centralised vulnerability reporting system, our intention is to annually report on our observations. Over the coming years, and certainly in another 9 years' time, it will be interesting to revisit this paper to see what has changed (if anything). In its current form this data relates to vulnerabilities broadly discovered by the UK and European offices of NCC Group, however we are in the process of merging and centralising all of our vulnerability reporting systems across the group to gain even more insight into possible trends in this space.

2. Some caveats

While we aim to pull out common themes and observations, there are some caveats to the data worth mentioning so that we're not at risk of over-generalising or over-simplifying the data and any patterns therein. Key things to note include:

- We can't draw too many conclusions around the types and numbers of bugs identified over the years. Any drops in the number of bugs reported could just be due to our consultants being particularly busy, while increases could be due to NCC Group running PACs, potentially producing spikes due to a particularly buggy PAC target.

- Inconsistency of bug type and labelling. We logged things differently between our old and new reporting system so bugs were not necessarily tagged with exactly the same title. For example an issue may have sometimes been logged as just a cross-site scripting (XSS) issue, while other times it was specifically called out whether it was stored or reflected. Additionally, there are entries in the data that were marked as 'Multiple Vulnerabilities'. This means that consultants who found many different bugs or classes of bug in one product or system may not have differentiated them as unique bugs and instead just created one, overarching entry pertaining to all bugs of that class found in that target.

- There is likely inconsistency depending on consultant perspective on the end-result of exploitation. For example an 'Arbitrary File Upload' vulnerability might actually be used to upload a web shell which results in remote code execution (RCE) on the target. While the vulnerability that led to RCE was due to the initial file upload vulnerability, some consultants may have logged their vulnerability entry as RCE, since that was the end-result.

3. Risk ratings

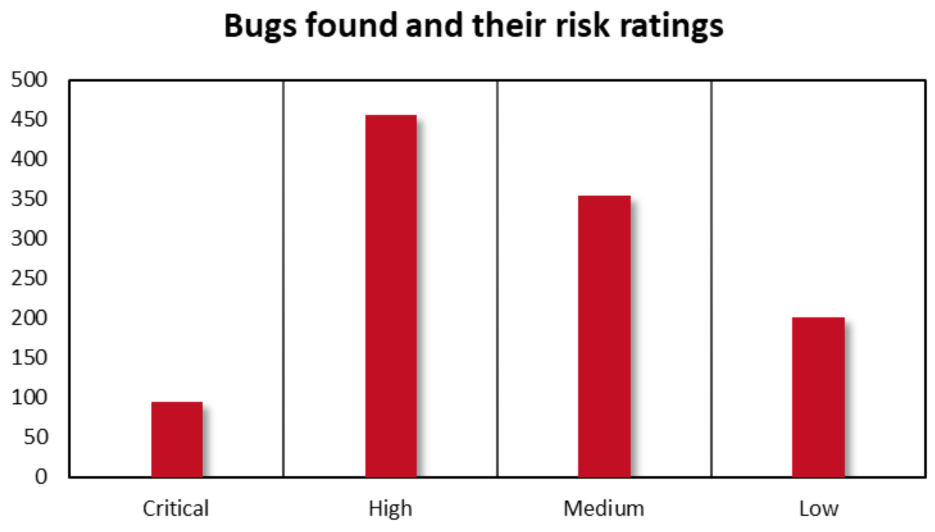

Figure 1

Looking at the risk ratings of the 1108 logged bugs (figure 1) we see that most were high (41%) or medium (32%) in terms of assessed risk rating. The actual breakdown was:

- Critical: 94
- High: 457
- Medium: 355
- Low: 202

To be transparent in our ratings, the risk classifications logged by consultants may be subjective and while they will broadly follow industry standard judgements, NCC Group have not provided specific guidelines for the classification of bugs for consultants to reference. To generalise the classification of our bugs is to say that the higher the threat and impact was, the higher the logged severity was.

