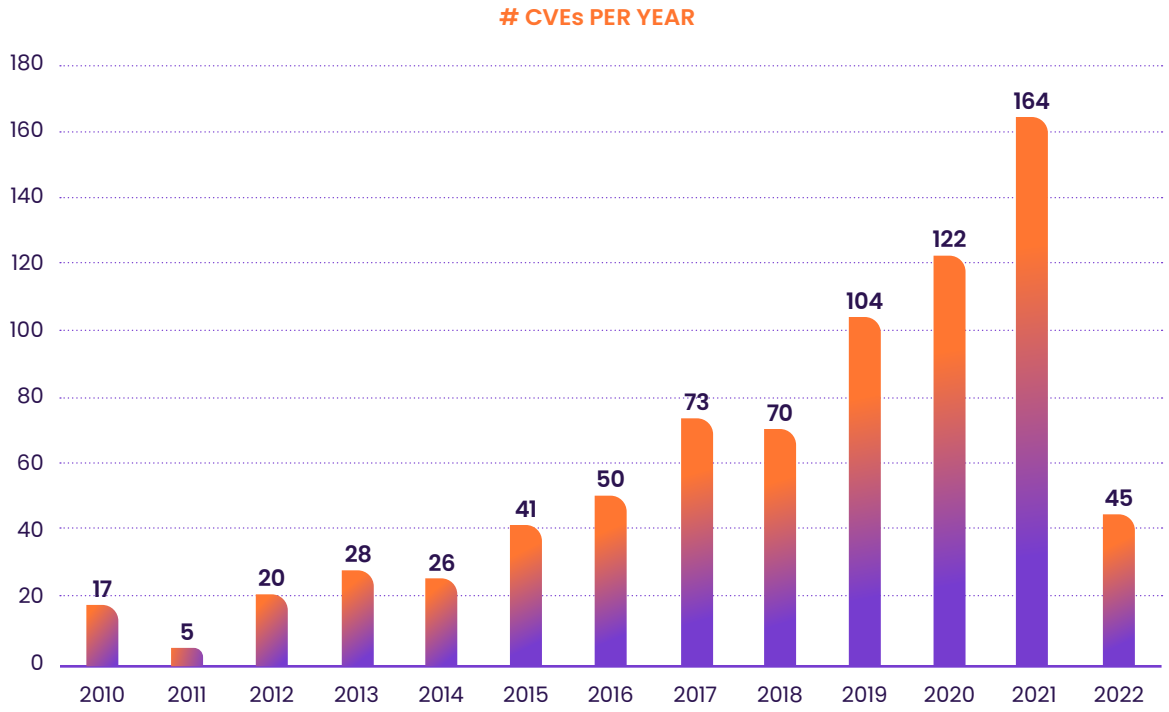
Rezilion

Vintage Vulnerabilities Are Still In Style

WHILE CYBER CRIMINALS' FASHION TASTE (at least according to popular media), remains loyal to the good old hoodie, their taste for vintage vulnerabilities is no different.

As we will demonstrate in the following research report, known vulnerabilities, even ones dating back more than a decade to the past, are still extremely common and as such, still pose significant risk. Even though fixes for these vulnerabilities have been available for years and despite them being known to be exploited in the wild, software and devices remain vulnerable.

In fact, if we explore the CISA Known Exploited Vulnerabilities list, we can see that there are over 400 known exploited vulnerabilities dating back before 2020.



We decided to take a closer look at some of these vulnerabilities and see whether they still pose a threat.

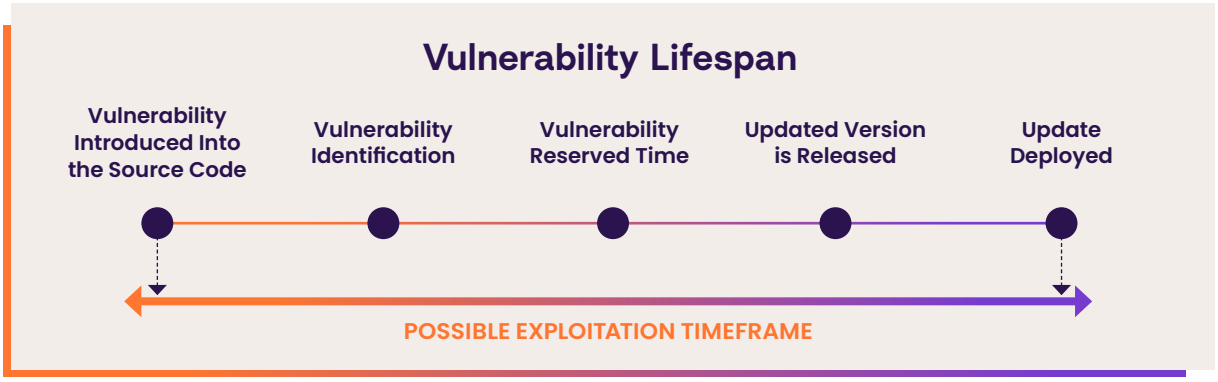
Overall, we have been able to identify over 4.5 million internet-facing devices which, to this date, are vulnerable to vulnerabilities discovered between 2010 to 2020. For most of these vulnerabilities, we also identified active scanning/exploitation attempts.

