DOVER
MICROSYSTEMS

CoreGuard® Cybersecurity Scorecard

Measuring the Effectiveness of CoreGuard's Cybersecurity Defense Mechanisms

July 2021 | VERSION 4.0

COREGUARD®
Protected. Trusted.

This scorecard uses widespread and well-known security standards to measure how effectively the CoreGuard solution from Dover Microsystems prevents the exploitation of software vulnerabilities and immunizes processors against entire classes of network-based attacks.

COREGUARD PROTECTS A SYSTEM AT ITS CORE

Today's processors blindly execute instructions, and do not have the knowledge to distinguish between good and bad. Compounding this issue is the fact that all software contains bugs, and attackers find and exploit these vulnerabilities. With an unprecedented approach to cybersecurity, CoreGuard is the only solution for embedded systems that addresses both facets of this problem.

CoreGuard is silicon IP that integrates with RISC processors to protect embedded systems from cyberattacks by enforcing security, safety, and privacy rules—called **micropolicies**—that precisely define allowed versus disallowed behavior. CoreGuard maintains micropolicy-relevant **metadata** about every word in memory, and then uses this metadata to crosscheck each instruction processed against the installed set of micropolicies. If an instruction violates any micropolicy, CoreGuard **Policy Enforcer** hardware stops it from executing before any damage is done.

With micropolicies that define valid and invalid behavior for the processor, CoreGuard is able to block entire classes of attacks—not just specific exploits. Because of this, CoreGuard is future proof, and can even stop zero-day attacks that take advantage of software vulnerabilities yet to be discovered.

THE EXPANDING UNIVERSE OF SOFTWARE VULNERABILITIES

According to Steve McConnell in his book, *Code Complete*, there are on average 15-50 bugs per thousand lines of delivered code. To give that statistic some context, consider this: the Android operating system has 15 million lines of source code, Windows 7 has 40 million, and a Ford F-150 has 150 million lines of code. Cybersecurity Ventures estimates that there are 111 billion lines of new software code produced each year. This makes for a huge, and ever-expanding universe of vulnerabilities. Furthermore, the FBI has found that at least two percent of all these vulnerabilities can be exploited by cyberattackers. That's the bad news.

The good news is that there's a vast amount of publicly available and frequently updated data on vulnerabilities that informs developers and security practitioners about where we are most exposed.

The even better news is that Dover has figured out how to use this data to write micropolicies that not only protect against documented classes of vulnerabilities but are designed to protect against future vulnerabilities.

NAVIGATING THE UNIVERSE

To sort through the mammoth universe of vulnerabilities, Dover uses two broadly-accepted databases: CVE (Common Vulnerabilities and Exposures) and CWE (Common Weakness Enumeration). Both are maintained by The MITRE Corporation and sponsored by the U.S. Department of Homeland Security (DHS) Cybersecurity and Infrastructure Security Agency (CISA).

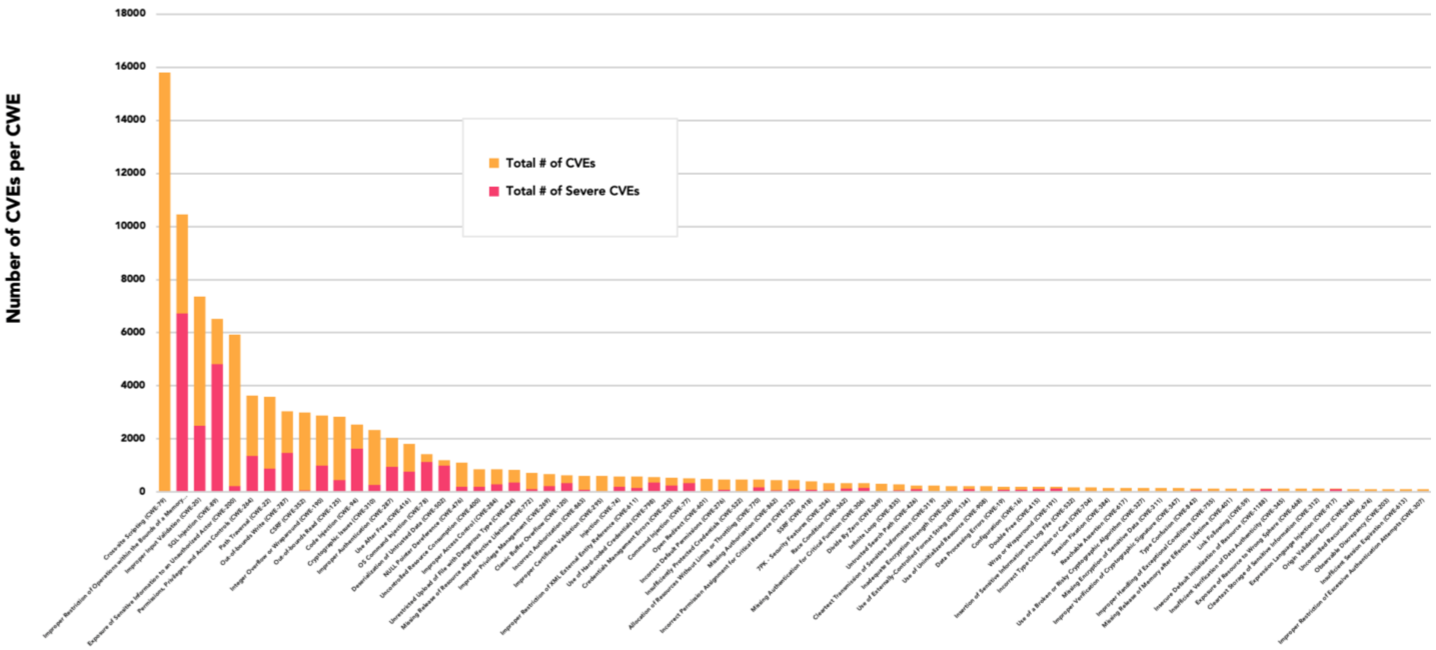
CVE is an open standard that provides globally unique identifiers for known cybersecurity vulnerabilities in software. It was started by MITRE in 1999 to address the confusion around different individuals and organizations using different names to talk about the same vulnerabilities. CVEs are assigned by CVE Numbering Authorities from around the world, with each having an identification number, a description, and at least one reference to an instance of that vulnerability in a specific piece of software.

While building CVE, MITRE developed a preliminary classification of vulnerabilities, attacks, and faults. This work evolved into the **CWE** list, which was first published in 2006. Where CVE lists specific **instances** of vulnerabilities, CWE defines **categories** of software weaknesses. A software weakness is an error that can lead to a software vulnerability. Software weaknesses include, for example, buffer overflows, code evaluation and injection, and insufficient verification of data.

MITRE created CWE with code assessment in mind. It "serves as a common language for describing software security weaknesses, a standard measuring stick for software security tools targeting these vulnerabilities, and a baseline standard for weakness identification, mitigation, and prevention efforts." CWE is also used to organize the massive list of CVEs. For example, thousands of individual vulnerabilities due to buffer errors map to one CWE category: (CWE-119) Improper Restriction of Operations within the Bounds of a Memory Buffer.

For the most reliable and actionable data, Dover uses the subset of CVEs that are assigned to CWEs.[1] As you can see in Figure 1, this subset includes nearly 99,000 vulnerabilities specific to network-based attacks grouped across 243 categories (CWEs). The distribution clearly shows which types of vulnerabilities are most prevalent, with the CWE-79 category (Cross-site Scripting) topping the chart at over 15,000 associated CVEs.

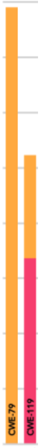
FIGURE 1: Number of Vulnerabilities in Each CWE Class[2,3]



[1] The CVE list was started in 1999, while the CWE and the formality of CVE Numbering Authorities did not begin until around 2006. Many of the earlier identifications are considered incomplete, not reproducible, or of general low quality. Consequently, about half of the CVEs do not map to a CWE.

[2] Charts in this document use CVE data downloaded July 2, 2021 and CWE data from Version 4.5 of the CWE list.

[3] For ease of visibility, charts in this document only reflects CWEs that have at least 100 identified CVEs.



Not all CVEs are created equal. When assessing security risk, it is important to look not only at the number of CVEs in a category, but also their severity.

The National Vulnerability Database (NVD) uses the CVSS (Common Vulnerability Scoring System) to assess the severity of CVEs using several metrics that approximate the ease and impact of an exploit. Scores range from 0 to 10, with 10 being the most severe.

In Figure 1 on the previous page, pink highlights the number of severe CVEs (those with a score > 7.0) in each category. Categories with more pink pose a greater threat. For example, consider the two categories: **CWE-79** (Cross-site Scripting) and **CWE-119** (Improper Restriction of Operations within the Bounds of a Memory Buffer). Both categories have a high volume of vulnerabilities, but in the CWE-119 category, almost two-thirds of the CVEs are severe, while in the CWE-79 category, you can hardly see any pink on the bar because the number is so small (only 35 severe vulnerabilities). This means stopping CWE-119 is more important and has a bigger impact than stopping CWE-79.

