Vulnerability Type Distributions in CVE

Document version: 1.1 **Date:** May 22, 2007

Author: Steve Christey, Robert A. Martin

URL: http://cwe.mitre.org/documents/vuln-trends.html

Table of Contents

Introduction

For the past 5 years, CVE has been tracking the types of errors that lead to publicly reported vulnerabilities, and periodically reporting trends on a limited scale. The primary goal of this study is to better understand research trends using publicly reported vulnerabilities.

It should be noted that the data is obtained from an uncontrolled population, i.e., decentralized public reports from a research community with diverse goals and interests, with an equally diverse set of vendors and developers. More specialized, exhaustive, and repeatable methods could be devised to evaluate software security. But until such methods reach maturity and widespread acceptance, the overall state of software security can be viewed through the lens of public reports.

Summary of Results

1) The total number of publicly reported web application vulnerabilities has risen sharply, to the point where they have overtaken buffer overflows. This is probably due to ease of detection and exploitation of web vulnerabilities, combined with the proliferation of low-grade software applications written by inexperienced developers. In 2005 and 2006, cross-site scripting (XSS) was number 1, and SQL injection was number 2.

2) PHP remote file inclusion (RFI) skyrocketed to number 3 in 2006, almost a 1000% increase over the previous year. Because RFI allows arbitrary code execution on a vulnerable server, this is a worrisome trend, although proper configuration is frequently sufficient to eliminate it. This trend is likely a reflection of RFI's role in creating botnets using web servers [Evron].

3) Buffer overflows are still the number 1 issue as reported in operating system (OS) vendor advisories. XSS is still high in this category, at number 3 in both 2005 and 2006, although other web application vulnerabilities appear much less frequently.

4) Integer overflows, barely in the top 10 overall in the past few years, are number 2 for OS vendor advisories, behind buffer overflows. This might indicate expert researcher interest in high-profile software.

5) There are noticeable differences in the types of vulnerabilities being reported in open and closed source OS vendor advisories. These merit further investigation because they might reflect important differences in development, research, and disclosure practices.

6) The data is inconclusive regarding whether there is a concrete improvement in overall software security. While there is a rise in new vulnerability classes, and increasing diversity of vulnerability types, the raw numbers for older classes have not changed significantly. Further investigation is also required in this area.

Changes From October 2006 Report

A draft of this report was released in October 2006, due to widespread demand. While this paper is largely based on that report, the following differences are most significant:

1) The 2006 statistics cover the entire year.

2) An important statistical gap with CSRF is reported; see Table 1 analysis.

3) PHP remote file inclusion (RFI) is #4 overall, not #5, and is much closer to SQL injection in 2006 than originally reported. RFI's linkage to web server botnets is mentioned in the 'Summary of Results' and 'Overall Trends' sections.

4) The complete report has minor statistical discrepancies with the October report regarding total numbers of CVEs, due to (a) incidental additions of IDs for older issues occurring in late 2006, or (b) removal of some IDs because they were duplicates or later proven to be false reports. Both occurrences are relatively common for any vulnerability information repository that seeks to maintain historical accuracy.

5) Unsafe storage under the web document root (webroot) is number 10 for all of 2006, not 13.

Data Sets

Three main data sets were used in this analysis.

OVERALL: this data set consists of all CVEs that were first publicly reported in 2001 or later (earlier CVEs do not have the appropriate fields filled out.) CVE includes all types of software, whether from a major vendor or an individual hobbyist programmer, as long as the associated vulnerability has been reported by the developer or posted by a researcher or third party to sources such as mailing lists and vulnerability databases. CVE only includes distributable software, i.e., it does not include issues that are reported for custom software in specific web sites. While CVE data is incomplete, it is estimated that it is 80% complete relative to all major mailing lists and vulnerability databases, with the likely exception of data from 2003.

OS VENDOR: this data set identifies CVEs that are associated with operating system (OS) vendor advisories, which would capture vulnerabilities in the kernel, as well as applications that are supported by the OS vendor. The data was limited to CVEs that have one or more references from the following sources. For open source OS vendors, the following sources were used: DEBIAN, FREEBSD, MANDRAKE/MANDRIVA, NETBSD, OPENBSD, REDHAT, and SUSE. The closed source OS vendors included: AIXAPAR, APPLE, CISCO, HP, MS, MSKB, SCO, SGI, SUN, and SUNALERT. CVE does not have the internal data fields to support more fine-grained analysis for major non-OS vendors.

OPEN/CLOSED SOURCE: open and closed source operating system (OS) vendors were using the same methods and categories as described in the 'OS VENDOR' section. Because some closed source vendors such as Apple have significant codebase overlap with open source products, any overlapping CVEs were removed

from the data set. Both open and closed sets had at least 1700 vulnerabilities.

In each data set, vulnerabilities were not removed if they were marked as 'disputed.' Many disputes are incorrect or unresolved.

For 2006 data, 95% of all of CVE's primary data sources were covered, in order to offer the most complete data feasible for this year. The remaining issues are extremely complex or pose larger questions for CVE's content decisions. Due to resource limitations, MITRE was not able to achieve this level of completeness for earlier years.

Trend Table Color Key

In the HTML pages, the following color key is used for trend tables.

GRAY: used in comparisons to help visually separate one data set from another

RED: a top 10 for that year

GREEN: during that year, the vulnerability's rank was at least 5 points BELOW the average rank for that vulnerability

YELLOW: during that year, the vulnerability's rank was at least 5 points ABOVE the average rank for that vulnerability

So, green on the left indicates vulns with RISING popularity, as will yellow on the right. Green on the right indicates vulns with FALLING popularity, as will yellow on the left.

Table 1 Analysis: Overall Trends

The most notable trend is the sharp rise in public reports for vulnerabilities that are specific to web applications.

Buffer overflows were number 1 year after year, but that changed in 2005 with the rise of web application vulnerabilities, including cross-site scripting (XSS), SQL injection, and remote file inclusion, although SQL injection is not limited just to web applications. In fact, so far in 2006, buffer overflows are only #4.

There are probably several contributing factors to this increase in web vulnerabilities:

1) The most basic data manipulations for these vulnerabilities are very simple to perform, e.g., `'` for SQL injection and `<script>alert('hi')</script>` for XSS. This makes it easy for beginning researchers to quickly test large amounts of software.

2) There is a plethora of freely available web applications. Much of the code is alpha or beta, written by inexperienced programmers with easy-to-learn languages such as PHP, and distributed on high-traffic sites. The applications might have a small or non-existent user base. Such software is often rife with easy-to-find vulnerabilities, and it is often a target for beginning researchers. The large number of these 'fish-in-a-barrel' applications is probably a major contributor to the overall trends.

3) With XSS, every input has the potential to be an attack vector, which does not occur with other vulnerability types. This leaves more opportunity for a single mistake to occur in a program that otherwise protects against XSS. SQL injection also has many potential attack vectors.

4) Despite popular opinion that XSS is easily prevented, it has many subtleties and variants. Even solid applications can have flaws in them; consider non-standard browser behaviors that try to 'fix' malformed HTML, which might slip by a filter that uses regular expressions. Finally, until early 2006, the PHP interpreter had a vulnerability in which it did not quote error messages, but many researchers only reported the surface-level 'resultant' XSS instead of figuring out whether there was a different 'primary' vulnerability that led to the error.

5) There is some evidence that over the past couple of years, web defacers have taken an interest in performing and publishing their own research. This is probably due to the ease of finding vulnerabilities, combined with the presence of high-risk problems such as PHP file inclusion, which can be used to remotely install powerful, easily-available backdoor code. Based on customer posts to numerous vendor forums, there is solid evidence that remote file inclusion is regularly used to compromise web servers, which also helps to explain its popularity.

Overall Trends: Other Interesting Results

1) PHP remote file inclusion skyrocketed in 2006, nearly 1000% over the previous year. This is most likely a reflection of RFI's role in creating botnets using web servers [Evron].

2) For 2006, the top 5 vulnerability types are responsible for 57% of all CVEs. With over 35 vulnerability types used in this report, and dozens more as currently identified in CWE, this shows how most public reports concentrate only on a handful of vulnerability types.

3) Cross-Site Request Forgery (CSRF) remains a 'sleeping giant' [Grossman]. CSRF appears very rarely in CVE, less than 0.1% in 2006, but its true prevalence is probably far greater than this. This is in stark contrast to the results found by web application security experts including Jeremiah Grossman, RSnake, Andrew van der Stock, and Jeff Williams. These researchers regularly find CSRF during contract work, noting that it is currently not easy to detect automatically. The dearth of CSRF in CVE suggests that non-contract researchers are simply not investigating this issue. If (and when) researchers begin to focus on this issue, there will likely be a significant increase in CSRF reports.

4) Over the years, there has been a noticeable decline in shell metacharacters, symbolic link following, and directory traversal. It is unclear whether software is actually improving with respect to these problems, or if they are not investigated as frequently.

5) Information leaks appear regularly. There are 2 main reasons for the prominence: 'information leak' is a more general class than others (see CWE for more precise sub-categories), and when an error message includes a full path, that is usually categorized as an information leak, although it might be resultant from a separate primary vulnerability.