CVE ID	AFFECTED PRODUCT	VULNERABILITY NAME	CVSS	# AFFFECTED DEVICES
CVE-2010-4344	Exim — Internet Mailer	Exim Heap-Based Buffer Overflow Vulnerability	9.3	4,797
CVE-2010-4345	Exim — Internet Mailer	Exim Privilege Escalation Vulnerability	6.9	12,891
CVE-2012-1823	PHP	PHP-CGI Query String Parameter Vulnerability	7.5	569,190
CVE-2014-0160	OpenSSL	HeartBleed — OpenSSL Information Disclosure Vulnerability	7.5	228,876
CVE-2015-1635	Microsoft HTTP.sys	Microsoft HTTP.sys Remote Code Execution Vulnerability	10	164,451
CVE-2017-7269	Microsoft Internet Information Services (IIS)	Microsft Windows Server 2003 R2 IIS WEBDAV buffer overflow Remote Code Execution vulnerability	9.8	9,557
CVE-2018-13379	Fortinet FortiOS	Fortinet FortiOS SSL VPN credential exposure vulnerability	9.8	538,310
CVE-2018-6789	Exim — Internet Mailer	Exim Buffer Overflow Vulnerability	9.8	73,592
CVE-2018-7600	Drupal	Drupal module configuration vulnerability	9.8	9,980
CVE-2018-7602	Drupal	Drupal Core Remote Code Execution Vulnerability	9.8	6,455
CVE-2018-8581	Microsoft Exchange Server	Microsoft Exchange Server Privilege Escalation Vulnerability	7.4	662
CVE-2019-0211	Apache HTTP Server	Apache HTTP Server scoreboard vulnerability	7.8	2,630,296
CVE-2019-0708	Microsoft Remote Desktop Services	"BlueKeep" Microsoft Windows Remote Desktop Remote Code Execution Vulnerability	9.8	75,237
CVE-2019-10149	Exim Mail Transfer Agent (MTA)	Exim Mail Transfer Agent (MTA) Improper Input Validation	9.8	84,421
CVE-2019-11043	PHP FastCGI Process Manager (FPM)	PHP FastCGI Process Manager (FPM) Buffer Overflow Vulnerability	9.8	294,007
CVE-2019-11510	Pulse Connect Secure	Pulse Connect Secure VPN arbitrary file reading vulnerability (COVID-19-CTI list)	10	293
CVE-2019-1579	Palo Alto Networks PAN-OS	Palo Alto Networks PAN-OS Remote Code Execution Vulnerability	8.1	890
CVE-2019-1652	Cisco RV320 and RV325 Routers	Cisco Small Business Routers Improper Input Validation Vulnerability	7.2	3,053
CVE-2019-1653	Cisco RV320 and RV325 Routers	Cisco RV320 and RV325 Routers Improper Access Control Vulnerability (COVID-19-CTI list)	7.5	3,051
CVE-2019-16928	Exim — Internet Mailer	Exim Out-of-bounds Write Vulnerability	9.8	14,115
CVE-2019-19781	Citrix Application Delivery Controller (ADC) & Gateway	Citrix Application Delivery Controller and Citrix Gateway Vulnerability	9.8	271
CVE-2019-6340	Drupal	Drupal Core Remote Code Execution Vulnerability	8.1	1,718
CVE-2020-0796	Microsoft SMBv3	Microsoft SMBv3 Remote Code Execution Vulnerability	10	165,096
CVE-2020-11651	SaltStack Salt	SaltStack Salt Authentication Bypass	9.8	26
CVE-2020-11652	SaltStack Salt	SaltStack directory traversal failure to sanitize untrusted input	6.5	26
CVE-2020-13671	Drupal	Drupal core Un-restricted Upload of File	8.8	28,225
CVE-2020-1938	Apache Tomcat	Apache Tomcat Improper Privilege Management Vulnerability	9.8	33,009
CVE-2020-2021	Palo Alto Networks PAN-OS	Palo Alto PAN-OS Authentication Bypass Vulnerability	9.3	4,209
CVE-2020-5902	F5 BIG-IP	F5 BIG-IP Traffic Management User Interface Remote Code Execution Vulnerability	9.8	40

Before we dive into the actual research, let's understand more about the lifetime of a vulnerability.