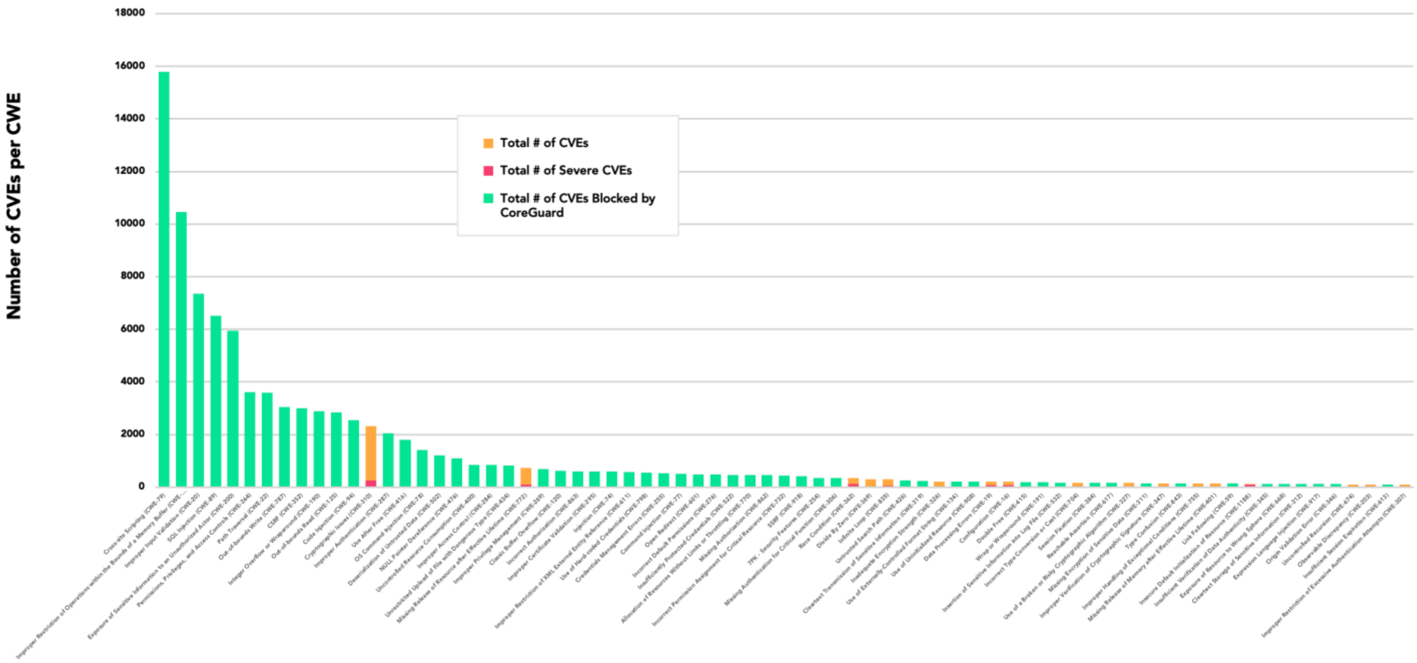
CVE Severity Scores (CVSS)

SEVERITY	SCORE	
None	0.0	
Low	0.1 – 3.9	
Med	4.0 – 6.9	
High	7.0 – 8.9	SEVERE
Critical	9.0 – 10.0	

IMMUNIZING PROCESSORS WITH COREGUARD

Figure 2 looks again at the universe of nearly 99,000 CVEs. Green highlights the categories of weaknesses—the CWEs—that CoreGuard micropolicies protect against. CoreGuard's full suite of micropolicies (including those available now and in upcoming releases) **immunize processors against 93% of all CVEs**. CVEs, of course, represent only those vulnerabilities we know about, which is just a fraction of all bugs out in the wild. Fortunately, CoreGuard micropolicies block entire classes of attack—not just specific attacks—to protect against both known and unknown vulnerabilities. We'll take a closer look at how micropolicies do this in the next section.

FIGURE 2: CoreGuard Protects Against Categories of Vulnerabilities

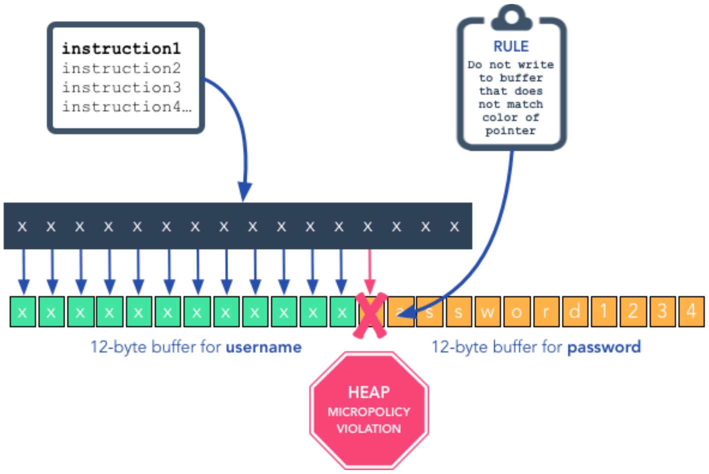


MICROPOLICIES TARGET CLASSES OF ATTACK

Equipped with a clear understanding of the common categories of software weaknesses, Dover has written CoreGuard micropolicies to protect against the classes of attack that exploit these weaknesses. Micropolicies translate the English definitions of CWEs into precise executable rules that define valid and invalid system behavior. In other words, **micropolicies codify CWEs**.

Take CWE-119 again (the second bar in Figure 2). It includes all classic buffer overflow vulnerabilities that allow an operation to read from or write to a memory location outside of the intended boundary of the buffer. CoreGuard Heap and Stack micropolicies (two of the micropolicies in CoreGuard's base set) are designed to block buffer overflow attacks that exploit bugs in the CWE-119 category.

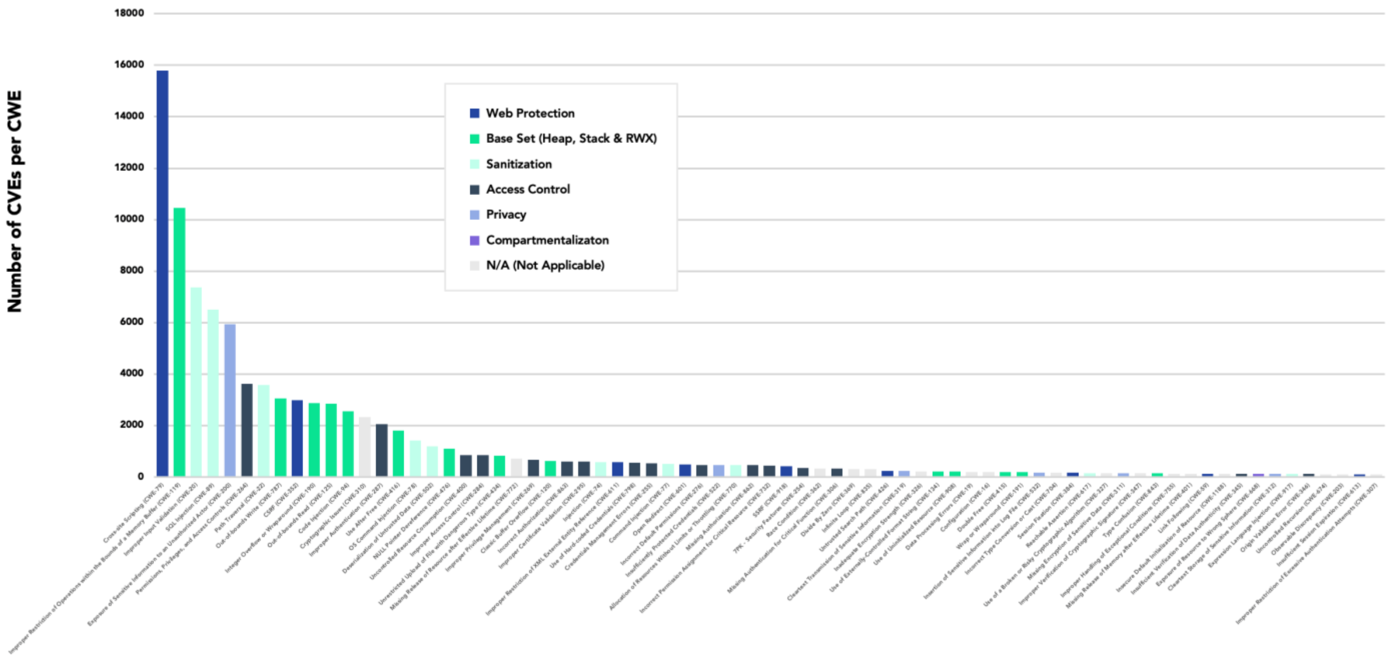
The Heap micropolicy, for example, protects memory on the heap by assigning a color to the buffer in which data resides, as well as to the pointer to the buffer. These color assignments are stored as metadata. Heap micropolicy rules dictate that an instruction cannot write data to a buffer with a color that doesn't match the color of the pointer to the buffer. In the example below, the application successfully writes the first 12 bytes of data to the green (username) buffer, but CoreGuard blocks it from overrunning that buffer to write any data to the adjacent yellow (password) buffer.


instruction1
instruction2
instruction3
instruction4…
RULE
Do not write to buffer that does not match color of pointer
x x x x x x x x x x x x x x x x
x x x x x x x x x x x x
a s s w o r d 1 2 3 4
12-byte buffer for **username**
12-byte buffer for **password**
HEAP
MICROPOLICY
VIOLATION

Today's processors may not be able to distinguish between good and bad instructions, but CoreGuard micropolicies can. Furthermore, because micropolicies are built to protect against entire categories of weaknesses and not just specific vulnerabilities, when a new buffer overflow vulnerability is discovered, the CoreGuard Heap and Stack micropolicies are already equipped to protect against it—no update required.