6) The inability to handle malformed inputs (dos-malform), which usually leads to a crash or hang, is also a general class. Malformed-input vulnerabilities have not been studied as closely as injection vulnerabilities, at least with respect to identifying the root cause of the problem. Also, many reports do not specify how an input is malformed. There are likely many cases in which a researcher accidentally triggers a more serious vulnerability but does not perform sufficient diagnosis to determine the primary issue. Finally, vendor reports might only identify an issue as being related to 'malformed input,' which obscures the primary cause.

7) As the percentage of buffer overflows has declined, there has been an increase in related vulnerability types, including integer overflows (int-overflow), signedness errors, and double frees (double-free). These are still very low-percentage, probably due to their relative newness and difficulty of detection compared to classic overflows. In addition, these newly emerging vulnerability types might be labeled as buffer overflows, since they often lead to buffer overflows, and the 'buffer overflow' term is used interchangeably for attack, cause, and effect.

8) Other interesting web application vulnerabilities are webroot (storage of sensitive files under the web document root), form-field (web parameter tampering), upload of files with executable extensions (e.g., file.php.gif), eval injection, and Cross-Site Request Forgery (CSRF).

Table 2 and 3 Analysis: OS vs. non-OS

Given the increase in web application vulnerabilities and the likelihood that it is partially due to researcher interest in software with small user bases, an analysis was performed based solely on advisories from operating system (OS) vendors. These advisories frequently include the OS kernel and key applications that are supported by the vendor. See the Data Sets section for more information. Unfortunately, more precise data sets could not be generated.

Table 2 provides the data for OS vendor advisories alone. Table 3 contrasts the OS vendor advisories with all other reported issues.

There are several notable results:

1) Integer overflows are heavily represented in OS vendor advisories, rising to number 2 in 2006, even though they represent less than 5% of vulnerabilities overall. This probably reflects growing interest by expert researchers in finding integer overflows, along with the tendency of expert researchers to evaluate widely deployed software. The affected software ranges widely, including the kernel, cryptographic modules, and multimedia file processors such as image viewers and music players. After 2004, many of the reported issues occur in libraries or common DLLs.

2) Buffer overflows are still #1. This is probably due to under-representation of web applications in OS advisories, relative to other CVEs. In addition, as related issues like integer overflows increase, they might be detected or reported as buffer

overflows, since buffer overflows are frequently resultant from integer overflows.

3) XSS is still very common, even in OS advisories, and it appears with nearly the same frequency as integer overflows in 2006. An informal analysis shows that the affected software includes web servers, web browsers, email clients, administrative interfaces, and Wiki/CMS.

4) With the exception of XSS, there is a wide gulf between web-related vulnerabilities in OS advisories and other issues. SQL injection is at number 7 for OS advisories, and PHP remote file inclusion is practically nonexistent. Many other web-related vulnerabilities occupy the bottom of the chart. For SQL injection, it is possible that most OS-supported applications do not use databases, or aren't web accessible. SQL injection vulnerabilities are not web-specific, but it seems that they are rarely reported for non-web applications, so it is possible that this reflects some researcher bias.

5) Directory traversal and format string vulnerabilities are frequently reported at a higher rate in OS vendor advisories than elsewhere. The reason is unclear, because these vulnerabilities are not restricted to local attack vectors, so one might expect that they would also appear regularly in web applications. However, it is likely that researchers do not focus on format strings because they are rarely exploitable for code execution in languages other than C. In the case of PHP, many PHP functions are subject to both remote file inclusion and directory traversal, and it might be that only the file inclusion is publicly reported. (In fact, the overlap is so close that this sometimes causes difficulties with classification).

6) In 2006 so far, more than a quarter of the OS vendor advisories did not have sufficient details to actually classify the vulnerability (type 'unk'), at 26.8%. This is in sharp contrast to the non-OS issues, which comprise less than 8%. However, because of the data sets in question, the non-OS CVEs will include many non-coordinated disclosures that would, by their

nature, provide more details. Table 4 demonstrates that it is not just closed source vendor advisories that omit sufficient details for vulnerability classification.

7) The 'top 5' and 'top 10' vulnerabilities in each year are a much smaller percentage of total vulnerabilities in OS vendor advisories than non-OS issues. For example, the 5 most common vulnerabilities in 2006 accounted for 30.2% of OS vendor issues, but 65.3% for non-OS. For OS issues, this suggests an increasing diversity in the kinds of vulnerabilities being reported, whereas for other issues, that diversity appears to be decreasing. This is also reflected in the 'other' category, in which OS vendors have a much larger percentage of 'other' issues in 2006 than non-OS. However, this could be another reflection of the domination of web application vulnerabilities.

Table 4 Analysis: Open and Closed Source

Table 4 compares the vulnerability type distribution between the open source and closed source operating system (OS) vendors. See the 'Data Sets' section for more information on how the data sets were generated. As a reminder, CVEs that overlapped both open and closed source sets were omitted.

** IMPORTANT ** It is inappropriate to use these results to objectively compare the relative security of open and closed source products, so the report excludes raw numbers. Both sets had at least 2500 vulnerabilities. There are too many variations in vendor advisory release policies, possible differences in research techniques, and other factors cited in [Christey]. And, simply put, there is too much potential for raw numbers to be misused and misinterpreted.

However, some results pose interesting questions that merit more in-depth investigation. These discrepancies might reflect differences in vulnerability research techniques, researcher sub-

communities, vendor disclosure policies, and development practices and APIs, but this has not been proven.

After the release of the draft in October 2006, various vendor and research representatives were consulted, but there were not any clear conclusions. The research and vendor communities are encouraged to investigate the underlying causes for these differences, which could provide lessons learned for all software developers, open and closed source alike.

Some of the most notable results are:

1) The percentage of 'unknown' vulnerabilities - those that could not be classified due to lack of details - is significantly higher in closed source than open source advisories, at 43% in 2006, compared to only 8% for open source. With such a wide discrepancy, it is difficult to know whether any of the remaining results in this section are significant.

2) Buffer overflows are number 1 for both open and closed, with roughly the same percentage in each year, with the exception of 2004.

3) Symbolic link vulnerabilities appear at a higher rate in open source than closed source, although this might be due to the non-Unix OSes in the data set. While Windows has 'shortcuts' (.LNK) that are similar to Unix links, they appear very rarely in Microsoft advisories, or for Windows-based applications. It is not clear whether this is due to under-research or API/development differences. The authors recall that at least one researcher for a Linux distribution regularly investigated symbolic link issues in 2004 and 2005, so researcher bias might also be a factor.

4) Format string vulnerabilities appear more frequently in open source. There are probably several factors. First, susceptible API library calls such as printf() are easily found in source code using crude methods, whereas binary reverse engineering techniques are not conducted by many researchers (this might

also be an explanation for symbolic link issues). Second, many format string problems seem to occur in rarely-triggered error conditions, which makes them more difficult to test with black box methods.

Perhaps most surprising: in 2006, the non-Unix closed source advisories barely covered any format strings at all. It is not clear why there would be such a radical difference.

5) Malformed-input vulnerabilities usually appeared more frequently in closed source advisories than open source, except for 2006. This historical tendency might be due to a lack of details in closed source advisories. If an advisory mentions a problem due to 'malformed data,' it might be assigned the dos-malform type. Another factor might be due to black box techniques. It seems likely that fuzzers and other tools would be used more frequently against closed source products than open source, but this is not known. A third factor might be modifications in CVE's data entry procedures, which eventually began to enter 'unknown' flaw types for vague terms such as 'memory corruption.'

6) XSS vulnerabilities appear more frequently in open source advisories than closed, but this might be a reflection of vendor release policies for advisories. It seems that open source vendors are more likely to release advisories for smaller packages.

7) Integer overflows have been roughly the same rank for open and closed source. This is a curious similarity, since one might not expect open and closed source analysis techniques to be equally capable in finding these problems.

8) Another interesting example is in the use of default or hard-coded passwords. Over the years, very few open source vendor advisories have mentioned default passwords, whereas they appear with some regularity in closed source advisories, even in the top 10 as recently as 2005. It is not clear whether this is a

difference in shipping/configuration practices or vendor disclosure policies.

9) During the October 2006 analysis, it was discovered that shell metacharacter issues appear less frequently in non-Unix closed source than other closed source advisories. This result was verified using the latest data; it is not evident in Table 4. This could be due to usage patterns of API functions such as CreateProcess() for Windows, and system() for Unix. This result is being reported because it is the most concrete example of how API functions might play a role in implementation-level vulnerabilities.

Possible Future Work

1) The vulnerability types could be tied to other CVE-normalized data, such as IDS, incident databases, or vulnerability scanning results. This could determine the types of vulnerabilities that are being actively exploited or detected in real-world enterprises.

2) More precise classification could be informative. Approximately 15% of CVEs have vulnerability types that cannot be described using the current classification scheme. Another 10% are 'unknown' vulnerabilities whose disclosures do not have sufficient details to determine any vulnerability type, but this problem is unavoidable, since some vendors do not release these details.

3) A crude measure of researcher diversity might be possible by linking data to other vulnerability databases that record this information. This could be used to determine if the raw number of researchers is increasing (probably), how that rate is increasing relative to the number of vulnerabilities (unknown), and how many different bug types are found by the average researcher (probably fairly small). If such data is available, then a further breakdown could be performed based on professional researchers versus others.