As an example, if a vulnerability was found in a system that could be exploited easily by anyone on the internet and that exploitation gave full access to the underlying server and all its data, then we would say the threat was very likely and the impact very high due to data breach, therefore this would be critical. However, a really nuanced vulnerability that perhaps was very hard to exploit and if exploited only resulted in partial control of an underlying system, then we might say the threat here was minimal, along with the impact, so this might be an example of a medium or low risk vulnerability.

4. Bug types

Over the past nine years we have tagged each bug as one of 53 classes of vulnerability. To visualise these classifications we have provided the top 20 bug classes in figure 2 and the bottom 33 classes in figure 3.

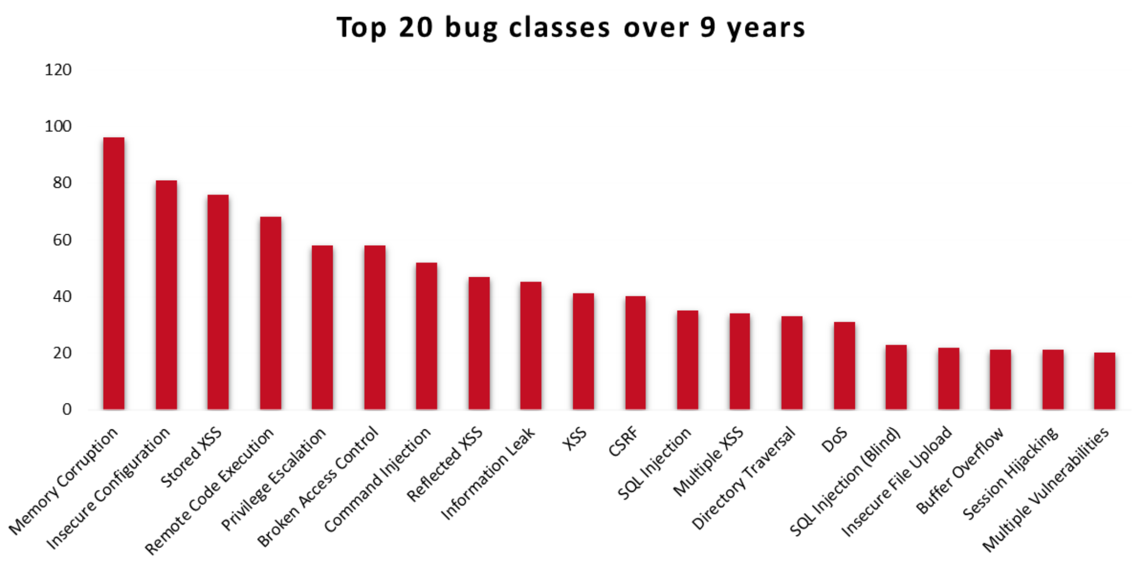

Figure 2

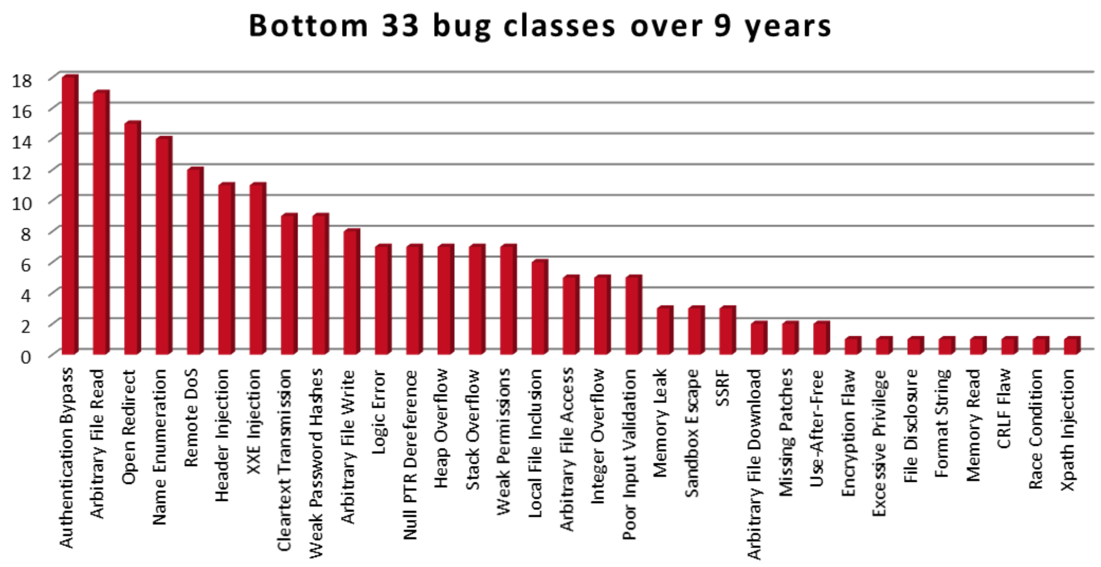

Figure 3