Figure 3 on the next page shows which micropolicies block the CVEs in each CWE category. For a description of each of these micropolicies, see Table 1.

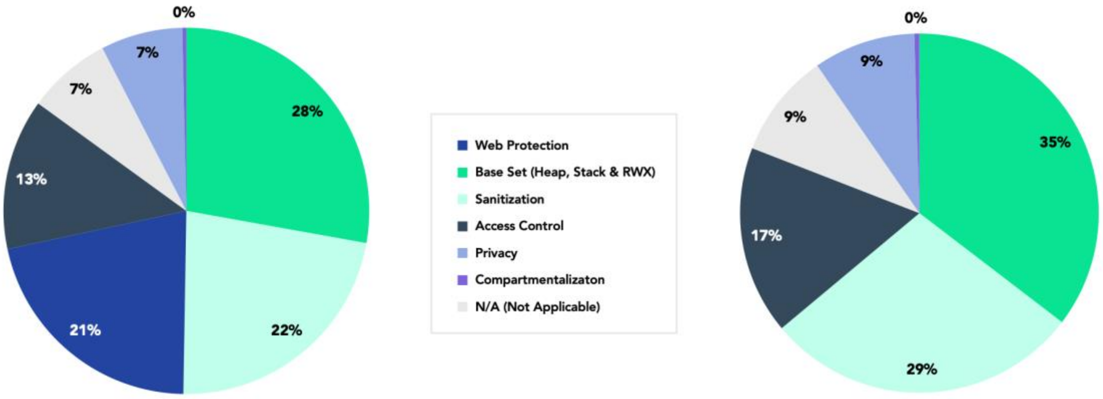
FIGURE 3: Protection Breakdown by Micropolicy



When looking at the data in Figure 3, it is important to think about which vulnerabilities are relevant to your application and threat vectors. The CWE groupings make it easier to do this. For example, if your application does not have a web front end, then you don't need to worry about vulnerabilities related to Web Protection.

Figure 4 shows two pie charts. 4A illustrates the percentage of all vulnerabilities that CoreGuard protects against with each micropolicy. 4B shows how the pie changes for an embedded system with no web front-end: the dark blue slice of the pie goes away, and each remaining micropolicy covers a larger percentage.

FIGURE 4: Micropolicy Breakdown Based on Use Case



A: Micropolicy Breakdown for Embedded Systems Where All CVEs Are Applicable

B: Micropolicy Breakdown for Embedded Systems Where Web Protection CVEs Are Not Applicable

The following table describes the micropolicies listed in Figures 3 and 4. The asterisk (*) identifies micropolicies that will be available in upcoming releases.

TABLE 1: CoreGuard Micropolicies

MICROPOLICIES	DESCRIPTION
Base Set (Heap, Stack & RWX) BLOCKS **27.82%** OF ALL CVES	The **Stack** micropolicy protects frames on the stack, including the return address, in order to increase control flow integrity. The **Heap** micropolicy prevents buffer overflows within heap memory by assigning a color to the buffer in which data resides, as well as to the pointer to the buffer, and then blocking any instruction from writing data to a buffer with a color that doesn't match the color of the pointer. The **RWX** micropolicy labels each word in memory with metadata that indicates whether it is readable, writable, and/or executable. RWX is designed to stop attacks such as code injection or modification of control data (e.g., virtual function tables).
Web Protection* BLOCKS **21.38%** OF ALL CVES	**Web Protection** micropolicies block attacks of web servers running on embedded systems by guaranteeing that carefully-crafted routines for processing scripts, HTML pages, XML fragments, URLs, and more are called prior to any change to the web server data.
Sanitization* BLOCKS **22.44%** OF ALL CVES	**Sanitization** micropolicies ensure that when data comes from the outside world into a system, that the application's data sanitization routine is called and completed immediately before any function using that data gets called.

MICROPOLICIES	DESCRIPTION
Confidentiality BLOCKS **7.23%** OF ALL CVES	The **Confidentiality** micropolicy prevents the exfiltration of private data by labeling confidential data as "private," propagating that label as data flows through the system and ensuring that "private" data never leaves the device unless it is encrypted.
Compartmentalization BLOCKS **0.35%** OF ALL CVES	**Compartmentalization** micropolicies partition an application's memory into program-defined "compartments," and define rules for how data inside a compartment can interact with data outside a compartment.
Access Control* BLOCKS **13.33%** OF ALL CVES	**Access Control** micropolicies provide fine-grained control over access to data. This can include who has access to the data, as well as what they are allowed to do with it.

COREGUARD BLOCKS TODAY'S AND TOMORROW'S THREATS

Nearly every cyberattack relies on exploiting a software vulnerability. With an average of over 300 million new lines of code produced per day, the universe of software vulnerabilities is a constantly expanding treasure trove for attackers. Fortunately, the CWE list organizes this massive dataset into categories that give security practitioners a logical framework for assessing vulnerabilities and developing defense mechanisms to mitigate risk.

In this paper, we described how CoreGuard's defense mechanisms—its micropolicies—protect against classes of attack that prey on categories of weaknesses. CoreGuard's base set of micropolicies alone can block classic buffer overflow attacks that exploit the most prevalent category of software vulnerabilities. Furthermore, because micropolicies protect against types of vulnerabilities rather than specific vulnerabilities, it doesn't matter how many new bugs are discovered within a category. The micropolicy will block the entire class of attack, whether it is associated with two vulnerabilities or two million.

Between the micropolicies available today and those on the roadmap, CoreGuard can immunize processors in our embedded systems against 93% of common vulnerabilities. Plus, if a new class of attack is discovered or you want to add additional micropolicies to your system for defense-in-depth, CoreGuard micropolicies can be updated on deployed devices without requiring any changes to CoreGuard hardware IP.

As we explain in The Cybersecurity Stack white paper, CoreGuard is a critical component of a defense-in-depth approach to securing embedded systems—and even CoreGuard itself offers layers of security with unique micropolicies that can be composed in different combinations to provide the most effective solution for a particular use case. CoreGuard delivers unparalleled cybersecurity that is effective today, and effective tomorrow.