4) More precise data sets could be identified, such as a cross-section of market leaders in various product categories, not just OS vendor advisories. CVE does not record this type of information.

Notes on Potential Bias

The diversity of both researchers and vendor disclosure practices introduces several unmeasurable biases, as described in more detail in [Christey].

In the overall results, 2003's issues have nearly 20% with vulnerabilities that are 'not specified' by the CVE analyst, which is inconsistent with statistics from other years. Many of these vulnerabilities were briefly reviewed in October 2006, and they are in fact of type 'other.' This discrepancy has not been sufficiently explained, although it is probably at least partially due to the relative percentage of CVEs in OS vendor advisories to other CVEs, since 2003 was a low-output year for CVE and thus the concentration was in high-priority software.

Some vulnerability types are probably under-represented due to classification difficulty. For example, the 'form-field' type (web parameter tampering) might occasionally get classified as an authentication error, depending on how the original researcher reports the issue.

(In)Frequently Asked Questions

1) *Why aren't you giving out raw numbers for open vs. closed source?*

   Answer: we already said why. See paragraph 2 of the Table 4 analysis for a reminder, the one marked 'IMPORTANT.'

2) *Why did you release the draft report in October, without waiting for complete 2006 data?*

   Answer: when MITRE mentioned the preliminary results at the Cyber Security Executive Summit on September 13, there was a lot more interest than we had originally anticipated. We hoped that follow-up discussion of the results might help us to provide a better report when 2006 was complete.

3) *How does this compare with the other summaries you've posted in the past? Why have the numbers and percentages changed for older years?*

   Answer: (1) we occasionally add CVEs for older issues, (2) some of the previously released summaries were cumulative instead of offering a year-by-year breakdown, and (3) eventually, as a new type of vulnerability is reported more frequently, the CVE project notices it enough to give it a name, or at least a type. Once we do that, we can go back and update the older CVEs that also had the issue. However, we often rely on keyword searches in CVE descriptions for doing these kinds of updates. The earliest reports of new vulnerability types probably don't get captured fully, because CVE descriptions frequently vary in the early days or months of a new vulnerability type. Most updates to these vulnerability trends trigger an informal review of the 'other' vulnerabilities for the data set in order to update the type fields.

4) *There are a lot more vulnerability types than what you've covered.*

   Answer: That's an observation, not a question. If a certain vulnerability type is not on the list, then it probably didn't appear frequently enough for the CVE project to track closely. There are several reasons: (1) the vulnerability type is selected from a large dropdown menu during CVE refinement, but also (2) our work in the Common Weakness Enumeration (CWE) is producing hundreds of vuln types, and we want that to become a little more stable before doing the next round of modifications to CVE data. Finally, (3) with approximately 4,000 vulnerabilities marked 'other' or 'not specified', it is cost-prohibitive to review each CVE when the set of categories is updated.

5) *Why isn't my favorite web vulnerability here?*

   Answer: Many web vulnerabilities are difficult to classify because they are 'multi-factor,' i.e., they are composed of multiple bugs, weaknesses, and/or design limitations. Other web issues are really just specialized attacks that use other primary vulnerabilities. For example, most HTTP response splitting problems rely on CRLF injection, so they are classified under CRLF injection.

Credits

Large-scale trend analyses like this are not possible without the body of knowledge that has been formed by hundreds or thousands of researchers, from hobbyists to professionals.

Thanks to the following for substantive feedback on the initial draft, sometimes in the form of a question that required more investigation: Bill Heinbockel, Chris Wysopal, and Mark Curphey. Thanks to Jeremiah Grossman, Andrew van der Stock, RSnake, and Jeff Williams for their feedback on CSRF detection.

References

[Christey] 'Open Letter on the Interpretation of 'Vulnerability Statistics'', Steve Christey, Bugtraq, Full-Disclosure January 5, 2006, http://lists.grok.org.uk/pipermail/full-disclosure/2006-January/041028.html

[Evron] 'Web server botnets and hosting farms as attack platforms', Gadi Evron, Kfir Damari & Noam Rathaus, Virus Bulletin, February 2007

[Grossman] 'CSRF, the sleeping giant', Jeremiah Grossman, http://jeremiahgrossman.blogspot.com/2006/09/csrf-sleeping-giant.html

Flaw Terminology

Type:	auth
CWE:	CWE-289, CWE-288, CWE-302, CWE-305, CWE 294, CWE-290, CWE-287, CWE-303
Description:	Weak/bad authentication problem
Type:	buf
CWE:	CWE-119, CWE-120
Description:	Buffer overflow
Type:	CF
CWE:	none
Description:	General configuration problem, not perm or default
Type:	crlf
CWE:	CWE-93
Description:	CRLF injection

Type:	crypt
CWE:	CWE-310, CWE-311, CWE-347, CWE-320, CWE-325
Description:	Cryptographic error (poor design or implementation), including plaintext storage/transmission of sensitive information.

Type:	CSRF
CWE:	CWE-352
Description:	Cross-Site Request Forgery (CSRF)

Type:	default
CWE:	N/A
Description:	Insecure default configuration, e.g., passwords or permissions

Type:	design
CWE:	none
Description:	Design problem, generally in protocols or programming languages. Since 2005, its use has been limited due to the highly general nature of this type.

Type:	dos-flood
CWE:	CWE-400
Description:	DoS caused by flooding with a large number of *legitimately formatted* requests/etc.; normally DoS is a crash, or spending a lot more time on a task than it ‘should’

Type:	dos-malform
CWE:	CWE-238, CWE-234, CWE-166, CWE-230, many others
Description:	DoS caused by malformed input

Type:	dos-release
CWE:	CWE-404
Description:	DoS because system does not properly release resources

Type:	dot
CWE:	CWE-22, CWE-23, CWE-36
Description:	Directory traversal (file access via '..' or variants)

Type:	double-free
CWE:	CWE-415
Description:	Double-free vulnerability

Type:	eval-inject
CWE:	CWE-95
Description:	Eval injection

Type:	form-field
CWE:	CWE-472
Description:	CGI program inherently trusts form field that should not be modified (i.e., should be stored locally)

Type:	format-string
CWE:	CWE-134
Description:	Format string vulnerability; user can inject format specifiers during string processing.

Type:	infoleak
CWE:	CWE-205, CWE-212, CWE-203, CWE-209, CWE-207, CWE-200, CWE-215, others
Description:	Information leak by a product, which is not the result of another vulnerability; typically by design or by producing different 'answers' that suggest the state; often related to configuration / permissions or error reporting/handling.

Type:	int-overflow
CWE:	CWE-190
Description:	A numeric value can be incremented to the point where it overflows and begins at the minimum value, with security implications. Overlaps signedness errors.
Type:	link
CWE:	CWE-61, CWE-64
Description:	Symbolic link following
Type:	memleak
CWE:	CWE-401
Description:	Memory leak (doesn't free memory when it should); use this instead of dos-release
Type:	metachar
CWE:	CWE-78
Description:	Unescaped shell metacharacters or other unquoted ‘special’ char's; currently includes SQL injection but not XSS.
Type:	msdos-device
CWE:	CWE-67
Description:	Problem due to file names with MS-DOS device names.
Type:	not-specified
CWE:	none
Description:	The CVE analyst has not assigned a flaw type to the issue, typically similar to ‘other’.
Type:	other
CWE:	none
Description:	Other vulnerability; issue could not be described with an available type at the time of analysis.

Type:	pass
CWE:	CWE-259
Description:	Default or hard-coded password
Type:	perm
CWE:	CWE-276
Description:	Assigns bad permissions, improperly calculates permissions, or improperly checks permissions
Type:	php-include
CWE:	CWE-98
Description:	PHP remote file inclusion
Type:	priv
CWE:	CWE-266, CWE-274, CWE-272, CWE-250, CWE-264, CWE-265, CWE-268, CWE-270, CWE-271, CWE-269, CWE-267
Description:	Bad privilege assignment, or privileged process/action is unprotected/unauthenticated.
Type:	race
CWE:	CWE-362, CWE-366, CWE-364, CWE-367, CWE-421, CWE-368, CWE-363, CWE-370
Description:	General race condition (NOT SYMBOLIC LINK FOLLOWING (link)!)
Type:	rand
CWE:	CWE-330, CWE-331, CWE-332, CWE-338, CWE-342, CWE-341, CWE-339, others
Description:	Generation of insufficiently random numbers, typically by using easily guessable sources of ‘random’ data
Type:	relpath
CWE:	CWE-426, CWE-428, CWE-114
Description:	Untrusted search path vulnerability - Relies on search paths to find other executable programs or files, opening up to Trojan horse attacks, e.g., PATH environment variable in Unix.

Type:	sandbox
CWE:	CWE-265
Description:	Java/etc. sandbox escape - NOT BY DOT-DOT!

Type:	signedness
CWE:	CWE-195, CWE-196
Description:	Signedness error; a numeric value in one format/representation is improperly handled when it is used as if it were another format/representation. Overlaps integer overflows and array index errors.

Type:	spoof
CWE:	CWE-290, CWE-350, CWE-347, CWE-345, CWE-247, CWE-292, CWE-291
Description:	Product is vulnerable to spoofing attacks, generally by not properly verifying authenticity.