XSS was the most common class of bug over the past nine years. In our data, XSS was logged over the years as either stored XSS, reflected XSS, multiple XSS (where one issue is logged but many instances of XSS are found) and just simply XSS (where there was no differentiation of XSS mode). The total number of XSS entries logged was 198, which represents 18% of all noted bugs.

While the combined XSS flaw categories render XSS the most common bug class found, we also see that memory corruption flaws were common (96 in total, or 9% of the total bugs found). When looking in more detail at the different types of memory-related flaws identified we can see a noticeable drop in frequency over time, as shown in figure 4.

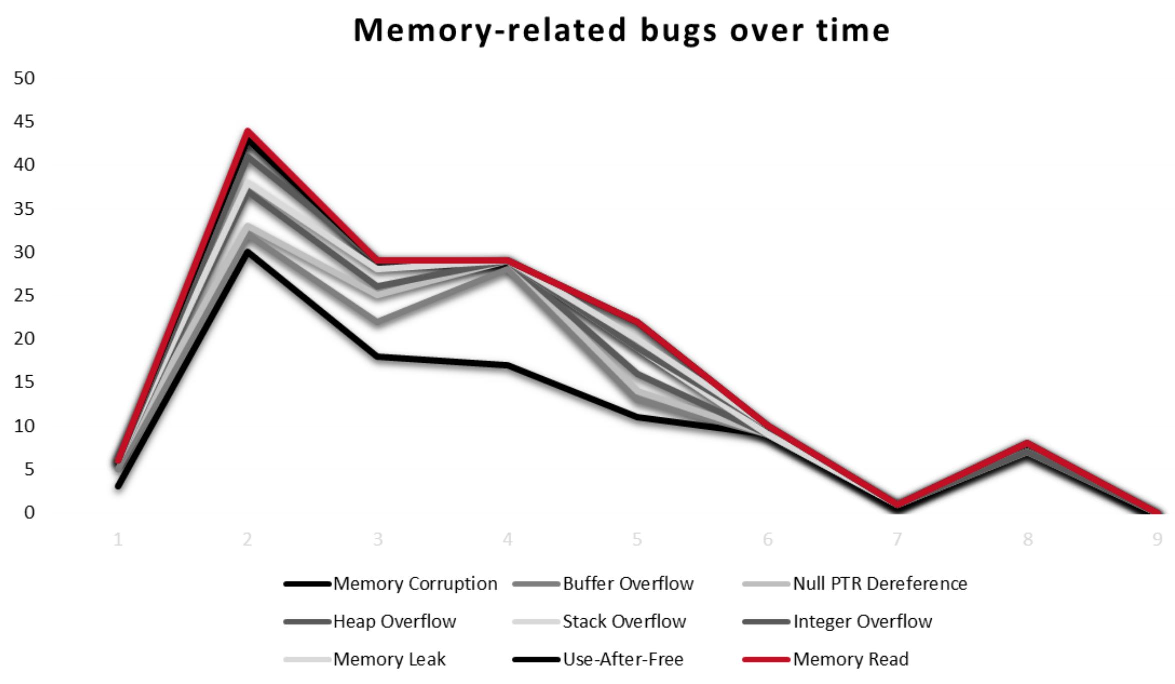

Figure 4

This drop over time could be explained by a number of different reasons, including:

- **Consultants focusing more on web-based bugs over time.** This implies that memory flaws may not be reducing but rather bug hunt attention is being placed elsewhere.