APPENDIX

TABLE 2: All CWEs with CVEs > 0

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
Base Set (Heap, Stack & RWX)	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.	6,728	10,466	10.58%	21.36%
	787	Out-of-bounds Write	The software writes data past the end, or before the beginning, of the intended buffer.	1,467	3,052	3.09%	4.66%
	190	Integer Overflow or Wraparound	The software performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value. This can introduce other weaknesses when the calculation is used for resource management or execution control.	990	2,882	2.91%	3.14%
	125	Out-of-bounds Read	The software reads data past the end, or before the beginning, of the intended buffer.	446	2,843	2.87%	1.42%
	94	Code Injection	The software constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment.	1,640	2,549	2.58%	5.21%
	416	Use After Free	Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.	757	1,804	1.82%	2.40%
	476	NULL Pointer Dereference	NULL pointer dereference issues can occur through a number of flaws, including race conditions, and simple programming omissions.	201	1,108	1.12%	0.64%
	434	Unrestricted Upload of File with Dangerous Type	The software allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment.	366	825	0.83%	1.16%
	120	Classic Buffer Overflow	The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.	331	630	0.64%	1.05%
	134	Use of Externally-Controlled Format String	The software uses a function that accepts a format string as an argument, but the format string originates from an external source.	119	220	0.22%	0.38%
	908	Use of Uninitialized Resource	The software uses or accesses a resource that has not been initialized.	45	215	0.22%	0.14%
	415	Double Free	The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.	97	197	0.20%	0.31%
	191	Wrap or Wraparound	The product subtracts one value from another, such that the result is less than the minimum allowable integer value, which produces a value that is not equal to the correct result.	140	188	0.19%	0.44%
	843	Type Confusion	The program allocates or initializes a resource such as a pointer, object, or variable using one type, but it later accesses that resource using a type that is incompatible with the original type.	62	142	0.14%	0.20%
	129	Improper Validation of Array Index	The product uses untrusted input when calculating or using an array index, but the product does not validate or incorrectly validates the index to ensure the index references a valid position within the array.	45	92	0.09%	0.14%
	193	Off-by-one Error	A product calculates or uses an incorrect maximum or minimum value that is 1 more, or 1 less, than the correct value.	29	61	0.06%	0.09%
	121	Stack-based Buffer Overflow	A stack-based buffer overflow condition is a condition where the buffer being overwritten is allocated on the stack (i.e., is a local variable or, rarely, a parameter to a function).	42	54	0.05%	0.13%
	824	Access of Uninitialized Pointer	The program accesses or uses a pointer that has not been initialized.	11	50	0.05%	0.03%
	763	Release of Invalid Pointer or Reference	The application attempts to return a memory resource to the system, but calls the wrong release function or calls the appropriate release function incorrectly.	17	46	0.05%	0.05%
	122	Heap-based Buffer Overflow	A heap overflow condition is a buffer overflow, where the buffer that can be overwritten is allocated in the heap portion of memory, generally meaning that the buffer was allocated using a routine such as malloc().	9	39	0.04%	0.03%
	788	Access of Memory Location After End of Buffer	The software reads or writes to a buffer using an index or pointer that references a memory location after the end of the buffer.	6	12	0.01%	0.02%
	131	Incorrect Calculation of Buffer Size	The software does not correctly calculate the size to be used when allocating a buffer, which could lead to a buffer overflow.	7	10	0.01%	0.02%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
Base Set (Heap, Stack & RWX) Cont'd	672	Operation on a Resource after Expiration or Release	The software uses, accesses, or otherwise operates on a resource after that resource has been expired, released, or revoked.	4	8	0.01%	0.01%
	123	Write-what-where Condition	Any condition where the attacker has the ability to write an arbitrary value to an arbitrary location, often as the result of a buffer overflow.	2	5	0.01%	0.01%
	170	Improper Null Termination	The software does not terminate or incorrectly terminates a string or array with a null character or equivalent terminator.	1	4	0.00%	0.00%
	471	MAID	The software does not properly protect an assumed-immutable element from being modified by an attacker.	-	3	0.00%	0.00%
	823	Use of Out-of-range Pointer Offset	The program performs pointer arithmetic on a valid pointer, but it uses an offset that can point outside of the intended range of valid memory locations for the resulting pointer.	-	2	0.00%	0.00%
	126	Buffer Over-read	The software reads from a buffer using buffer access mechanisms such as indexes or pointers that reference memory locations after the targeted buffer.	1	1	0.00%	0.00%
	130	Improper Handling of Length Parameter Inconsistency	The software parses a formatted message or structure, but it does not handle or incorrectly handles a length field that is inconsistent with the actual length of the associated data.	-	1	0.00%	0.00%
	680	Integer Overflow to Buffer Overflow	The product performs a calculation to determine how much memory to allocate, but an integer overflow can occur that causes less memory to be allocated than expected, leading to a buffer overflow.	-	1	0.00%	0.00%
	805	Buffer Access with Incorrect Length Value	The software uses a sequential operation to read or write a buffer, but it uses an incorrect length value that causes it to access memory that is outside of the bounds of the buffer.	1	1	0.00%	0.00%
	1285	Improper Validation of Specified Index, Position, or Offset in Input	The product receives input that is expected to specify an index, position, or offset into an indexable resource such as a buffer or file, but it does not validate or incorrectly validates that the specified index/position/offset has the required properties.	-	1	0.00%	0.00%
TOTAL	32			13,564	27,512	27.82%	43.06%
Web Protection*	79	Cross-site Scripting	The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.	35	15,791	15.97%	0.11%
	352	CSRF	The web application does not, or can not, sufficiently verify whether a well-formed, valid, consistent request was intentionally provided by the user who submitted the request.	70	2,994	3.03%	0.22%
	611	Improper Restriction of XML External Entity Reference	The software processes an XML document that can contain XML entities with URIs that resolve to documents outside of the intended sphere of control, causing the product to embed incorrect documents into its output.	147	587	0.59%	0.47%
	601	Open Redirect	A web application accepts a user-controlled input that specifies a link to an external site, and uses that link in a Redirect. This simplifies phishing attacks.	1	486	0.49%	0.00%
	918	SSRF	The web server receives a URL or similar request from an upstream component and retrieves the contents of this URL, but it does not sufficiently ensure that the request is being sent to the expected destination.	86	415	0.42%	0.27%
	426	Untrusted Search Path	The application searches for critical resources using an externally-supplied search path that can point to resources that are not under the application's direct control.	114	249	0.25%	0.36%
	384	Session Fixation	Authenticating a user, or otherwise establishing a new user session, without invalidating any existing session identifier gives an attacker the opportunity to steal authenticated sessions.	28	163	0.16%	0.09%
	59	Link Following	The software attempts to access a file based on the filename, but it does not properly prevent that filename from identifying a link or shortcut that resolves to an unintended resource.	29	135	0.14%	0.09%
	613	Insufficient Session Expiration	Insufficient Session Expiration is when a web site permits an attacker to reuse old session credentials or session IDs for authorization	24	105	0.11%	0.08%
	444	HTTP Request Smuggling	When malformed or abnormal HTTP requests are interpreted by one or more entities in the data flow between the user and the web server, such as a proxy or firewall, they can be interpreted inconsistently, allowing the attacker to "smuggle" a request to one device without the other device being aware of it.	14	92	0.09%	0.04%
	425	Forced Browsing	The web application does not adequately enforce appropriate authorization on all restricted URLs, scripts, or files.	11	47	0.05%	0.03%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
Web Protection* Cont'd	93	CRLF Injection	The software uses CRLF (carriage return line feeds) as a special element, e.g. to separate lines or records, but it does not neutralize or incorrectly neutralizes CRLF sequences from inputs.	2	34	0.03%	0.01%
	113	HTTP Response Splitting	The software receives data from an upstream component, but does not neutralize or incorrectly neutralizes CR and LF characters before the data is included in outgoing HTTP headers.	1	21	0.02%	0.00%
	565	Reliance on Cookies without Validation and Integrity Checking	The application relies on the existence or values of cookies when performing security-critical operations, but it does not properly ensure that the setting is valid for the associated user.	4	20	0.02%	0.01%
	472	External Control of Assumed-Immutable Web Parameter	The web application does not sufficiently verify inputs that are assumed to be immutable but are actually externally controllable, such as hidden form fields.	7	7	0.01%	0.02%
	539	Use of Persistent Cookies Containing Sensitive Information	The web application uses persistent cookies, but the cookies contain sensitive information.	-	1	0.00%	0.00%
TOTAL	16			573	21,147	21.38%	1.82%
Sanitization*	20	Improper Input Validation	The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly.	2,501	7,366	7.45%	7.94%
	89	SQL Injection	The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.	4,827	6,514	6.59%	15.32%
	22	Path Traversal	The software uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the software does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory.	881	3,586	3.63%	2.80%
	78	OS Command Injection	The software constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.	1,134	1,426	1.44%	3.60%
	502	Deserialization of Untrusted Data	The application deserializes untrusted data without sufficiently verifying that the resulting data will be valid.	993	1,201	1.21%	3.15%
	74	Injection	The software constructs all or part of a command, data structure, or record using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify how it is parsed or interpreted when it is sent to a downstream component.	208	589	0.60%	0.66%
	77	Command Injection	The software constructs all or part of a command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended command when it is sent to a downstream component.	343	514	0.52%	1.09%
	770	Allocation of Resources Without Limits or Throttling	The software allocates a reusable resource or group of resources on behalf of an actor without imposing any restrictions on the size or number of resources that can be allocated, in violation of the intended security policy for that actor.	166	466	0.47%	0.53%
	617	Reachable Assertion	The product contains an assert() or similar statement that can be triggered by an attacker, which leads to an application exit or other behavior that is more severe than necessary.	15	161	0.16%	0.05%
	917	Expression Language Injection	The software constructs all or part of an expression language (EL) statement in a Java Server Page (JSP) using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended EL statement before it is executed.	118	123	0.12%	0.37%
	1236	Improper Neutralization of Formula Elements in a CSV File	The software saves user-provided information into a Comma-Separated Value (CSV) file, but it does not neutralize or incorrectly neutralizes special elements that could be interpreted as a command when the file is opened by spreadsheet software.	20	68	0.07%	0.06%
	88	Argument Injection	The software constructs a string for a command to executed by a separate component in another control sphere, but it does not properly delimit the intended arguments, options, or switches within that command string.	33	59	0.06%	0.10%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
Sanitization* Cont'd	91	XPath Injection	The software does not properly neutralize special elements that are used in XML, allowing attackers to modify the syntax, content, or commands of the XML before it is processed by an end system.	10	53	0.05%	0.03%
	470	Unsafe Reflection	The application uses external input with reflection to select which classes or code to use, but it does not sufficiently prevent the input from selecting improper classes or code.	12	16	0.02%	0.04%
	428	Unquoted Search Path or Element	The product uses a search path that contains an unquoted element, in which the element contains whitespace or other separators. This can cause the product to access resources in a parent path.	4	12	0.01%	0.01%