Type:	sql-inject
CWE:	CWE-89
Description:	SQL injection vulnerability

Type:	type-check
CWE:	unknown
Description:	Product incorrectly identifies the type of an input parameter or file, then dispatches the wrong ‘executable’ (possibly itself) to process the input, or otherwise misrepresents the input in a security-critical way.

Type:	undiag
CWE:	none
Description:	Undiagnosed vulnerability; report contains enough details so that the type could be determined by additional in-depth research, such as an un-commented exploit, or diffs in an open source product.

Type:	unk
CWE:	none
Description:	Unknown vulnerability; report is too vague to determine type of issue.

Type:	upload
CWE:	CWE-434
Description:	Product does not restrict the extensions for files that can be uploaded to the web server, leading to code execution if executable extensions are used in filenames, such as .asp, .php, and .shtml.
Type:	webroot
CWE:	CWE-219, CWE-433
Description:	Storage of sensitive data under web document root with insufficient access control.

Type:	XSS
CWE:	CWE-79, CWE-80, CWE-87, CWE-85, CWE-82, CWE-81, CWE-83, CWE-84
Description:	Cross-site scripting (aka XSS)

Table 1: Overall Results

Rank	Flaw	TOTAL	2001	2002	2003	2004	2005	2006
Total		18809	1432	2138	1190	2546	4559	6944
[1]	XSS	13.8%	02.2% (11)	08.7% (2)	07.5% (2)	10.9% (2)	16.0% (1)	18.5% (1)
		2595	31	187	89	278	728	1282
[2]	buf	12.6%	19.5% (1)	20.4% (1)	22.5% (1)	15.4% (1)	09.8% (3)	07.8% (4)
		2361	279	436	268	392	445	541
[3]	sql-inject	09.3%	00.4% (28)	01.8% (12)	03.0% (4)	05.6% (3)	12.9% (2)	13.6% (2)
		1754	6	38	36	142	588	944
[4]	php-include	05.7%	00.1% (31)	00.3% (26)	01.0% (13)	01.4% (10)	02.1% (6)	13.1% (3)
		1065	1	7	12	36	96	913
[5]	dot	04.7%	08.9% (2)	05.1% (4)	02.9% (5)	04.2% (4)	04.3% (4)	04.5% (5)
		888	127	110	34	106	196	315
[6]	infoleak	03.4%	02.6% (9)	04.2% (5)	02.8% (6)	03.8% (5)	03.8% (5)	03.1% (6)
		646	37	89	33	98	175	214
[7]	dos-malform	02.8%	04.8% (3)	05.2% (3)	02.5% (8)	03.4% (6)	01.8% (8)	02.0% (7)
		521	69	111	30	86	83	142
[8]	link	01.8%	04.5% (4)	02.1% (9)	03.5% (3)	02.8% (7)	01.9% (7)	00.4% (16)
		341	64	45	42	72	87	31
[9]	format-string	01.7%	03.2% (7)	01.8% (10)	02.7% (7)	02.4% (8)	01.7% (9)	00.9% (11)
		317	46	39	32	62	76	62
[10]	crypt	01.5%	03.8% (5)	02.7% (6)	01.5% (9)	00.9% (16)	01.5% (10)	00.8% (13)
		278	55	58	18	22	69	56
[11]	priv	01.3%	02.5% (10)	02.2% (8)	01.1% (12)	01.3% (11)	01.5% (11)	00.8% (14)
		249	36	46	13	33	67	54
[12]	perm	01.3%	02.7% (8)	01.8% (11)	01.3% (11)	00.9% (15)	01.1% (13)	01.1% (9)
		241	39	39	15	24	48	76
[13]	metachar	01.2%	03.8% (6)	02.6% (7)	00.7% (18)	01.0% (14)	01.3% (12)	00.4% (17)
		233	55	56	8	26	59	29
[14]	int-overflow	01.0%	00.1% (32)	00.4% (25)	01.3% (10)	01.8% (9)	00.8% (14)	01.2% (8)
		190	1	8	16	47	36	82
[15]	auth	00.8%	01.5% (13)	01.3% (15)	00.5% (19)	00.7% (17)	00.5% (19)	00.9% (12)
		155	22	27	6	17	21	62
[16]	dos-flood	00.7%	02.0% (12)	01.7% (13)	00.5% (20)	01.2% (12)	00.2% (27)	00.4% (19)
		138	29	36	6	31	11	25

Table 1: Overall Results (continued)

[17]	**pass**	**00.7%**	01.1% (17)	01.3% (14)	00.2% (29)	01.1% (13)	00.8% (15)	00.4% (18)
		135	*16*	*28*	*2*	*28*	*36*	*25*
[18]	**webroot**	**00.6%**	00.1% (29)	00.2% (32)	00.3% (25)	00.2% (29)	00.7% (16)	01.0% (10)
		117	*2*	*5*	*3*	*5*	*33*	*69*
[19]	**form-field**	**00.5%**	00.7% (23)	00.8% (17)	00.5% (21)	00.2% (26)	00.4% (20)	00.6% (15)
		97	*10*	*17*	*6*	*6*	*19*	*39*
[20]	**race**	**00.4%**	00.5% (26)	00.4% (24)	00.7% (17)	00.4% (21)	00.6% (17)	00.3% (23)
		81	*7*	*8*	*8*	*10*	*26*	*22*
[21]	**relpath**	**00.4%**	00.8% (22)	00.3% (29)	00.8% (15)	00.5% (18)	00.3% (22)	00.3% (21)
		81	*12*	*7*	*10*	*14*	*15*	*23*
[22]	**memleak**	**00.4%**	01.1% (18)	00.2% (33)	00.4% (22)	00.5% (19)	00.3% (23)	00.2% (27)
		66	*16*	*5*	*5*	*13*	*15*	*12*
[23]	**crlf**	**00.3%**	...	00.2% (31)	00.2% (26)	00.5% (20)	00.4% (21)	00.3% (20)
		62	*0*	*5*	*2*	*13*	*18*	*24*
[24]	**msdos-device**	**00.3%**	01.0% (19)	00.6% (19)	00.9% (14)	00.2% (27)	00.2% (28)	00.0% (35)
		57	*15*	*13*	*11*	*6*	*10*	*2*
[25]	**upload**	**00.3%**	...	00.0% (35)	00.1% (30)	00.2% (25)	00.5% (18)	00.3% (22)
		54	*0*	*1*	*1*	*6*	*23*	*23*
[26]	**rand**	**00.3%**	01.2% (15)	00.6% (20)	00.3% (24)	00.2% (32)	00.0% (35)	00.2% (25)
		52	*17*	*12*	*3*	*4*	*2*	*14*
[27]	**spoof**	**00.3%**	01.0% (20)	00.3% (27)	00.1% (33)	00.1% (33)	00.2% (26)	00.2% (26)
		51	*15*	*7*	*1*	*3*	*11*	*14*
[28]	**sandbox**	**00.3%**	01.2% (14)	01.0% (16)	...	00.2% (28)	00.0% (34)	00.1% (33)
		50	*17*	*22*	*0*	*5*	*2*	*4*
[29]	**default**	**00.3%**	01.1% (16)	00.7% (18)	00.1% (32)	00.2% (24)	00.1% (33)	00.1% (29)
		49	*16*	*16*	*1*	*6*	*3*	*7*
[30]	**signedness**	**00.2%**	00.1% (30)	00.4% (23)	00.8% (16)	00.2% (22)	00.3% (24)	00.1% (30)
		42	*1*	*8*	*9*	*6*	*12*	*6*
[31]	**CF**	**00.2%**	00.7% (24)	00.3% (30)	00.2% (27)	...	00.1% (31)	00.1% (28)
		33	*10*	*7*	*2*	*0*	*4*	*10*
[32]	**eval-inject**	**00.2%**	...	00.0% (36)	...	00.0% (35)	00.2% (25)	00.3% (24)
		31	*0*	*1*	*0*	*1*	*11*	*18*
[33]	**dos-release**	**00.2%**	00.9% (21)	00.5% (21)	00.2% (28)	00.2% (31)	...	...
		30	*13*	*10*	*2*	*5*	*0*	*0*
[34]	**design**	**00.1%**	00.6% (25)	00.4% (22)	00.1% (31)	00.0% (34)	00.1% (32)	00.0% (34)
		23	*8*	*8*	*1*	*1*	*3*	*2*
[35]	**double-free**	**00.1%**	...	00.1% (34)	00.3% (23)	00.2% (23)	00.1% (30)	00.1% (32)

Table 1: Overall Results (concluded)

		22	*0*	*2*	*4*	*6*	*5*	*5*
[36]	**CSRF**	**00.1%**	...	00.0% (37)	...	00.2% (30)	00.2% (29)	00.1% (31)
		19	*0*	*1*	*0*	*5*	*8*	*5*
[37]	**type-check**	**00.1%**	00.4% (27)	00.3% (28)	...	...	00.0% (36)	00.0% (36)
		15	*6*	*7*	*0*	*0*	*1*	*1*
UNKNOWN/UNSPECIFIED ITEMS								
n/a	**unk**	**09.3%**	07.9%	07.1%	07.0%	08.3%	08.9%	11.2%
		1743	*113*	*151*	*83*	*211*	*405*	*780*
n/a	**other**	**15.0%**	16.7%	19.0%	11.9%	17.2%	13.3%	14.4%
		2826	*239*	*406*	*142*	*437*	*605*	*997*
n/a	**undiag**	**00.0%**	00.0%	00.0%	00.0%	00.0%	00.0%	00.0%
		1	*0*	*0*	*0*	*0*	*1*	*0*
n/a	**not-specified**	**05.8%**	00.1%	02.8%	19.8%	11.1%	11.1%	00.2%
		1100	*2*	*59*	*236*	*283*	*506*	*14*

Top 5/10 Diversity Percentages per year

For the 'top N' vulnerabilities in each year, the table identifies the total percentage of overall vulnerabilities. For example, a figure of 45.0 for Top 5 says that the Top 5 accounted for 45% of all reported vulnerabilities in that year. This provides a rough estimate of how diverse the reported vulnerabilities were.