- **Improvements in operating system and compiler defences.** These improvements could then reduce (or at least render much harder) the exploitation of memory corruption flaws such as DEP, NX, ASLR.

- **A likely reduction in thick clients and compiled code.** Most modern applications tend to be web-based and/or leverage feature-rich HTML and JavaScript for client data entry. Therefore, there are likely fewer thick client targets compared to nine years ago that might otherwise be available for fuzzing and reverse engineering to uncover memory-related flaws in their native code implementations.

- **Broadly there may be less use of native code today** as managed code is more commonplace compared to nine years ago.

5. Web bugs

Looking at the main web vulnerability classes identified over time we can see no discernible pattern, increases or decreases (Figure 5). From this we might infer that web-based flaws, such as the Open Web Application Security Project (OWASP) top ten [2], have remained fairly static over the past nine years by way of manifestation and instances found.

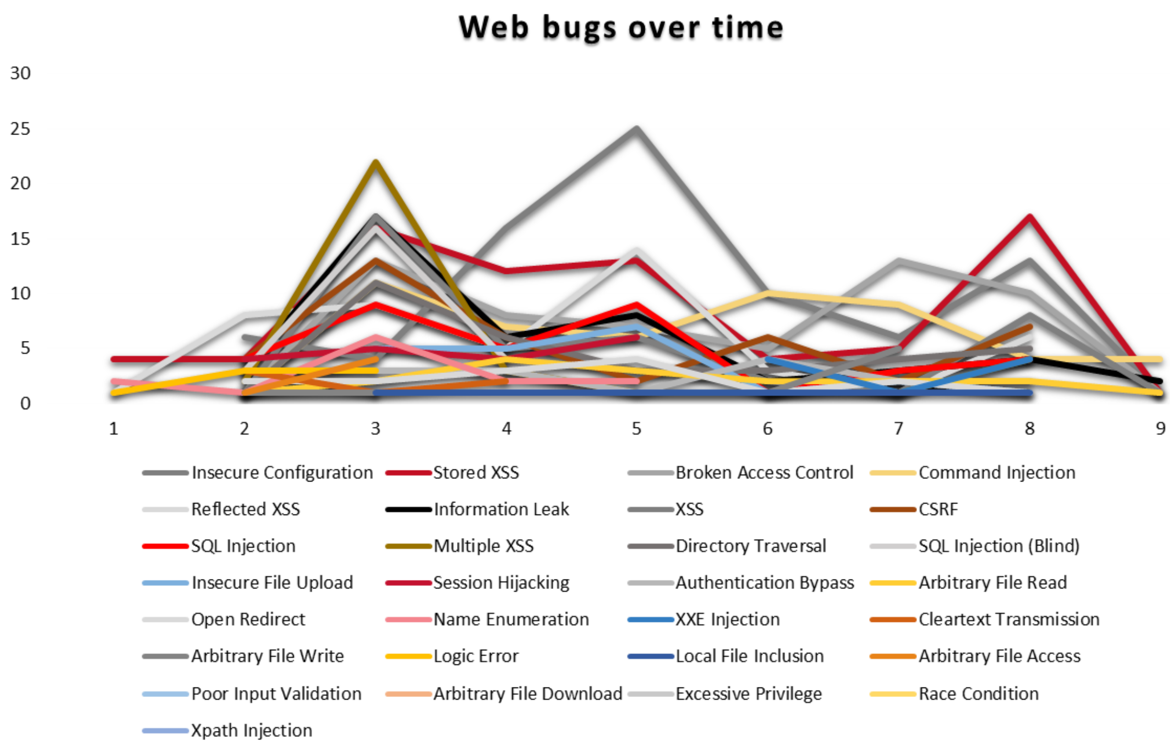

Figure 5

This lack of discernible pattern can be seen with even with the most prevalent vulnerability class (XSS) reported over the nine years as seen in figure 6; there is no obvious trend except for a noticeable bumper year for XSS in 2012 (year 3):