	80	Basic XSS	The software receives input from an upstream component, but it does not neutralize or incorrectly neutralizes special characters such as "<", ">", and "&" that could be interpreted as web-scripting elements when they are sent to a downstream component that processes web pages.	-	11	0.01%	0.00%
	90	LDAP Injection	The software constructs all or part of an LDAP query using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended LDAP query when it is sent to a downstream component.	3	10	0.01%	0.01%
	23	Relative Path Traversal	The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize sequences such as ".." that can resolve to a location that is outside of that directory.	1	7	0.01%	0.00%
	943	Improper Neutralization of Special Elements in Data Query Logic	The application generates a query intended to access or manipulate data in a data store such as a database, but it does not neutralize or incorrectly neutralizes special elements that can modify the intended logic of the query.	3	4	0.00%	0.01%
	707	Improper Neutralization	The product does not ensure or incorrectly ensures that structured messages or data are well-formed and that certain security properties are met before being read from an upstream component or sent to a downstream component.	-	3	0.00%	0.00%
	1286	Improper Validation of Syntactic Correctness of Input	The product receives input that is expected to be well-formed - i.e., to comply with a certain syntax - but it does not validate or incorrectly validates that the input complies with the syntax.	-	2	0.00%	0.00%
	87	Improper Neutralization of Alternate XSS Syntax	The software does not neutralize or incorrectly neutralizes user-controlled input for alternate script syntax.	-	1	0.00%	0.00%
	96	Static Code Injection	The software receives input from an upstream component, but it does not neutralize or incorrectly neutralizes code syntax before inserting the input into an executable resource, such as a library, configuration file, or template.	-	1	0.00%	0.00%
	117	Improper Output Neutralization for Logs	The software does not neutralize or incorrectly neutralizes output that is written to logs.	-	1	0.00%	0.00%
	178	Improper Handling of Case Sensitivity	The software does not properly account for differences in case sensitivity when accessing or determining the properties of a resource, leading to inconsistent results.	1	1	0.00%	0.00%
	644	Improper Neutralization of HTTP Headers for Scripting Syntax	The application does not neutralize or incorrectly neutralizes web scripting syntax in HTTP headers that can be used by web browser components that can process raw headers, such as Flash.	-	1	0.00%	0.00%
TOTAL	26			11,273	22,196	22.44%	35.79%
Confidentiality	200	Exposure of Sensitive Information to an Unauthorized Actor	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.	214	5,941	6.01%	0.68%
	522	Insufficiently Protected Credentials	The product transmits or stores authentication credentials, but it uses an insecure method that is susceptible to unauthorized interception and/or retrieval.	40	472	0.48%	0.13%
	319	Cleartext Transmission of Sensitive Information	The software transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors.	19	236	0.24%	0.06%
	532	Insertion of Sensitive Information into Log File	Information written to log files can be of a sensitive nature and give valuable guidance to an attacker or expose sensitive user information.	2	179	0.18%	0.01%
	311	Missing Encryption of Sensitive Data	The software does not encrypt sensitive or critical information before storage or transmission.	8	149	0.15%	0.03%
	312	Cleartext Storage of Sensitive Information	The application stores sensitive information in cleartext within a resource that might be accessible to another control sphere.	5	124	0.13%	0.02%
	922	Insecure Storage of Sensitive Information	The software stores sensitive information without properly limiting read or write access by unauthorized actors.	1	38	0.04%	0.00%
	538	Insertion of Sensitive Information into Externally-Accessible File or Directory	The product places sensitive information into files or directories that are accessible to actors who are allowed to have access to the files, but not to the sensitive information.	-	9	0.01%	0.00%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
Confidentiality Cont'd	924	Improper Enforcement of Message Integrity During Transmission in a Communication Channel	The software establishes a communication channel with an endpoint and receives a message from that endpoint, but it does not sufficiently ensure that the message was not modified during transmission.	-	5	0.01%	0.00%
TOTAL	9			289	7,153	7.23%	0.92%
Compartmentalization	668	Exposure of Resource to Wrong Sphere	The product exposes a resource to the wrong control sphere, providing unintended actors with inappropriate access to the resource.	22	129	0.13%	0.07%
	1021	Improper Restriction of Rendered UI Layers or Frames	The web application does not restrict or incorrectly restricts frame objects or UI layers that belong to another application or domain, which can lead to user confusion about which interface the user is interacting with.	3	90	0.09%	0.01%
	829	Inclusion of Functionality from Untrusted Control Sphere	The software imports, requires, or includes executable functionality (such as a library) from a source that is outside of the intended control sphere.	7	37	0.04%	0.02%
	494	Download of Code Without Integrity Check	The product downloads source code or an executable from a remote location and executes the code without sufficiently verifying the origin and integrity of the code.	15	36	0.04%	0.05%
	610	Externally Controlled Reference to a Resource in Another Sphere	The product uses an externally controlled name or reference that resolves to a resource that is outside of the intended control sphere.	8	24	0.02%	0.03%
	669	Incorrect Resource Transfer Between Spheres	The product does not properly transfer a resource/behavior to another sphere, or improperly imports a resource/behavior from another sphere, in a manner that provides unintended control over that resource.	3	16	0.02%	0.01%
	706	Use of Incorrectly-Resolved Name or Reference	The software uses a name or reference to access a resource, but the name/reference resolves to a resource that is outside of the intended control sphere.	3	15	0.02%	0.01%
	270	Privilege Context Switching Error	The software does not properly manage privileges while it is switching between different contexts that have different privileges or spheres of control.	1	2	0.00%	0.00%
	497	Exposure of Sensitive System Information to an Unauthorized Control Sphere	The application does not properly prevent sensitive system-level information from being accessed by unauthorized actors who do not have the same level of access to the underlying system as the application does.	-	1	0.00%	0.00%
	527	Exposure of Version-Control Repository to an Unauthorized Control Sphere	The product stores a CVS, git, or other repository in a directory, archive, or other resource that is stored, transferred, or otherwise made accessible to unauthorized actors.	-	1	0.00%	0.00%
TOTAL	10			62	351	0.35%	0.20%
Access Control*	264	Permissions, Privileges, and Access Controls	The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize "../" sequences that can resolve to a location that is outside of that directory.	1,364	3,625	3.67%	4.33%
	287	Improper Authentication	When an actor claims to have a given identity, the software does not prove or insufficiently proves that the claim is correct.	953	2,050	2.07%	3.03%
	400	Uncontrolled Resource Consumption	The software does not properly control the allocation and maintenance of a limited resource, thereby enabling an actor to influence the amount of resources consumed, eventually leading to the exhaustion of available resources.	208	858	0.87%	0.66%
	284	Improper Access Control	The software does not restrict or incorrectly restricts access to a resource from an unauthorized actor.	283	857	0.87%	0.90%
	269	Improper Privilege Management	The software does not properly assign, modify, track, or check privileges for an actor, creating an unintended sphere of control for that actor.	218	679	0.69%	0.69%
	863	Incorrect Authorization	The software performs an authorization check when an actor attempts to access a resource or perform an action, but it does not correctly perform the check. This allows attackers to bypass intended access restrictions.	96	610	0.62%	0.30%
	295	Improper Certificate Validation	The software does not validate, or incorrectly validates, a certificate.	42	604	0.61%	0.13%
	798	Use of Hard-coded Credentials	The software contains hard-coded credentials, such as a password or cryptographic key, which it uses for its own inbound authentication, outbound communication to external components, or encryption of internal data.	365	554	0.56%	1.16%
	255	Credentials Management Errors	Weaknesses in this category are related to the management of credentials.	251	539	0.55%	0.80%
	276	Incorrect Default Permissions	During installation, installed file permissions are set to allow anyone to modify those files.	96	477	0.48%	0.30%
	862	Missing Authorization	The software does not perform an authorization check when an actor attempts to access a resource or perform an action.	62	460	0.47%	0.20%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
Access Control* Cont'd	732	Incorrect Permission Assignment for Critical Resource	The product specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors.	101	445	0.45%	0.32%
	254	7PK - Security Features	Software security is not security software. Here we're concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management.	56	346	0.35%	0.18%
	306	Missing Authentication for Critical Function	The software does not perform any authentication for functionality that requires a provable user identity or consumes a significant amount of resources.	160	341	0.34%	0.51%
	345	Insufficient Verification of Data Authenticity	The software does not sufficiently verify the origin or authenticity of data, in a way that causes it to accept invalid data.	32	132	0.13%	0.10%
	346	Origin Validation Error	The software does not properly verify that the source of data or communication is valid.	15	119	0.12%	0.05%
	639	Authorization Bypass Through User-Controlled Key	The system's authorization functionality does not prevent one user from gaining access to another user's data or record by modifying the key value identifying the data.	9	94	0.10%	0.03%
	290	Authentication Bypass by Spoofing	This attack-focused weakness is caused by improperly implemented authentication schemes that are subject to spoofing attacks.	8	75	0.08%	0.03%
	285	Improper Authorization	The software does not perform or incorrectly performs an authorization check when an actor attempts to access a resource or perform an action.	13	72	0.07%	0.04%
	915	Improperly Controlled Modification of Dynamically-Determined Object Attributes	The software receives input from an upstream component that specifies multiple attributes, properties, or fields that are to be initialized or updated in an object, but it does not properly control which attributes can be modified.	21	55	0.06%	0.07%
	404	Improper Resource Shutdown or Release	The program does not release or incorrectly releases a resource before it is made available for re-use.	12	53	0.05%	0.04%
	281	Improper Preservation of Permissions	The software does not preserve permissions or incorrectly preserves permissions when copying, restoring, or sharing objects, which can cause them to have less restrictive permissions than intended.	5	45	0.05%	0.02%
	275	Permission Issues	Weaknesses in this category are related to improper assignment or handling of permissions.	4	35	0.04%	0.01%
	913	Improper Control of Dynamically-Managed Code Resources	The software does not properly restrict reading from or writing to dynamically-managed code resources such as variables, objects, classes, attributes, functions, or executable instructions or statements.	5	14	0.01%	0.02%
	297	Improper Validation of Certificate with Host Mismatch	The software communicates with a host that provides a certificate, but the software does not properly ensure that the certificate is actually associated with that host.	-	10	0.01%	0.00%
	749	Exposed Dangerous Method or Function	The software provides an Applications Programming Interface (API) or similar interface for interaction with external actors, but the interface includes a dangerous method or function that is not properly restricted.	5	10	0.01%	0.02%
	288	Authentication Bypass Using an Alternate Path or Channel	A product requires authentication, but the product has an alternate path or channel that does not require authentication.	2	6	0.01%	0.01%
	441	Confused Deputy	The product receives a request, message, or directive from an upstream component, but the product does not sufficiently preserve the original source of the request before forwarding the request to an external actor that is outside of the product's control sphere	-	6	0.01%	0.00%
	920	Improper Restriction of Power Consumption	The software operates in an environment in which power is a limited resource that cannot be automatically replenished, but the software does not properly restrict the amount of power that its operation consumes.	5	6	0.01%	0.02%