Top n	TOTAL	2001	2002	2003	2004	2005	2006
5	46.1	41.5	43.6	39.4	39.9	46.8	57.5
10	57.3	56.3	55	50.2	51.7	55.8	65.9

Table 2: OS Vendors

Rank	Flaw	TOTAL	2001	2002	2003	2004	2005	2006
Total		4893	443	664	530	745	1216	1295
[1]	buf	19.6%	21.0% (1)	26.8% (1)	24.7% (1)	20.4% (1)	16.0% (1)	16.1% (1)
		958	*93*	*178*	*131*	*152*	*195*	*209*
[2]	link	03.8%	07.4% (2)	03.3% (4)	04.2% (2)	05.1% (2)	04.2% (2)	01.5% (8)
		186	*33*	*22*	*22*	*38*	*51*	*20*
[3]	dos-malform	03.7%	05.6% (3)	06.2% (2)	02.6% (4)	04.4% (4)	01.8% (7)	03.6% (4)
		182	*25*	*41*	*14*	*33*	*22*	*47*
[4]	XSS	03.4%	01.6% (14)	04.4% (3)	03.0% (3)	01.5% (7)	04.2% (3)	04.2% (3)
		168	*7*	*29*	*16*	*11*	*51*	*54*
[5]	int-overflow	02.9%	...	01.2% (12)	02.3% (6)	04.6% (3)	02.1% (6)	04.7% (2)
		140	*0*	*8*	*12*	*34*	*25*	*61*
[6]	format-string	02.3%	05.2% (4)	01.5% (9)	02.3% (5)	02.8% (5)	02.4% (5)	01.5% (9)
		114	*23*	*10*	*12*	*21*	*29*	*19*
[7]	priv	01.9%	04.1% (5)	02.3% (6)	00.8% (14)	00.8% (13)	02.5% (4)	01.6% (5)
		95	*18*	*15*	*4*	*6*	*31*	*21*
[8]	perm	01.7%	04.1% (6)	02.1% (8)	01.1% (9)	01.1% (9)	01.6% (8)	01.3% (11)
		83	*18*	*14*	*6*	*8*	*20*	*17*
[9]	dot	01.4%	01.6% (12)	01.5% (10)	01.1% (8)	01.6% (6)	01.2% (11)	01.4% (10)
		68	*7*	*10*	*6*	*12*	*15*	*18*
[10]	infoleak	01.3%	00.9% (19)	01.2% (13)	01.1% (11)	01.1% (10)	01.3% (10)	01.6% (6)
		63	*4*	*8*	*6*	*8*	*16*	*21*
[11]	metachar	01.2%	02.0% (9)	02.6% (5)	00.8% (13)	00.7% (17)	01.2% (12)	00.8% (13)
		60	*9*	*17*	*4*	*5*	*15*	*10*
[12]	race	01.1%	01.1% (17)	00.9% (16)	00.4% (19)	00.9% (11)	01.6% (9)	01.1% (12)
		53	*5*	*6*	*2*	*7*	*19*	*14*
[13]	sql-inject	01.0%	00.2% (28)	00.6% (19)	01.1% (7)	00.7% (16)	00.9% (14)	01.6% (7)
		48	*1*	*4*	*6*	*5*	*11*	*21*
[14]	crypt	00.8%	01.6% (11)	01.4% (11)	01.1% (10)	00.4% (18)	00.4% (18)	00.6% (15)
		38	*7*	*9*	*6*	*3*	*5*	*8*
[15]	memleak	00.8%	02.0% (10)	00.6% (18)	00.8% (16)	00.9% (12)	00.9% (13)	00.2% (24)
		38	*9*	*4*	*4*	*7*	*11*	*3*
[16]	sandbox	00.7%	02.7% (7)	02.1% (7)	...	00.1% (22)	00.2% (28)	00.2% (23)
		32	*12*	*14*	*0*	*1*	*2*	*3*

Table 2: OS Vendors (continued)

[17]	**relpath**	**00.6%**	01.6% (13)	00.3% (25)	00.4% (18)	01.1% (8)	00.2% (25)	00.7% (14)
		31	*7*	*2*	*2*	*8*	*3*	*9*
[18]	**dos-flood**	**00.6%**	02.5% (8)	00.6% (21)	00.2% (24)	00.3% (21)	00.2% (27)	00.6% (16)
		29	*11*	*4*	*1*	*2*	*3*	*8*
[19]	**auth**	**00.5%**	01.4% (16)	01.1% (14)	00.6% (17)	00.3% (20)	00.3% (19)	00.3% (19)
		26	*6*	*7*	*3*	*2*	*4*	*4*
[20]	**signedness**	**00.5%**	00.2% (23)	00.9% (15)	00.9% (12)	00.4% (19)	00.6% (15)	00.2% (22)
		25	*1*	*6*	*5*	*3*	*7*	*3*
[21]	**pass**	**00.5%**	00.2% (26)	00.8% (17)	00.2% (23)	00.8% (14)	00.3% (22)	00.5% (17)
		23	*1*	*5*	*1*	*6*	*4*	*6*
[22]	**double-free**	**00.4%**	...	00.3% (30)	00.8% (15)	00.8% (15)	00.3% (23)	00.3% (18)
		20	*0*	*2*	*4*	*6*	*4*	*4*
[23]	**rand**	**00.3%**	01.4% (15)	00.5% (22)	00.2% (21)	00.1% (24)	...	00.2% (28)
		13	*6*	*3*	*1*	*1*	*0*	*2*
[24]	**crlf**	**00.3%**	...	00.5% (23)	00.2% (26)	...	00.4% (17)	00.3% (21)
		13	*0*	*3*	*1*	*0*	*5*	*4*
[25]	**spoof**	**00.2%**	00.2% (24)	00.3% (27)	...	...	00.3% (24)	00.3% (20)
		11	*1*	*2*	*0*	*0*	*4*	*4*
[26]	**form-field**	**00.2%**	00.5% (22)	00.3% (28)	00.2% (20)	...	00.4% (16)	...
		10	*2*	*2*	*1*	*0*	*5*	*0*
[27]	**default**	**00.2%**	00.2% (27)	00.5% (24)	...	00.1% (23)	00.2% (26)	00.2% (30)
		10	*1*	*3*	*0*	*1*	*3*	*2*
[28]	**CF**	**00.2%**	00.9% (20)	00.2% (31)	00.2% (25)	...	...	00.2% (25)
		9	*4*	*1*	*1*	*0*	*0*	*3*
[29]	**type-check**	**00.2%**	00.7% (21)	00.6% (20)	...	...	00.1% (30)	...
		8	*3*	*4*	*0*	*0*	*1*	*0*
[30]	**dos-release**	**00.1%**	00.9% (18)	00.3% (26)	00.2% (22)	...	...	...
		7	*4*	*2*	*1*	*0*	*0*	*0*
[31]	**eval-inject**	**00.1%**	...	...	...	...	00.3% (20)	00.2% (29)
		6	*0*	*0*	*0*	*0*	*4*	*2*
[32]	**php-include**	**00.1%**	...	...	...	...	00.3% (21)	00.2% (27)
		6	*0*	*0*	*0*	*0*	*4*	*2*
[33]	**design**	**00.1%**	00.2% (25)	00.3% (29)	00.2% (28)	...	00.1% (29)	...
		5	*1*	*2*	*1*	*0*	*1*	*0*
[34]	**CSRF**	**00.0%**	...	...	...	...	00.1% (31)	00.1% (31)
		2	*0*	*0*	*0*	*0*	*1*	*1*
[35]	**webroot**	**00.0%**	...	...	...	...	...	00.2% (26)

Table 2: OS Vendors (concluded)

		2	*0*	*0*	*0*	*0*	*0*	*2*
[36]	**msdos-device**	**00.0%**	...	...	00.2% (27)	...	...	...
		1	*0*	*0*	*1*	*0*	*0*	*0*
[37]	**upload**	**00.0%**	...	...	...	...	00.1% (32)	...
		1	*0*	*0*	*0*	*0*	*1*	*0*
UNKNOWN/UNSPECIFIED ITEMS								
n/a	**unk**	**16.9%**	12.4%	12.5%	10.6%	12.3%	16.1%	26.8%
		829	*55*	*83*	*56*	*92*	*196*	*347*
n/a	**other**	**17.3%**	15.3%	15.8%	12.1%	12.1%	14.9%	26.2%
		847	*68*	*105*	*64*	*90*	*181*	*339*
n/a	**undiag**	**00.0%**	00.0%	00.0%	00.0%	00.0%	00.1%	00.0%
		1	*0*	*0*	*0*	*0*	*1*	*0*
n/a	**not-specified**	**12.9%**	00.2%	05.9%	25.7%	24.6%	21.9%	00.5%
		632	*1*	*39*	*136*	*183*	*266*	*7*

Top 5/10 Diversity Percentages per year

For the 'top N' vulnerabilities in each year, the table identifies the total percentage of overall vulnerabilities. For example, a figure of 45.0 for Top 5 says that the Top 5 accounted for 45% of all reported vulnerabilities in that year. This provides a rough estimate of how diverse the reported vulnerabilities were.