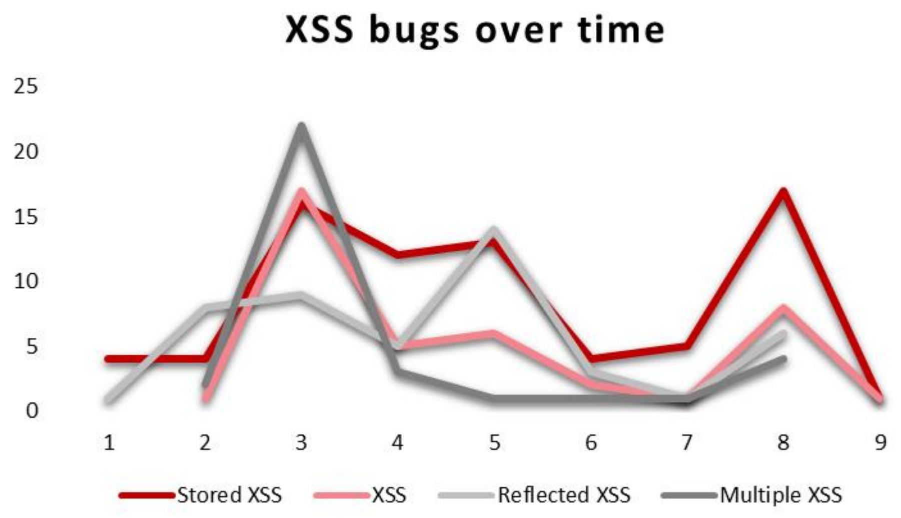

Figure 6

6. Have any bug classes been squished?

The bug classes with a minimal number of instances reported (i.e. 1) are as follows:

- Encryption flaw
- Excessive privilege
- File disclosure
- Format string
- Memory read
- Carriage Return Line Feed (CRLF) flaw
- Race condition
- XPath injection

The low number of reported instances may be due to lack of investigation around those bug classes as opposed to a confirmed reduction in those types of bugs over time. However, the inclusion of format string and memory read bugs may tally with our broader observation of a likely decrease in memory-related flaws over time.

CRLF flaws are often used as a means to perform another attack and therefore may not be explored as much compared to more easily and verifiably exploitable vulnerabilities. Similarly, race conditions can be tricky to exploit and as they rely on timing then they don't necessarily give the exploit reliability that many consultants or researchers look for. Finally, a low numbers of XPath injections may be indicative of how data formats such as JSON have seemingly become more prevalent across web technologies for data exchange and storage.

7. Status of bug entries: A whole pot of zero-days?

Within our vulnerability tracking and disclosure system we have a number of statuses that are used to track bug disclosure from initial discovery by our consultants, through to vendor notification and eventual public reporting, such as with Common Vulnerabilities and Exposures (CVE). The definitions of our statuses are as follows:

- **Unconfirmed**: New bugs submitted begin as unconfirmed, where our internal moderators will then work with the bug reporter to confirm the vulnerability and risk rating, usually through a proof of concept (PoC).
- **Approved ready to report**: Once moderators have approved bug submissions consultants can then engage with the vendor on coordinated, responsible disclosure.
- **Reported to vendor**: This status is set when we are waiting for a response from the vendor.
- **Pending publishing**: This status means that we are about to publish and are possibly waiting for some final details from the vendor or QA of a technical advisory and any accompanying public messaging or blog post.
- **Closed**: Once an issue has been responsibly disclosed (through advisory) it is marked as closed (or if the issue is dismissed for whatever reason, e.g. no PoC/vendor won't fix/doesn't think is an issue etc.)
- **Unknown**: This status is set when vulnerabilities do not get resolved, or if we don't know how far vendor communication has previously gone.

8. Rumsfeldian observations: There are known unknowns & unknown unknowns

While the listed statuses are present in our current reporting system, our older reporting system didn't have these exact statuses. Therefore, most of the issues imported from the older system that were not previously marked as closed were marked as unknown in the new system. The tag of unknown in this context broadly means that the vulnerability was likely reported to the vendor but no fix occurred. This could be more various reasons, including the vendor not responding despite multiple efforts to establish contact. Some of the responsibility for lack of clarity on status also falls on us, as some of the unknown statuses could be due to:

- A consultant leaving NCC Group before closing down the issue or handing over to someone else to continue vendor liaison.