	300	Channel Accessible by Non-Endpoint	The product does not adequately verify the identity of actors at both ends of a communication channel, or does not adequately ensure the integrity of the channel, in a way that allows the channel to be accessed or influenced by an actor that is not an endpoint.	-	3	0.00%	0.00%
	353	Missing Support for Integrity Check	The software uses a transmission protocol that does not include a mechanism for verifying the integrity of the data during transmission, such as a checksum.	-	2	0.00%	0.00%
	379	Creation of Temporary File in Directory with Insecure Permissions	The software creates a temporary file in a directory whose permissions allow unintended actors to determine the file's existence or otherwise access that file.	-	2	0.00%	0.00%
	299	Improper Check for Certificate Revocation	The software does not check or incorrectly checks the revocation status of a certificate, which may cause it to use a certificate that has been compromised.	-	1	0.00%	0.00%
	350	Reliance on Reverse DNS Resolution for a Security-Critical Action	The software performs reverse DNS resolution on an IP address to obtain the hostname and make a security decision, but it does not properly ensure that the IP address is truly associated with the hostname.	-	1	0.00%	0.00%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
Access Control* Cont'd	405	Amplification	Software that does not appropriately monitor or control resource consumption can lead to adverse system performance.	-	1	0.00%	0.00%
TOTAL	35			4,391	13,187	13.33%	13.94%
N/A (No Micropolicy)	310	Cryptographic Issues	Weaknesses in this category are related to the design and implementation of data confidentiality and integrity. Frequently these deal with the use of encoding techniques, encryption libraries, and hashing algorithms. The weaknesses in this category could lead to a degradation of the quality data if they are not addressed.	257	2,331	2.36%	0.82%
	772	Missing Release of Resource after Effective Lifetime	The software does not release a resource after its effective lifetime has ended, i.e., after the resource is no longer needed.	100	724	0.73%	0.32%
	362	Race Condition	The program contains a code sequence that can run concurrently with other code, and the code sequence requires temporary, exclusive access to a shared resource, but a timing window exists in which the shared resource can be modified by another code sequence that is operating concurrently.	124	341	0.34%	0.39%
	369	Divide By Zero	This weakness typically occurs when an unexpected value is provided to the product, or if an error occurs that is not properly detected. It frequently occurs in calculations involving physical dimensions such as size, length, width, and height.	27	302	0.31%	0.09%
	835	Infinite Loop	The program contains an iteration or loop with an exit condition that cannot be reached, i.e., an infinite loop.	65	301	0.30%	0.21%
	326	Inadequate Encryption Strength	The software stores or transmits sensitive data using an encryption scheme that is theoretically sound, but is not strong enough for the level of protection required.	26	220	0.22%	0.08%
	19	Data Processing Errors	Weaknesses in this category are typically found in functionality that processes data. Data processing is the manipulation of input to retrieve or save information.	79	206	0.21%	0.25%
	16	Configuration	Weaknesses in this category are typically introduced during the configuration of the software.	81	204	0.21%	0.26%
	704	Incorrect Type Conversion or Cast	The software does not correctly convert an object, resource, or structure from one type to a different type.	34	178	0.18%	0.11%
	327	Use of a Broken or Risky Cryptographic Algorithm	The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the exposure of sensitive information.	14	159	0.16%	0.04%
	347	Improper Verification of Cryptographic Signature	The software does not verify, or incorrectly verifies, the cryptographic signature for data.	22	147	0.15%	0.07%
	755	Improper Handling of Exceptional Conditions	The software does not handle or incorrectly handles an exceptional condition	36	136	0.14%	0.11%
	401	Missing Release of Memory after Effective Lifetime	The software does not sufficiently track and release allocated memory after it has been used, which slowly consumes remaining memory.	37	135	0.14%	0.12%
	1188	Insecure Default Initialization of Resource	The software initializes or sets a resource with a default that is intended to be changed by the administrator, but the default is not secure.	97	133	0.13%	0.31%
	674	Uncontrolled Recursion	The product does not properly control the amount of recursion which takes place, consuming excessive resources, such as allocated memory or the program stack.	6	108	0.11%	0.02%
	203	Observable Discrepancy	The product behaves differently or sends different responses under different circumstances in a way that is observable to an unauthorized actor, which exposes security-relevant information about the state of the product, such as whether a particular operation was successful or not.	2	107	0.11%	0.01%
	307	Improper Restriction of Excessive Authentication Attempts	The software does not implement sufficient measures to prevent multiple failed authentication attempts within in a short time frame, making it more susceptible to brute force attacks.	13	100	0.10%	0.04%
	209	Generation of Error Message Containing Sensitive Information	The software generates an error message that includes sensitive information about its environment, users, or associated data.	2	97	0.10%	0.01%
	754	Improper Check for Unusual or Exceptional Conditions	The software does not check or incorrectly checks for unusual or exceptional conditions that are not expected to occur frequently during day to day operation of the software.	13	93	0.09%	0.04%
	640	Weak Password Recovery Mechanism for Forgotten Password	The software contains a mechanism for users to recover or change their passwords without knowing the original password, but the mechanism is weak.	14	85	0.09%	0.04%
	330	Use of Insufficiently Random Values	The software uses insufficiently random numbers or values in a security context that depends on unpredictable numbers.	11	84	0.08%	0.03%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
N/A (No Micropolicy) Cont’d	521	Weak Password Requirements	The product does not require that users should have strong passwords, which makes it easier for attackers to compromise user accounts.	23	68	0.07%	0.07%
	665	Improper Initialization	The software does not initialize or incorrectly initializes a resource, which might leave the resource in an unexpected state when it is accessed or used.	21	68	0.07%	0.07%
	427	Uncontrolled Search Path Element	The product uses a fixed or controlled search path to find resources, but one or more locations in that path can be under the control of unintended actors.	29	65	0.07%	0.09%
	909	Missing Initialization of Resource	The software does not initialize a critical resource.	-	59	0.06%	0.00%
	116	Improper Encoding or Escaping of Output	The software prepares a structured message for communication with another component, but encoding or escaping of the data is either missing or done incorrectly. As a result, the intended structure of the message is not preserved.	8	50	0.05%	0.03%
	552	Files or Directories Accessible to External Parties	The product makes files or directories accessible to unauthorized actors, even though they should not be.	8	50	0.05%	0.03%
	294	Authentication Bypass by Capture-replay	A capture-replay flaw exists when the design of the software makes it possible for a malicious user to sniff network traffic and bypass authentication by replaying it to the server in question to the same effect as the original message (or with minor changes).	6	46	0.05%	0.02%
	681	Incorrect Conversion between Numeric Types	When converting from one data type to another, such as long to integer, data can be omitted or translated in a way that produces unexpected values. If the resulting values are used in a sensitive context, then dangerous behaviors may occur.	15	46	0.05%	0.05%
	834	Excessive Iteration	The software performs an iteration or loop without sufficiently limiting the number of times that the loop is executed.	26	45	0.05%	0.08%
	338	Use of Cryptographically Weak Pseudo-Random Number Generator	The product uses a Pseudo-Random Number Generator (PRNG) in a security context, but the PRNG's algorithm is not cryptographically strong.	6	42	0.04%	0.02%
	320	Key Management Errors	Weaknesses in this category are related to errors in the management of cryptographic keys.	8	41	0.04%	0.03%
	776	XML Entity Expansion	The software uses XML documents and allows their structure to be defined with a Document Type Definition (DTD), but it does not properly control the number of recursive definitions of entities.	4	39	0.04%	0.01%
	682	Incorrect Calculation	The software performs a calculation that generates incorrect or unintended results that are later used in security-critical decisions or resource management.	10	34	0.03%	0.03%
	354	Improper Validation of Integrity Check Value	The software does not validate or incorrectly validates the integrity check values or "checksums" of a message. This may prevent it from detecting if the data has been modified or corrupted in transmission.	4	32	0.03%	0.01%
	367	TOCTOU Race Condition	The software checks the state of a resource before using that resource, but the resource's state can change between the check and the use in a way that invalidates the results of the check. This can cause the software to perform invalid actions when the resource is in an unexpected state.	11	31	0.03%	0.03%
	693	Protection Mechanism Failure	The product does not use or incorrectly uses a protection mechanism that provides sufficient defense against directed attacks against the product.	4	31	0.03%	0.01%
	358	Improperly Implemented Security Check for Standard	The software does not implement or incorrectly implements one or more security-relevant checks as specified by the design of a standardized algorithm, protocol, or technique.	4	27	0.03%	0.01%
	252	Unchecked Return Value	The software does not check the return value from a method or function, which can prevent it from detecting unexpected states and conditions.	7	26	0.03%	0.02%
	331	Insufficient Entropy	The software uses an algorithm or scheme that produces insufficient entropy, leaving patterns or clusters of values that are more likely to occur than others.	1	26	0.03%	0.00%
	388	7PK - Errors	It includes weaknesses that occur when an application does not properly handle errors that occur during processing.	14	26	0.03%	0.04%
	916	Use of Password Hash With Insufficient Computational Effort	The software generates a hash for a password, but it uses a scheme that does not provide a sufficient level of computational effort that would make password cracking attacks infeasible or expensive.	4	24	0.02%	0.01%
	185	Incorrect Regular Expression	When the regular expression is used in protection mechanisms such as filtering or validation, this may allow an attacker to bypass the intended restrictions on the incoming data.	4	17	0.02%	0.01%
	670	Always-Incorrect Control Flow Implementation	The code contains a control flow path that does not reflect the algorithm that the path is intended to implement, leading to incorrect behavior any time this path is navigated.	3	14	0.01%	0.01%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
N/A (No Micropolicy) Cont'd	697	Incorrect Comparison	The software compares two entities in a security-relevant context, but the comparison is incorrect, which may lead to resultant weaknesses.	4	14	0.01%	0.01%
	417	Communication Channel Errors	Weaknesses in this category are related to improper handling of communication channels and access paths.	3	13	0.01%	0.01%
	667	Improper Locking	The software does not properly acquire or release a lock on a resource, leading to unexpected resource state changes and behaviors.	7	13	0.01%	0.02%
	118	Range Error	The software does not restrict or incorrectly restricts operations within the boundaries of a resource that is accessed using an index or pointer, such as memory or files.	8	12	0.01%	0.03%
	459	Incomplete Cleanup	The software does not properly "clean up" and remove temporary or supporting resources after they have been used.	2	12	0.01%	0.01%
	73	External Control of File Name or Path	The software allows user input to control or influence paths or file names that are used in filesystem operations.	-	11	0.01%	0.00%
	332	Insufficient Entropy in PRNG	The lack of entropy available for, or used by, a Pseudo-Random Number Generator (PRNG) can be a stability and security threat.	1	11	0.01%	0.00%
	335	Incorrect Usage of Seeds in Pseudo-Random Number Generator	The software uses a Pseudo-Random Number Generator (PRNG) that does not correctly manage seeds.	1	11	0.01%	0.00%
	172	Encoding Error	The software does not properly encode or decode the data, resulting in unexpected values.	5	9	0.01%	0.02%
	212	Improper Removal of Sensitive Information Before Storage or Transfer	The product stores, transfers, or shares a resource that contains sensitive information, but it does not properly remove that information before the product makes the resource available to unauthorized actors.	2	8	0.01%	0.01%