Top n	TOTAL	2001	2002	2003	2004	2005	2006
5	33.4	43.3	43.3	36.8	37.3	29.3	30.2
10	42	56.6	52.8	43.5	43.7	37.7	37.8

Table 3: OS Vendors vs. Others

Rank	Flaw	TOTAL	2001	2002	2003	2004	2005	2006
Total	OS-ven	4893	443	664	530	745	1216	1295
	Other	13916	989	1474	660	1801	3343	5649
[1]	XSS	03.4%	01.6% (12)	04.4% (3)	03.0% (3)	01.5% (7)	04.2% (2)	04.2% (3)
		168	7	29	16	11	51	54
		17.4%	02.4% (8)	10.7% (2)	11.1% (2)	14.8% (1)	20.3% (1)	21.7% (1)
		2427	24	158	73	267	677	1228
[2]	buf	19.6%	21.0% (1)	26.8% (1)	24.7% (1)	20.4% (1)	16.0% (1)	16.1% (1)
		958	93	178	131	152	195	209
		10.1%	18.8% (1)	17.5% (1)	20.8% (1)	13.3% (2)	07.5% (3)	05.9% (4)
		1403	186	258	137	240	250	332
[3]	sql-inject	01.0%	00.2% (25)	00.6% (21)	01.1% (7)	00.7% (17)	00.9% (14)	01.6% (5)
		48	1	4	6	5	11	21
		12.3%	00.5% (25)	02.3% (8)	04.5% (3)	07.6% (3)	17.3% (2)	16.3% (2)
		1706	5	34	30	137	577	923
[4]	php-include	00.1%	...	...	...	...	00.3% (19)	00.2% (28)
		6	0	0	0	0	4	2
		07.6%	00.1% (30)	00.5% (22)	01.8% (10)	02.0% (8)	02.8% (6)	16.1% (3)
		1059	1	7	12	36	92	911
[5]	dot	01.4%	01.6% (13)	01.5% (9)	01.1% (10)	01.6% (6)	01.2% (11)	01.4% (10)
		68	7	10	6	12	15	18
		05.9%	12.1% (2)	06.8% (3)	04.2% (4)	05.2% (4)	05.4% (4)	05.3% (5)
		820	120	100	28	94	181	297
[6]	infoleak	01.3%	00.9% (20)	01.2% (12)	01.1% (11)	01.1% (8)	01.3% (10)	01.6% (7)
		63	4	8	6	8	16	21
		04.2%	03.3% (6)	05.5% (4)	04.1% (5)	05.0% (5)	04.8% (5)	03.4% (6)
		583	33	81	27	90	159	193
[7]	dos-malform	03.7%	05.6% (3)	06.2% (2)	02.6% (4)	04.4% (4)	01.8% (7)	03.6% (4)
		182	25	41	14	33	22	47
		02.4%	04.4% (5)	04.7% (5)	02.4% (8)	02.9% (6)	01.8% (8)	01.7% (7)
		339	44	70	16	53	61	95
[8]	link	03.8%	07.4% (2)	03.3% (4)	04.2% (2)	05.1% (2)	04.2% (3)	01.5% (8)
		186	33	22	22	38	51	20
		01.1%	03.1% (7)	01.6% (14)	03.0% (6)	01.9% (9)	01.1% (12)	00.2% (24)

Table 3: OS Vendors vs. Others (continued)

		155	*31*	*23*	*20*	*34*	*36*	*11*
[9]	**format-string**	**02.3%**	05.2% (4)	01.5% (10)	02.3% (5)	02.8% (5)	02.4% (5)	01.5% (9)
		114	*23*	*10*	*12*	*21*	*29*	*19*
		01.5%	02.3% (9)	02.0% (11)	03.0% (7)	02.3% (7)	01.4% (9)	00.8% (12)
		203	*23*	*29*	*20*	*41*	*47*	*43*
[10]	**crypt**	**00.8%**	01.6% (14)	01.4% (11)	01.1% (9)	00.4% (18)	00.4% (16)	00.6% (15)
		38	*7*	*9*	*6*	*3*	*5*	*8*
		01.7%	04.9% (3)	03.3% (6)	01.8% (9)	01.1% (14)	01.9% (7)	00.8% (11)
		240	*48*	*49*	*12*	*19*	*64*	*48*
[11]	**priv**	**01.9%**	04.1% (5)	02.3% (6)	00.8% (16)	00.8% (15)	02.5% (4)	01.6% (6)
		95	*18*	*15*	*4*	*6*	*31*	*21*
		01.1%	01.8% (11)	02.1% (10)	01.4% (13)	01.5% (11)	01.1% (11)	00.6% (14)
		154	*18*	*31*	*9*	*27*	*36*	*33*
[12]	**perm**	**01.7%**	04.1% (6)	02.1% (8)	01.1% (8)	01.1% (9)	01.6% (8)	01.3% (11)
		83	*18*	*14*	*6*	*8*	*20*	*17*
		01.1%	02.1% (10)	01.7% (12)	01.4% (12)	00.9% (15)	00.8% (15)	01.0% (9)
		158	*21*	*25*	*9*	*16*	*28*	*59*
[13]	**metachar**	**01.2%**	02.0% (9)	02.6% (5)	00.8% (15)	00.7% (16)	01.2% (12)	00.8% (13)
		60	*9*	*17*	*4*	*5*	*15*	*10*
		01.2%	04.7% (4)	02.6% (7)	00.6% (19)	01.2% (13)	01.3% (10)	00.3% (19)
		173	*46*	*39*	*4*	*21*	*44*	*19*
[14]	**int-overflow**	**02.9%**	...	01.2% (13)	02.3% (6)	04.6% (3)	02.1% (6)	04.7% (2)
		140	*0*	*8*	*12*	*34*	*25*	*61*
		00.4%	00.1% (31)	...	00.6% (20)	00.7% (17)	00.3% (21)	00.4% (16)
		50	*1*	*0*	*4*	*13*	*11*	*21*
[15]	**auth**	**00.5%**	01.4% (15)	01.1% (14)	00.6% (17)	00.3% (20)	00.3% (22)	00.3% (19)
		26	*6*	*7*	*3*	*2*	*4*	*4*
		00.9%	01.6% (13)	01.4% (15)	00.5% (21)	00.8% (16)	00.5% (17)	01.0% (10)
		129	*16*	*20*	*3*	*15*	*17*	*58*
[16]	**dos-flood**	**00.6%**	02.5% (8)	00.6% (20)	00.2% (26)	00.3% (21)	00.2% (27)	00.6% (16)
		29	*11*	*4*	*1*	*2*	*3*	*8*
		00.8%	01.8% (12)	02.2% (9)	00.8% (17)	01.6% (10)	00.2% (23)	00.3% (20)
		109	*18*	*32*	*5*	*29*	*8*	*17*
[17]	**pass**	**00.5%**	00.2% (27)	00.8% (17)	00.2% (20)	00.8% (13)	00.3% (23)	00.5% (17)
		23	*1*	*5*	*1*	*6*	*4*	*6*
		00.8%	01.5% (16)	01.6% (13)	00.2% (26)	01.2% (12)	01.0% (14)	00.3% (18)
		112	*15*	*23*	*1*	*22*	*32*	*19*

Table 3: OS Vendors vs. Others (continued)