Some Background

THERE ARE VARIOUS STAGES THROUGHOUT THE LIFETIME OF A VULNERABILITY from the time the vulnerability is introduced into the code until the point it is effectively patched and does not pose any additional risk.



That timespan can be divided into two distinct ranges:

1. From the moment the vulnerable code is introduced until the vulnerability is identified/reported. During that time period, the vulnerability exists, potentially exploitable yet no one is aware of it and it will not be patched. If by any chance malicious actors become aware of the vulnerability at this stage it is classified as a 0-day vulnerability.
2. From the moment the vendor/maintainer responsible for the vulnerable code issues a patch or a fixed release until the patch is deployed. At this stage, the vulnerability is already known and will only be exploitable in environments in which the patch/fix was not yet installed.

This research will focus on the latter group and will highlight some of the most critical vulnerabilities which already have a fix, yet as evidence shows, the attack surface vulnerable to these vulnerabilities still exists, hence the likelihood of exploitation is high.

It is important to note that the results presented in this research are just the tip of the iceberg in terms of the actual vulnerable attack surface for the following reasons:

- ✓ We only focused on CVEs discovered in 2020 or before.
- ✓ The information presented is limited by the capabilities of the threat intelligence tools utilized. These tools only have visibility into software elements that are on internet-facing (i.e publicly accessible) servers. It's safe to assume that there are numerous other vulnerable servers that are not publicly facing (i.e on internal networks).
- ✓ This research focuses mostly on vulnerabilities that can be exploited over the network using a remote attack vector. There are many other types of vulnerabilities that remained out of scope for this research (such as local privilege escalation for example) which an attacker can utilize once establishing an initial foothold of the system to perform lateral movement, data exfiltration, and more.

Why Does It Matter?

SIMPLY PUT, THIS SHOULD BE THE EASY PART. If we as an industry can't get to a point where vulnerabilities that were made public and fixed years ago, and are known to be exploited for years are irrelevant, then we are playing into the hand of the adversaries.

As a recent study conducted by researchers from Trento University in Italy suggests, contrary to the common misconception, most Advanced Persistent Threat (APT) groups utilize publicly known vulnerabilities.

The researchers have manually curated a dataset of APT attacks that covers 86 APTs and 350 campaigns that occurred between 2008 to 2020, examining the attack vectors and types of exploited vulnerabilities utilized (e.g., zero-days vs public vulnerabilities), as well as affected software and versions.

The data shows that the majority of vulnerabilities used by APT groups are known, unpatched vulnerabilities and highlights the importance of applying timely patches:

"In contrast to expectation, we showed that preventive mechanisms like updates can influence the probability of being compromised by APT."

Data Sources

THROUGHOUT THIS RESEARCH, we have utilized multiple data sources. In this section, we will provide a quick overview of the main data sources used.

Shodan.io — A search engine for internet-connected devices. Shodan gathers information about all devices directly connected to the internet. Most of the data Sodan collects is taken from banners, which are metadata about software that's running on a device.

GreyNoise — GreyNoise passively collects packets from hundreds of thousands of IPs seen scanning the internet every day. It then analyzes and enriches this data to identify behavior, methods, and intent, providing insights regarding active exploitation attempts.

Let's explore some of the notable "vintage" vulnerabilities, dating back before 2020 which are still very much in fashion.