- Consultants not chasing vendors enough (noting that we rely on consultants to do this in their time outside of client work).

At the time of writing there are currently 722 unknown issue statuses in our vulnerability reporting system, as seen in figure 7. This means that up to 65% of the bugs logged could conceivably be unresolved and thus still be zero-days.

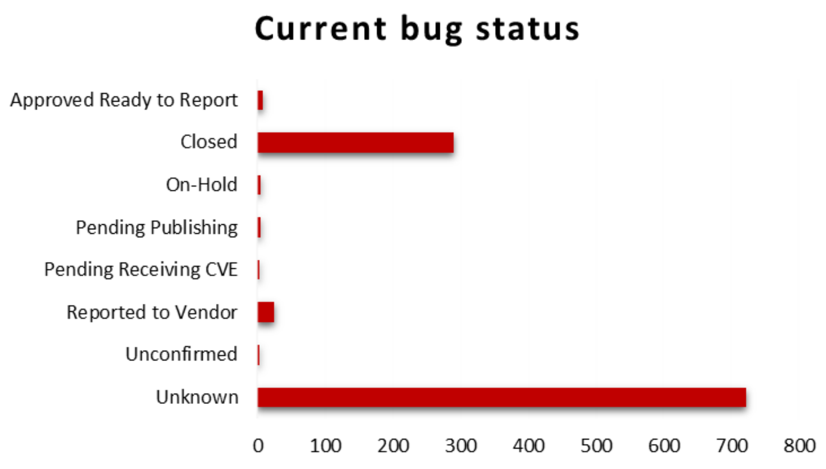

Figure 7

Digging into the data we found that the earliest logged unknown status bug was from 01/08/2010 and so is almost eight years old. It is a critical vulnerability in a file upload mechanism of a popular web CMS system resulting in arbitrary code execution on the underlying server. The unknown status and lack of CVE might be a strong indicator that the vulnerability was never patched. While it is possible that the actual security flaw has remained manifested within the code base across new version releases, it is also possible that many of the legacy unknowns are no longer an issue due to software and systems being updated or replaced over time.

9. Entering the echo chamber of disclosure…

On the subject of vendor notification our experience over the years has been mixed. Quite often we experience difficulty in gaining any response from an initial email regarding vulnerability details that we want to responsibly disclose, and this is certainly more common among less established vendors. This is a long term and ongoing problem as we have recently had difficulty in identifying suitable security contacts at an established software company to responsibly disclose critical vulnerability information we have on their product. Our efforts have included multiple search attempts through Twitter (including direct contact with their social media teams) and LinkedIn for technical personnel who we can contact with the aim of establishing secure communications for disclosure. We have even contacted the CEO directly, yet all channels have sadly been met with radio silence.

While our disclosure policy [3] has a 30 day schedule for publication after initial notification we try to accommodate those vendors who actively respond and who may need more time to work on a fix. When we are met with no response from vendors we find ourselves in a difficult position as we often don’t want to expose our affected clients (and the wider internet community) to a vulnerability unless we know that there’s a confirmed fix. This is often a difficult decision to make and can result in bugs logged in our system receiving the unknown status, which in many cases remain as zero-day vulnerabilities.

10. For the love of Great Odin's raven, give me a CVE already!

Over the years we have noticed many issues in obtaining CVEs for the vulnerabilities that we have found and reported. Of the 1108 vulnerabilities logged only 27 (or 2.4%) have resulted in a CVE. This is echoed on the advisory page of our website [4] which only holds a limited number of published advisories, with not all of those advisories having CVEs.

The difficulty in obtaining CVEs has been known to the industry for some time [5] and is likely due to limited resources at MITRE in conjunction with a large global researcher base who are finding a myriad of vulnerabilities in large volumes, each requiring CVEs on a daily basis. When we ask vendors if they will apply for a CVE during disclosure it is often our experience that they won't unless they are a CVE Numbering Authority (CNA), of which there are fewer than 100, with most of these being specific vendor and project CNAs. We are often met with vendor responses such as "we are not going to request a CVE for this", "we don't know how to request a CVE" or quite commonly (and surprisingly) "what is a CVE?"