[18]	**webroot**	**00.0%**	...	...	...	...	...	00.2% (29)
		2	*0*	*0*	*0*	*0*	*0*	*2*
		00.8%	00.2% (29)	00.3% (26)	00.5% (22)	00.3% (25)	01.0% (13)	01.2% (8)
		115	*2*	*5*	*3*	*5*	*33*	*67*
[19]	**form-field**	**00.2%**	00.5% (22)	00.3% (26)	00.2% (23)	...	00.4% (18)	...
		10	*2*	*2*	*1*	*0*	*5*	*0*
		00.6%	00.8% (20)	01.0% (16)	00.8% (16)	00.3% (23)	00.4% (18)	00.7% (13)
		87	*8*	*15*	*5*	*6*	*14*	*39*
[20]	**relpath**	**00.6%**	01.6% (11)	00.3% (25)	00.4% (19)	01.1% (10)	00.2% (25)	00.7% (14)
		31	*7*	*2*	*2*	*8*	*3*	*9*
		00.4%	00.5% (24)	00.3% (25)	01.2% (14)	00.3% (19)	00.4% (20)	00.2% (22)
		50	*5*	*5*	*8*	*6*	*12*	*14*
[21]	**race**	**01.1%**	01.1% (17)	00.9% (15)	00.4% (18)	00.9% (11)	01.6% (9)	01.1% (12)
		53	*5*	*6*	*2*	*7*	*19*	*14*
		00.2%	00.2% (28)	00.1% (31)	00.9% (15)	00.2% (32)	00.2% (26)	00.1% (27)
		28	*2*	*2*	*6*	*3*	*7*	*8*
[22]	**memleak**	**00.8%**	02.0% (10)	00.6% (18)	00.8% (14)	00.9% (12)	00.9% (13)	00.2% (23)
		38	*9*	*4*	*4*	*7*	*11*	*3*
		00.2%	00.7% (21)	00.1% (33)	00.2% (30)	00.3% (22)	00.1% (29)	00.2% (26)
		28	*7*	*1*	*1*	*6*	*4*	*9*
[23]	**crlf**	**00.3%**	...	00.5% (24)	00.2% (27)	...	00.4% (17)	00.3% (18)
		13	*0*	*3*	*1*	*0*	*5*	*4*
		00.4%	...	00.1% (29)	00.2% (27)	00.7% (18)	00.4% (19)	00.4% (17)
		49	*0*	*2*	*1*	*13*	*13*	*20*
[24]	**msdos-device**	**00.0%**	...	...	00.2% (21)	...	...	...
		1	*0*	*0*	*1*	*0*	*0*	*0*
		00.4%	01.5% (14)	00.9% (17)	01.5% (11)	00.3% (20)	00.3% (22)	00.0% (32)
		56	*15*	*13*	*10*	*6*	*10*	*2*
[25]	**upload**	**00.0%**	...	...	...	...	00.1% (32)	...
		1	*0*	*0*	*0*	*0*	*1*	*0*
		00.4%	...	00.1% (35)	00.2% (29)	00.3% (21)	00.7% (16)	00.4% (15)
		53	*0*	*1*	*1*	*6*	*22*	*23*
[26]	**rand**	**00.3%**	01.4% (16)	00.5% (23)	00.2% (28)	00.1% (22)	...	00.2% (30)
		13	*6*	*3*	*1*	*1*	*0*	*2*
		00.3%	01.1% (18)	00.6% (19)	00.3% (23)	00.2% (30)	00.1% (31)	00.2% (23)
		39	*11*	*9*	*2*	*3*	*2*	*12*
[27]	**spoof**	**00.2%**	00.2% (23)	00.3% (28)	...	...	00.3% (24)	00.3% (21)

Table 3: OS Vendors vs. Others (continued)

		11	*1*	*2*	*0*	*0*	*4*	*4*
		00.3%	01.4% (17)	00.3% (27)	00.2% (31)	00.2% (29)	00.2% (25)	00.2% (25)
		40	*14*	*5*	*1*	*3*	*7*	*10*
[28]	**sandbox**	**00.7%**	02.7% (7)	02.1% (7)	...	00.1% (24)	00.2% (28)	00.2% (24)
		32	*12*	*14*	*0*	*1*	*2*	*3*
		00.1%	00.5% (26)	00.5% (21)	...	00.2% (28)	...	00.0% (36)
		18	*5*	*8*	*0*	*4*	*0*	*1*
[29]	**default**	**00.2%**	00.2% (24)	00.5% (22)	...	00.1% (23)	00.2% (26)	00.2% (26)
		10	*1*	*3*	*0*	*1*	*3*	*2*
		00.3%	01.5% (15)	00.9% (18)	00.2% (25)	00.3% (27)	...	00.1% (29)
		39	*15*	*13*	*1*	*5*	*0*	*5*
[30]	**signedness**	**00.5%**	00.2% (28)	00.9% (16)	00.9% (12)	00.4% (19)	00.6% (15)	00.2% (25)
		25	*1*	*6*	*5*	*3*	*7*	*3*
		00.1%	...	00.1% (30)	00.6% (18)	00.2% (31)	00.1% (28)	00.1% (31)
		17	*0*	*2*	*4*	*3*	*5*	*3*
[31]	**CF**	**00.2%**	00.9% (19)	00.2% (31)	00.2% (25)	...	...	00.2% (22)
		9	*4*	*1*	*1*	*0*	*0*	*3*
		00.2%	00.6% (23)	00.4% (23)	00.2% (24)	...	00.1% (30)	00.1% (28)
		24	*6*	*6*	*1*	*0*	*4*	*7*
[32]	**eval-inject**	**00.1%**	...	...	...	...	00.3% (20)	00.2% (27)
		6	*0*	*0*	*0*	*0*	*4*	*2*
		00.2%	...	00.1% (34)	...	00.1% (34)	00.2% (24)	00.3% (21)
		25	*0*	*1*	*0*	*1*	*7*	*16*
[33]	**dos-release**	**00.1%**	00.9% (18)	00.3% (27)	00.2% (24)	...	...	...
		7	*4*	*2*	*1*	*0*	*0*	*0*
		00.2%	00.9% (19)	00.5% (20)	00.2% (28)	00.3% (26)	...	...
		23	*9*	*8*	*1*	*5*	*0*	*0*
[34]	**design**	**00.1%**	00.2% (26)	00.3% (30)	00.2% (22)	...	00.1% (29)	...
		5	*1*	*2*	*1*	*0*	*1*	*0*
		00.1%	00.7% (22)	00.4% (24)	...	00.1% (33)	00.1% (32)	00.0% (33)
		18	*7*	*6*	*0*	*1*	*2*	*2*
[35]	**double-free**	**00.4%**	...	00.3% (29)	00.8% (13)	00.8% (14)	00.3% (21)	00.3% (20)
		20	*0*	*2*	*4*	*6*	*4*	*4*
		00.0%	...	...	...	...	00.0% (33)	00.0% (34)
		2	*0*	*0*	*0*	*0*	*1*	*1*
[36]	**CSRF**	**00.0%**	...	...	...	...	00.1% (30)	00.1% (31)
		2	*0*	*0*	*0*	*0*	*1*	*1*

Table 3: OS Vendors vs. Others (concluded)

		00.1%	...	00.1% (32)	...	00.3% (24)	00.2% (27)	00.1% (30)
		17	*0*	*1*	*0*	*5*	*7*	*4*
[37]	**type-check**	**00.2%**	00.7% (21)	00.6% (19)	...	...	00.1% (31)	...
		8	*3*	*4*	*0*	*0*	*1*	*0*
		00.1%	00.3% (27)	00.2% (28)	...	...	...	00.0% (35)
		7	*3*	*3*	*0*	*0*	*0*	*1*
UNKNOWN/UNSPECIFIED ITEMS								
n/a	**unk**	**16.9%**	12.4%	12.5%	10.6%	12.3%	16.1%	26.8%
		829	*55*	*83*	*56*	*92*	*196*	*347*
		06.6%	05.9%	04.6%	04.1%	06.6%	06.3%	07.7%
		914	*58*	*68*	*27*	*119*	*209*	*433*
n/a	**other**	**17.3%**	15.3%	15.8%	12.1%	12.1%	14.9%	26.2%
		847	*68*	*105*	*64*	*90*	*181*	*339*
		14.2%	17.3%	20.4%	11.8%	19.3%	12.7%	11.6%
		1979	*171*	*301*	*78*	*347*	*424*	*658*
n/a	**undiag**	**00.0%**	00.0%	00.0%	00.0%	00.0%	00.1%	00.0%
		1	*0*	*0*	*0*	*0*	*1*	*0*
		00.0%	00.0%	00.0%	00.0%	00.0%	00.0%	00.0%
		0	*0*	*0*	*0*	*0*	*0*	*0*
n/a	**not-specified**	**12.9%**	00.2%	05.9%	25.7%	24.6%	21.9%	00.5%
		632	*1*	*39*	*136*	*183*	*266*	*7*
		03.4%	00.1%	01.4%	15.2%	05.6%	07.2%	00.1%
		468	*1*	*20*	*100*	*100*	*240*	*7*

Top 5/10 Diversity Percentages per year

For the 'top N' vulnerabilities in each year, the table identifies the total percentage of overall vulnerabilities. For example, a figure of 45.0 for Top 5 says that the Top 5 accounted for 45% of all reported vulnerabilities in that year. This provides a rough estimate of how diverse the reported vulnerabilities were.