	436	Interpretation Conflict	Product A handles inputs or steps differently than Product B, which causes A to perform incorrect actions based on its perception of B's state.	1	8	0.01%	0.00%
	273	Improper Check for Dropped Privileges	The software attempts to drop privileges but does not check or incorrectly checks to see if the drop succeeded.	5	7	0.01%	0.02%
	361	7PK - Time and State	It includes weaknesses related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads	3	7	0.01%	0.01%
	662	Improper Synchronization	The software utilizes multiple threads or processes to allow temporary access to a shared resource that can only be exclusive to one process at a time, but it does not properly synchronize these actions, which might cause simultaneous accesses of this resource by multiple threads or processes.	3	7	0.01%	0.01%
	184	Incomplete List of Disallowed Inputs	The product implements a protection mechanism that relies on a list of inputs (or properties of inputs) that are not allowed by policy or otherwise require other action to neutralize before additional processing takes place, but the list is incomplete, leading to resultant weaknesses.	2	6	0.01%	0.01%
	266	Incorrect Privilege Assignment	A product incorrectly assigns a privilege to a particular actor, creating an unintended sphere of control for that actor.	-	5	0.01%	0.00%
	305	Authentication Bypass by Primary Weakness	The authentication algorithm is sound, but the implemented mechanism can be bypassed as the result of a separate weakness that is primary to the authentication error.	3	5	0.01%	0.01%
	694	Use of Multiple Resources with Duplicate Identifier	The software uses multiple resources that can have the same identifier, in a context in which unique identifiers are required.	-	5	0.01%	0.00%
	822	Untrusted Pointer Dereference	The program obtains a value from an untrusted source, converts this value to a pointer, and dereferences the resulting pointer.	-	5	0.01%	0.00%
	99	Resource Injection	The software receives input from an upstream component, but it does not restrict or incorrectly restricts the input before it is used as an identifier for a resource that may be outside the intended sphere of control.	2	4	0.00%	0.01%
	199	Information Management Errors	Weaknesses in this category are related to improper handling of sensitive information.	-	4	0.00%	0.00%
	201	Insertion of Sensitive Information Into Sent Data	The code transmits data to another actor, but a portion of the data includes sensitive information that should not be accessible to that actor.	-	4	0.00%	0.00%
	407	Inefficient Algorithmic Complexity	An algorithm in a product has an inefficient worst-case computational complexity that may be detrimental to system performance and can be triggered by an attacker, typically using crafted manipulations that ensure that the worst case is being reached.	1	4	0.00%	0.00%
	36	Absolute Path Traversal	The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize absolute path sequences such as "/abs/path" that can resolve to a location that is outside of that directory.	2	3	0.00%	0.01%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
N/A (No Micropolicy) Cont'd	256	Unprotected Storage of Credentials	Storing a password in plaintext may result in a system compromise.	-	3	0.00%	0.00%
	321	Use of Hard-coded Cryptographic Key	The use of a hard-coded cryptographic key significantly increases the possibility that encrypted data may be recovered.	1	3	0.00%	0.00%
	774	Allocation of File Descriptors or Handles Without Limits or Throttling	The software allocates file descriptors or handles on behalf of an actor without imposing any restrictions on how many descriptors can be allocated, in violation of the intended security policy for that actor.	-	3	0.00%	0.00%
	838	Inappropriate Encoding for Output Context	The software uses or specifies an encoding when generating output to a downstream component, but the specified encoding is not the same as the encoding that is expected by the downstream component.	1	3	0.00%	0.00%
	35	Path Traversal: '.../...//'	The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize '.../...//' (doubled triple dot slash) sequences that can resolve to a location that is outside of that directory.	-	2	0.00%	0.00%
	197	Numeric Truncation Error	Truncation errors occur when a primitive is cast to a primitive of a smaller size and data is lost in the conversion.	-	2	0.00%	0.00%
	248	Uncaught Exception	An exception is thrown from a function, but it is not caught.	-	2	0.00%	0.00%
	250	Execution with Unnecessary Privileges	The software performs an operation at a privilege level that is higher than the minimum level required, which creates new weaknesses or amplifies the consequences of other weaknesses.	1	2	0.00%	0.00%
	303	Incorrect Implementation of Authentication Algorithm	The requirements for the software dictate the use of an established authentication algorithm, but the implementation of the algorithm is incorrect.	-	2	0.00%	0.00%
	342	Predictable Exact Value from Previous Values	An exact value or random number can be precisely predicted by observing previous values.	1	2	0.00%	0.00%
	385	Covert Timing Channel	Covert timing channels convey information by modulating some aspect of system behavior over time, so that the program receiving the information can observe system behavior and infer protected information.	-	2	0.00%	0.00%
	642	External Control of Critical State Data	The software stores security-critical state information about its users, or the software itself, in a location that is accessible to unauthorized actors.	1	2	0.00%	0.00%
	664	Improper Control of a Resource Through its Lifetime	The software does not maintain or incorrectly maintains control over a resource throughout its lifetime of creation, use, and release.	1	2	0.00%	0.00%
	759	Use of a One-Way Hash without a Salt	The software uses a one-way cryptographic hash against an input that should not be reversible, such as a password, but the software does not also use a salt as part of the input.	-	2	0.00%	0.00%
	912	Hidden Functionality	The software contains functionality that is not documented, not part of the specification, and not accessible through an interface or command sequence that is obvious to the software's users or administrators.	2	2	0.00%	0.01%
	98	PHP Remote File Inclusion	The PHP application receives input from an upstream component, but it does not restrict or incorrectly restricts the input before its usage in "require," "include," or similar functions.	-	1	0.00%	0.00%
	115	Misinterpretation of Input	The software misinterprets an input, whether from an attacker or another product, in a security-relevant fashion.	1	1	0.00%	0.00%
	208	Observable Timing Discrepancy	Two separate operations in a product require different amounts of time to complete, in a way that is observable to an actor and reveals security-relevant information about the state of the product, such as whether a particular operation was successful or not.	-	1	0.00%	0.00%
	240	Improper Handling of Inconsistent Structural Elements	The software does not handle or incorrectly handles when two or more structural elements should be consistent, but are not.	-	1	0.00%	0.00%
	257	Storing Passwords in a Recoverable Format	The storage of passwords in a recoverable format makes them subject to password reuse attacks by malicious users. In fact, it should be noted that recoverable encrypted passwords provide no significant benefit over plaintext passwords since they are subject not only to reuse by malicious attackers but also by malicious insiders. If a system administrator can recover a password directly, or use a brute force search on the available information, the administrator can use the password on other accounts.	-	1	0.00%	0.00%
	259	Use of Hard-coded Password	The software contains a hard-coded password, which it uses for its own inbound authentication or for outbound communication to external components.	1	1	0.00%	0.00%
	261	Weak Encoding for Password	Obscuring a password with a trivial encoding does not protect the password.	-	1	0.00%	0.00%
	271	Privilege Dropping / Lowering Errors	The software does not drop privileges before passing control of a resource to an actor that does not have those privileges.	-	1	0.00%	0.00%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
N/A (No Micropolicy) Cont'd	**280**	Improper Handling of Insufficient Permissions or Privileges	The application does not handle or incorrectly handles when it has insufficient privileges to access resources or functionality as specified by their permissions. This may cause it to follow unexpected code paths that may leave the application in an invalid state.	1	1	0.00%	0.00%
	302	Authentication Bypass by Assumed-Immutable Data	The authentication scheme or implementation uses key data elements that are assumed to be immutable, but can be controlled or modified by the attacker.	-	1	0.00%	0.00%
	313	Cleartext Storage in a File or on Disk	The application stores sensitive information in cleartext in a file, or on disk.	-	1	0.00%	0.00%
	317	Cleartext Storage of Sensitive Information in GUI	The application stores sensitive information in cleartext within the GUI.	-	1	0.00%	0.00%
	323	Reusing a Nonce, Key Pair in Encryption	Nonces should be used for the present occasion and only once.	-	1	0.00%	0.00%
	334	Small Space of Random Values	The number of possible random values is smaller than needed by the product, making it more susceptible to brute force attacks.	-	1	0.00%	0.00%
	371	State Issues	Weaknesses in this category are related to improper management of system state.	-	1	0.00%	0.00%
	406	Network Amplification	The software does not sufficiently monitor or control transmitted network traffic volume, so that an actor can cause the software to transmit more traffic than should be allowed for that actor.	-	1	0.00%	0.00%
	435	Improper Interaction Between Multiple Correctly-Behaving Entities	An interaction error occurs when two entities have correct behavior when running independently of each other, but when they are integrated as components in a larger system or process, they introduce incorrect behaviors that may cause resultant weaknesses.	-	1	0.00%	0.00%
	457	Use of Uninitialized Variable	The code uses a variable that has not been initialized, leading to unpredictable or unintended results.	-	1	0.00%	0.00%
	506	Embedded Malicious Code	The application contains code that appears to be malicious in nature.	-	1	0.00%	0.00%
	507	Trojan Horse	The software appears to contain benign or useful functionality, but it also contains code that is hidden from normal operation that violates the intended security policy of the user or the system administrator.	1	1	0.00%	0.00%
	540	Inclusion of Sensitive Information in Source Code	Source code on a web server or repository often contains sensitive information and should generally not be accessible to users.	-	1	0.00%	0.00%
	567	Unsynchronized Access to Shared Data in a Multithreaded Context	The product does not properly synchronize shared data, such as static variables across threads, which can lead to undefined behavior and unpredictable data changes.	-	1	0.00%	0.00%
	573	Improper Following of Specification by Caller	The software does not follow or incorrectly follows the specifications as required by the implementation language, environment, framework, protocol, or platform.	-	1	0.00%	0.00%
	603	Use of Client-Side Authentication	A client/server product performs authentication within client code but not in server code, allowing server-side authentication to be bypassed via a modified client that omits the authentication check.	1	1	0.00%	0.00%
	649	Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking	The software uses obfuscation or encryption of inputs that should not be mutable by an external actor, but the software does not use integrity checks to detect if those inputs have been modified.	-	1	0.00%	0.00%
	684	Incorrect Provision of Specified Functionality	The code does not function according to its published specifications, potentially leading to incorrect usage.	-	1	0.00%	0.00%
	710	Improper Adherence to Coding Standards	The software does not follow certain coding rules for development, which can lead to resultant weaknesses or increase the severity of the associated vulnerabilities.	1	1	0.00%	0.00%
	757	Algorithm Downgrade	A protocol or its implementation supports interaction between multiple actors and allows those actors to negotiate which algorithm should be used as a protection mechanism such as encryption or authentication, but it does not select the strongest algorithm that is available to both parties.	-	1	0.00%	0.00%
	778	Insufficient Logging	When a security-critical event occurs, the software either does not record the event or omits important details about the event when logging it.	-	1	0.00%	0.00%
	895	SFP Primary Cluster: Information Leak	This category identifies Software Fault Patterns (SFPs) within the Information Leak cluster (SFP23).	-	1	0.00%	0.00%
	923	Improper Restriction of Communication Channel to Intended Endpoints	The software establishes a communication channel to (or from) an endpoint for privileged or protected operations, but it does not properly ensure that it is communicating with the correct endpoint.	-	1	0.00%	0.00%