This is in spite of the following process being documented in the ISO standard (29147) for vulnerability disclosure [6]:

"Therefore, to receive a CVE-ID number, the vendor should do one of the following:

a) Contact one of the CVE Numbering Authorities (CNAs) listed in the link below, which will then include a CVE-ID number in its initial public announcement about your new vulnerability;

b) Contact an emergency response team such as CERT/CC, DOE-CIAC, CanCERT, etc.;

c) Provide the information to a vulnerability analysis team;

d) Provide the assigned number CVE or other to the finder."

CVEs are important for a number of reasons; again borrowing from ISO 29147:

"The intention of the CVE is to be comprehensive with respect to all publicly known vulnerabilities and exposures. By citing the CVE in an advisory, users can more easily distinguish which vulnerability is the subject of the advisory."

Sadly the current process of obtaining CVEs is either unknown or misunderstood, difficult to navigate, commonly non-responsive or a combination thereof.

11. Closed issues & time to fix

In reference to the bugs that we have successfully closed (either via the issue being fixed or the risk being explicitly accepted or dismissed by the vendor) we have a total of 289, or 26% of all bugs logged. Looking across all closed issues, the average time to fix these (based on the delta between logged dates of initial reporting and the date of advisory publication) was 60 days. Another caveat must be placed here, as we may not have been religious in our process of updating our reporting system with advisory publication dates. This may result in our analysis in this area being slightly exaggerated, but when compared as averages it does provide some approximate indication of time to fix.

Considering every closed issue across all risk ratings, the longest time to fix was 244 days, while the shortest was an impressive one day (this particular example was a stored XSS in a messaging gateway).

Drilling down to those closed issues that were critical in risk rating, the longest time to fix was 235 days, while again, the shortest was an impressive one day. The average times to fix across different risk-rated closed issues were:

- Critical issues: 74 days
- High risk issues: 34 days
- Medium risk issues: 77 days
- Low risk issues: 96 days

From these averages we can see that they all fall outside of our 30 day disclosure period (it is also worth noting that there is no suggested period in ISO 29147). An average 74 days to fix critical risk issues is quite high, while 96 days (a longer average time to fix) for low risk issues might be expected given that lower risk might translate to lower priority on the vendor’s side.

Overall, our observations around disclosure and fix timelines have not been flattering from the perspective or pragmatism, let alone urgency.

12. What of the next nine years?

Looking at the types of vulnerability that our consultants are finding in recent times, we are seeing an increase in the prevalence of:

- Deserialisation flaws [7]
- Server-side request forgery (SSRF) [8]
- Chaining of bugs: Multiple low risk issues exploited in a chain across a large or complex web application or infrastructure which results in full unauthorised control
- Misconfiguration
- Hardware security: As we engage more on embedded systems and IoT we are seeing more hardware-related design and implementation flaws

Chris Anley, Chief Scientist at NCC Group, recently presented on the topic of chaining of bugs and misconfiguration at CyberUK [9], where he spoke to the following ten core, commonly chained issues, specifically used in web attacks:

1) Unpatched third party code
2) Lateral brute force and elevation
3) Hardcoded credentials
4) Forgotten password or registration
5) Misapplied cryptography
6) Cloud metadata
7) API client redirection
8) Direct object access
9) Internal applications
10) Many small issues combined leading to domain admin

It will certainly be interesting to see if there is further decline in reporting of memory-related flaws, while for web application flaws there is no hint that there will be a near-future decline in various input validation or output encoding-based flaws such as SQLi and XSS. This is in spite of these vulnerability classes being around for over 20 years, with the fixes and mitigations for such bug manifestations known for pretty much the same length of time.

13. Conclusions & recommendations for the future

It is a pity that the tech industry has not managed to kill off certain classes of bug over time, despite the most common vulnerabilities such as the OWASP Top 10 and memory corruption flaws being well known and understood for decades, and despite a wealth of available information on secure design, implementation and mitigation. There is clearly a lot more investment required around secure development lifecycles (SDL) and secure software development training.