Top n	TOTAL	2001	2002	2003	2004	2005	2006
5	33.4	43.3	43.3	36.8	37.3	29.3	30.2
	53.3	44.9	45.2	44.7	45.9	55.3	65.3
10	42	56.6	52.8	43.5	43.7	37.7	37.8
	64.3	58.1	57.7	56.7	56.6	64.5	73.6

Table 4: Open and Closed Source (OS vendors)

Rank	Flaw	TOTAL	2001	2002	2003	2004	2005	2006
Total	Open	*raw numbers omitted*						
	Closed	*raw numbers omitted*						
[1]	**buf**	**19.7%**	20.1% (1)	24.6% (1)	25.0% (1)	25.3% (1)	14.6% (1)	17.2% (1)
		19.6%	20.2% (1)	27.6% (1)	25.7% (1)	15.0% (1)	18.5% (1)	15.7% (1)
[2]	**link**	**06.3%**	14.2% (2)	04.9% (3)	04.9% (2)	08.7% (2)	06.4% (2)	02.5% (6)
		01.4%	01.0% (18)	01.8% (9)	03.0% (2)	01.9% (6)	00.8% (8)	01.2% (9)
[3]	**dos-malform**	**03.2%**	02.7% (7)	04.5% (4)	02.6% (6)	03.5% (5)	01.7% (8)	05.1% (3)
		04.8%	09.1% (2)	08.1% (2)	02.5% (3)	07.1% (2)	02.1% (4)	03.1% (4)
[4]	**XSS**	**04.6%**	02.7% (8)	06.0% (2)	03.0% (5)	01.7% (7)	05.5% (3)	06.2% (2)
		02.3%	00.5% (22)	03.6% (4)	02.5% (4)	00.8% (11)	02.1% (3)	03.1% (3)
[5]	**format-string**	**04.1%**	08.7% (3)	03.0% (5)	03.0% (4)	05.2% (3)	03.8% (4)	03.0% (5)
		00.8%	01.9% (9)	00.6% (18)	02.0% (5)	00.8% (8)	00.8% (11)	00.2% (19)
[6]	**int-overflow**	**02.7%**	...	02.2% (7)	03.4% (3)	04.1% (4)	02.2% (6)	03.4% (4)
		01.9%	...	...	01.0% (9)	03.4% (3)	00.8% (10)	04.1% (2)
[7]	**priv**	**02.2%**	05.5% (5)	01.9% (10)	01.5% (12)	01.2% (12)	02.3% (5)	01.9% (7)
		01.7%	01.9% (10)	03.0% (6)	...	00.4% (13)	03.3% (2)	01.2% (10)
[8]	**perm**	**02.1%**	05.5% (4)	01.1% (14)	01.9% (8)	01.7% (8)	02.2% (7)	01.3% (11)
		01.6%	02.4% (6)	03.3% (5)	00.5% (12)	00.4% (14)	00.8% (13)	01.8% (7)
[9]	**dot**	**01.5%**	00.5% (17)	01.9% (8)	01.1% (16)	02.0% (6)	01.5% (11)	01.7% (8)
		01.1%	01.9% (8)	00.9% (13)	01.0% (8)	01.5% (7)	00.5% (17)	01.3% (8)
[10]	**infoleak**	**01.1%**	00.5% (20)	01.5% (12)	01.1% (15)	01.2% (11)	01.0% (14)	01.3% (12)
		01.4%	01.4% (15)	00.9% (12)	01.0% (7)	00.4% (18)	01.5% (5)	02.1% (6)
[11]	**metachar**	**01.7%**	03.2% (6)	03.0% (6)	01.5% (11)	00.6% (15)	01.5% (12)	01.7% (9)
		00.5%	01.0% (17)	01.2% (11)	...	00.4% (17)	00.5% (15)	00.2% (23)
[12]	**sql-inject**	**01.2%**	00.5% (19)	00.7% (15)	01.9% (9)	00.9% (14)	01.6% (9)	01.3% (13)
		00.9%	...	00.6% (16)	00.5% (18)	00.4% (15)	...	02.5% (5)
[13]	**race**	**01.5%**	02.3% (9)	01.9% (11)	00.4% (18)	01.5% (10)	01.5% (10)	01.7% (10)
		00.3%	...	...	00.5% (21)	...	00.8% (14)	00.5% (17)
[14]	**memleak**	**00.9%**	...	00.7% (19)	01.1% (14)	01.5% (9)	01.2% (13)	00.4% (19)
		00.7%	04.3% (4)	00.3% (21)	00.5% (15)	...	00.8% (9)	00.2% (20)
[15]	**crypt**	**00.7%**	01.8% (10)	00.7% (17)	01.9% (7)	...	00.4% (21)	00.4% (16)
		00.8%	01.0% (16)	02.1% (8)	...	00.8% (10)	00.5% (16)	00.7% (14)
[16]	**sandbox**	**00.2%**	00.5% (15)	...	...	00.3% (19)	00.3% (25)	...

Table 4: Open and Closed Source (OS vendors) (continued)

		01.3%	05.3% (3)	04.2% (3)	...	...	...	00.2% (24)
[17]	relpath	00.6%	01.8% (11)	00.7% (18)	00.4% (19)	00.3% (18)	00.4% (22)	00.4% (20)
		00.8%	01.4% (14)	...	00.5% (16)	01.9% (5)	...	01.2% (11)
[18]	dos-flood	00.3%	01.4% (12)	00.4% (23)	...	...	00.3% (23)	...
		01.0%	03.8% (5)	00.9% (15)	00.5% (17)	00.8% (12)	00.3% (21)	01.0% (12)
[19]	auth	00.1%	00.5% (16)	...	...	00.3% (17)	...	00.2% (23)
		01.1%	02.4% (7)	02.1% (7)	01.5% (6)	00.4% (16)	01.0% (6)	00.5% (16)
[20]	pass	00.0%	...	...	...	...	00.1% (27)	...
		01.0%	00.5% (20)	01.5% (10)	00.5% (13)	02.3% (4)	00.8% (7)	00.8% (13)
[21]	signedness	00.8%	...	01.9% (9)	01.5% (10)	00.3% (20)	00.6% (16)	00.6% (14)
		00.1%	...	...	00.5% (19)	...	00.5% (18)	...
[22]	double-free	00.5%	...	00.4% (22)	01.1% (13)	00.9% (13)	00.3% (24)	00.4% (18)
		00.2%	...	...	00.5% (20)	00.8% (9)	00.3% (19)	...
[23]	spoof	00.2%	...	00.7% (20)	...	...	00.4% (18)	...
		00.3%	00.5% (23)	...	...	...	00.3% (20)	00.7% (15)
[24]	rand	00.2%	00.9% (14)	00.4% (21)	00.4% (17)	...	...	00.2% (24)
		00.2%	01.9% (11)	00.3% (22)	...	...	...	...
[25]	crlf	00.4%	...	01.1% (13)	...	...	00.6% (17)	00.6% (15)
		00.0%	...	...	...	...	...	...
[26]	form-field	00.4%	00.5% (18)	00.7% (16)	00.4% (20)	...	00.7% (15)	...
		00.0%	00.5% (25)	...	...	...	...	...
[27]	default	00.1%	...	...	...	00.3% (16)	...	00.2% (22)
		00.3%	00.5% (24)	00.6% (17)	...	...	00.8% (12)	00.2% (21)
[28]	CF	00.1%	00.9% (13)	...	...	...	...	...
		00.3%	01.0% (19)	...	00.5% (11)	...	...	00.5% (18)
[29]	type-check	00.0%	...	...	...	...	00.1% (28)	...
		00.3%	01.4% (12)	00.9% (14)	...	...	...	...
[30]	dos-release	00.1%	00.5% (21)	00.4% (24)	00.4% (21)	...	...	...
		00.2%	01.4% (13)	00.3% (20)	...	...	...	...
[31]	eval-inject	00.2%	...	...	...	...	00.4% (20)	00.4% (17)
		00.0%	...	...	...	...	...	...
[32]	design	00.0%	...	...	...	...	...	...
		00.2%	00.5% (21)	00.3% (19)	00.5% (14)	...	00.3% (22)	...
[33]	php-include	00.2%	...	...	...	...	00.4% (19)	00.2% (26)
		00.0%	...	...	...	...	...	...
[34]	CSRF	00.1%	...	...	...	...	00.1% (26)	00.2% (21)
		00.0%	...	...	...	...	...	...

Table 4: Open and Closed Source (OS vendors) (concluded)

[35]	**webroot**	**00.0%**	...	...	...	...	...	00.2% (25)
		00.0%	...	...	...	...	...	00.2% (22)
[36]	**msdos-device**	**00.0%**	...	...	...	...	...	...
		00.0%	...	...	00.5% (10)	...	...	...
[37]	**upload**	**00.0%**	...	...	...	...	00.1% (29)	...
		00.0%	...	...	...	...	...	...
UNKNOWN/UNSPECIFIED ITEMS								
n/a	**unk**	**09.3%**	12.3%	10.1%	04.5%	07.8%	11.8%	07.9%
		27.4%	13.0%	15.6%	20.3%	23.3%	26.9%	43.3%
n/a	**other**	**19.9%**	12.8%	20.5%	15.7%	10.5%	15.3%	38.9%
		12.9%	18.8%	12.9%	05.4%	14.7%	11.8%	13.4%
n/a	**undiag**	**00.0%**	00.0%	00.0%	00.0%	00.0%	00.1%	00.0%
		00.0%	00.0%	00.0%	00.0%	00.0%	00.0%	00.0%
n/a	**not-specified**	**12.5%**	00.0%	04.1%	21.6%	20.3%	20.4%	00.4%
		11.7%	00.5%	06.3%	28.2%	22.9%	23.6%	00.3%

For the 'top N' vulnerabilities in each year, the table identifies the total percentage of overall vulnerabilities. For example, a figure of 45.0 for Top 5 says that the Top 5 accounted for 45% of all reported vulnerabilities in that year. This provides a rough estimate of how diverse the reported vulnerabilities were.

Top n	TOTAL	2001	2002	2003	2004	2005	2006
5	37.9	54	43	39.3	46.8	32.6	34.9
	30.3	42.7	46.8	35.7	29.7	27.5	28.5
10	48.1	66.7	53.9	49.1	55.2	41.8	44.4
	37.1	53.2	57.3	40.7	35.5	31.7	36.1