Prominent Examples

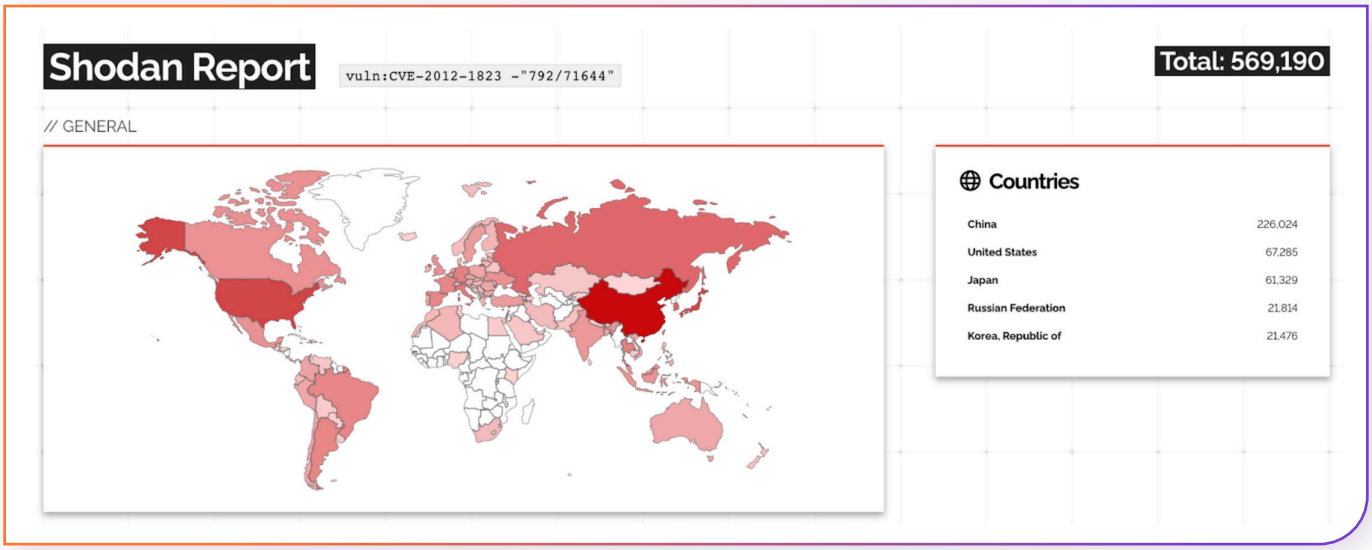
CVE-2012-1823 — PHP CGI Remote Code Execution
CVSS2 — 7.5
Known to be exploited in the wild — Yes
Vulnerability Age: **10 Years**

An improper input validation vulnerability that allows remote attackers to execute arbitrary code by placing command-line options in the query string.

The bug is related to an error on how the URI is used and provided to PHP CGI when a URL lacks the `=` character. Since the URI is passed to the *php-cgi* binary without enough filtering or encoding, an attacker is able to pass an extra argument to the *php-cgi* command line.

The interesting aspect about this vulnerability is that it was discovered during a Capture the Flag competition and it took around four months until a patch was released.

Potential Vulnerable Internet-Facing Applications According to Shodan.io — over 569,000



Even though this vulnerability is a decade old, GreyNoise reports on **15** unique IP addresses trying to exploit CVE-2012-1823 in the last 30 days.

CVE-2014-0160 ("HeartBleed") — OpenSSL Sensitive Information Leak From Process Memory Vulnerability
CVSS2 — 7.5
Known to be exploited in the wild — Yes
Vulnerability Age: **8 Years**

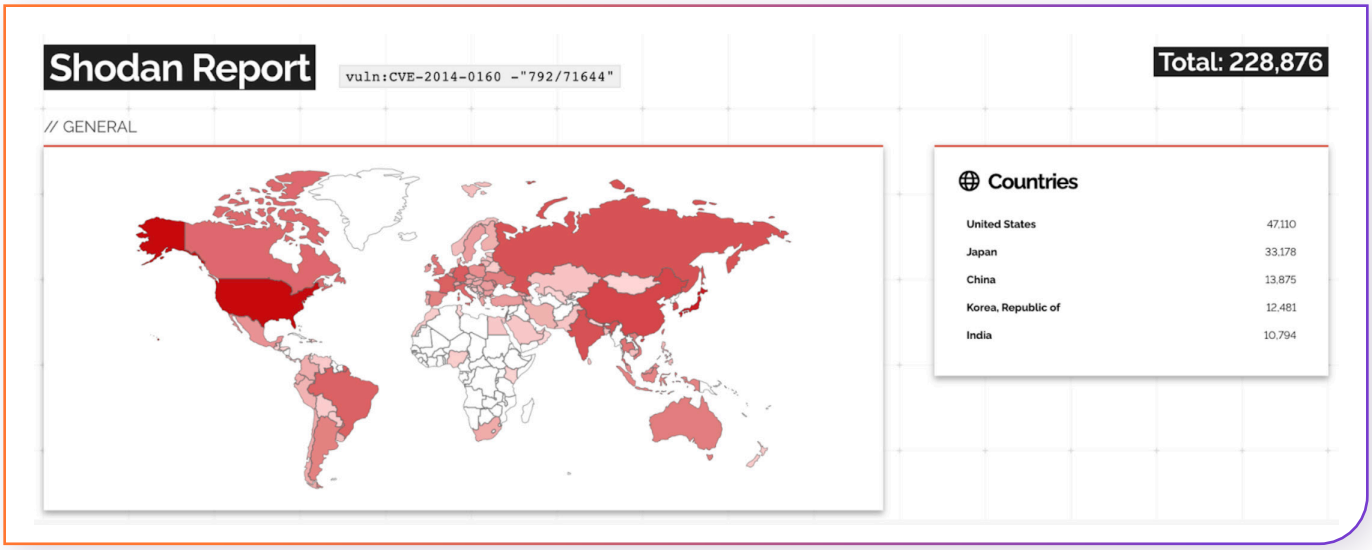
What is it?