MICROPOLICY GROUPING	CWE ID #	CWE NAME	CWE DESCRIPTION	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	% OF ALL CVES (98,898)	% OF SEVERE CVES
N/A (No Micropolicy) Cont'd	1076	Insufficient Adherence to Expected Conventions	The product's architecture, source code, design, documentation, or other artifact does not follow required conventions.	-	1	0.00%	0.00%
TOTAL	115			1,349	7,352	7.43%	4.28%
GRAND TOTAL	243			31,501	98,898	100%	100%

TABLE 3: CoreGuard Micropolicy Protection Breakdown

MICROPOLICY GROUPING	TOTAL # OF SEVERE CVES (CVSS > 7.0)	TOTAL # OF CVES	TOTAL # OF CVES BLOCKED BY COREGUARD	% OF ALL CVES BLOCKED BY COREGUARD (98,898)	CWEs COVERED
Base Set (Heap, Stack & RWX)	13,564	27,512	27,512	27.82%	119 , 787 , 190 , 125 , 94 , 416 , 476 , 434 , 120 , 134 , 908 , 415 , 191 , 843 , 129 , 193 , 121 , 824 , 763 , 122 , 788 , 131 , 672 , 123 , 170 , 471 , 823 , 126 , 130 , 680 , 805 , 1285
Web Protection*	573	21,147	21,147	21.38%	79 , 352 , 611 , 601 , 918 , 426 , 384 , 59 , 613 , 444 , 425 , 93 , 113 , 565 , 472 , 539
Sanitization*	11,273	22,196	22,196	22.44%	20 , 89 , 22 , 78 , 502 , 74 , 77 , 770 , 617 , 917 , 1236 , 88 , 91 , 470 , 428 , 80 , 90 , 23 , 943 , 80 , 707 , 1286 , 87 , 96 , 117 , 178 , 644
Confidentiality	289	7,153	7,153	7.23%	200 , 522 , 319 , 532 , 311 , 312 , 922 , 538 , 924
Compartmentalization	62	351	351	0.35%	668 , 1021 , 829 , 494 , 610 , 669 , 706 , 270 , 497 , 527
Access Control*	4,391	13,187	13,187	13.33%	264 , 287 , 400 , 284 , 269 , 863 , 295 , 798 , 255 , 276 , 862 , 732 , 254 , 306 , 345 , 346 , 639 , 290 , 285 , 915 , 404 , 281 , 275 , 913 , 297 , 749 , 288 , 441 , 920 , 300 , 353 , 379 , 299 , 350 , 405
N/A (No Micropolicy)	1,349	7,352	0	0.00%	310 , 772 , 362 , 369 , 835 , 326 , 19 , 16 , 704 , 327 , 347 , 755 , 401 , 1188 , 674 , 203 , 307 , 209 , 754 , 640 , 330 , 521 , 665 , 427 , 909 , 116 , 552 , 294 , 681 , 834 , 338 , 320 , 776 , 682 , 354 , 367 , 693 , 358 , 252 , 331 , 388 , 916 , 185 , 670 , 697 , 417 , 667 , 118 , 459 , 73 , 332 , 335 , 172 , 212 , 436 , 273 , 361 , 662 , 184 , 266 , 305 , 694 , 822 , 99 , 199 , 201 , 407 , 36 , 256 , 321 , 774 , 838 , 35 , 197 , 248 , 250 , 303 , 342 , 385 , 642 , 664 , 759 , 912 , 98 , 115 , 208 , 240 , 257 , 259 , 261 , 271 , 280 , 302 , 313 , 317 , 323 , 334 , 371 , 406 , 435 , 457 , 506 , 507 , 540 , 567 , 573 , 603 , 649 , 684 , 710 , 757 , 778 , 895 , 923 , 1076
TOTAL	31,501	98,898	91,546	92.57%	