There is also clearly much improvement required around disclosure, the issuance of CVEs and time allowed to fix. Vendors need to be more aware of the importance of disclosure and have clearly defined policies and processes for consuming vulnerability information and pursuing fixes or remediation in a timely manner. The lack of awareness in this area is quite striking (despite there being freely available standards such as ISO 29147) given the relative maturity of the cyber security industry and the efforts it puts into what is commonly pro-bono vulnerability disclosure. As researchers and disclosers, we also hold our hands up and accept that we need to be better at maintaining communication with vendors and working towards fixes.

To support this internally at NCC Group we have created a successful quarterly and annual bug finding prize as part of our recent initiatives. Each quarter, there is a prize for the most '1337 bug' (voted for by the consultancy team) logged in our bug tracking system. To be eligible for the prize the bug must be approved for release to the vendor which then forces initiation of the disclosure process. There are technology-related prizes per quarter and at the end of each year all bug submissions are given points based on bug type and criticality, resulting in a 'Bug Hunter of the Year' prize. Similarly, all four quarter winners of the '1337 bug' category are pitched against each other in a vote for 'Bug of the Year'. Winners of the annual competition get to attend BlackHat and Defcon in Las Vegas, or a similar security conference of their choosing. For reference, this year's annual winners were:

- Bug of the Year – Cedric Halbronn – Cisco ASA Remote Code Execution & Denial of Service Vulnerability [10]
- Bug Hunter of the Year – Soroush Dalili [11]

We perhaps also need to be more aggressive on our disclosure timelines if we feel that vendors aren’t cooperating, yet exposing their (and our) customers to high risk. Improvements are needed on the process of obtaining CVEs and there is likely an education piece here for vendors on why CVEs are important and understanding the process of requesting identifiers. Larger vendors who are not CNAs but who may regularly consume disclosed vulnerabilities in their products should consider application for CNA status in order to be able to obtain blocks of CVE IDs, thus removing any burden on (and time delay from) back-and-forth with MITRE. Note there is also an ISO standard (30111) [12] under development on the vulnerability handling process which should benefit vendors who perhaps need more guidance in this space.

Compliance and regulation may also play a part in improving disclosure and vulnerability fixes over the coming years. Regulations such as the General Data Protection Regulation (GDPR) may help focus minds around certain vulnerability classes, in particular those that when exploited could result in data breach such as SQLi.

Overall, we can’t draw too many conclusions or insights from the data as it stands due to the inconsistencies the process has experienced over time. However, as discussed in our introduction it is useful to lay down this data and analysis now as we can look to repeat this exercise over the coming years and certainly revisit this piece in nine years’ time to see what, if any, improvements have been made and perhaps what new classes of vulnerability are plaguing the world in 2027.

14. References

[1] https://www.bugzilla.org/

[2] https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

[3] https://www.nccgroup.trust/globalassets/our-research/uk/disclosure-policy/disclosure-policy.pdf

[4] https://www.nccgroup.trust/uk/our-research/?research=Technical+advisories

[5] https://forums.theregister.co.uk/forum/1/2016/05/25/mitre_fighter_deploys_name_logo_website_combo_in_cve_plea/

[6] http://standards.iso.org/ittf/PubliclyAvailableStandards/c045170_ISO_IEC_29147_2014.zip

[7] https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2018/june/finding-deserialisation-issues-has-never-been-easier-freddy-the-serialisation-killer/

[8] https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2017/august/when-a-web-application-ssrf-causes-the-cloud-to-rain-credentials-and-more/

[9] https://www.ncsc.gov.uk/blog-post/cyberuk-practice-track-1-vulnerabilities-and-bug-hunting

[10] https://www.nccgroup.trust/uk/about-us/newsroom-and-events/press-releases/2018/january/critical-security-vulnerability-found-in-business-firewalls/

[11] https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2018/may/smb-hash-hijacking-and-user-tracking-in-ms-outlook/

[12] https://www.iso.org/standard/53231.html