The vulnerability, made public in April 2014, affects the heartbeat extension (RFC 6520) implemented in OpenSSL 1.0.1 through 1.0.1f can result in the leak of memory contents from the server to the client and from the client to the server when exploited.

Effectively, allowing anyone on the internet to read the memory of the systems using the vulnerable versions of the OpenSSL software.

The fact that exploitation of the vulnerability isn't very complex as well as leaves no visible trace on the attacked machine, has made the HeartBleed vulnerability one of the most impactful vulnerabilities in recent years.

Potential Vulnerable Internet-Facing Applications According to Shodan.io — over 228,000

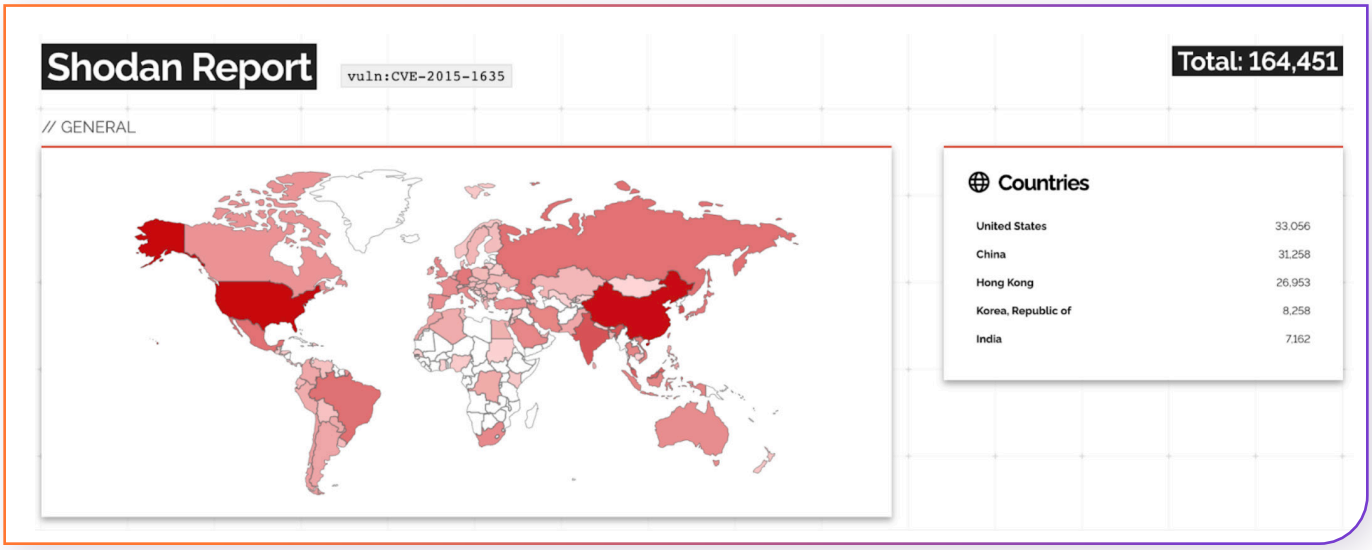


CVE-2015-1635 — Microsoft HTTP.sys Remote Code Execution Vulnerability
CVSS2 — 10
Known to be exploited in the wild — Yes
Vulnerability Age: **7.5 Years**

What is it?

CVE-2015-1635 (also known as MS15-034) is a remote code execution vulnerability in the HTTP protocol processing module (HTTP.sys) in Microsoft Internet Information Service. The vulnerability could allow remote code execution by sending a specially crafted HTTP request to an affected Windows system.

Potential Vulnerable Internet-Facing Applications According to Shodan.io — over 164,000



CVE-2018-13379 — Fortinet FortiOS and FortiProxy
CVSS3— 9.8
Known to be exploited in the wild — Yes
Vulnerability Age: **4.5 Years**

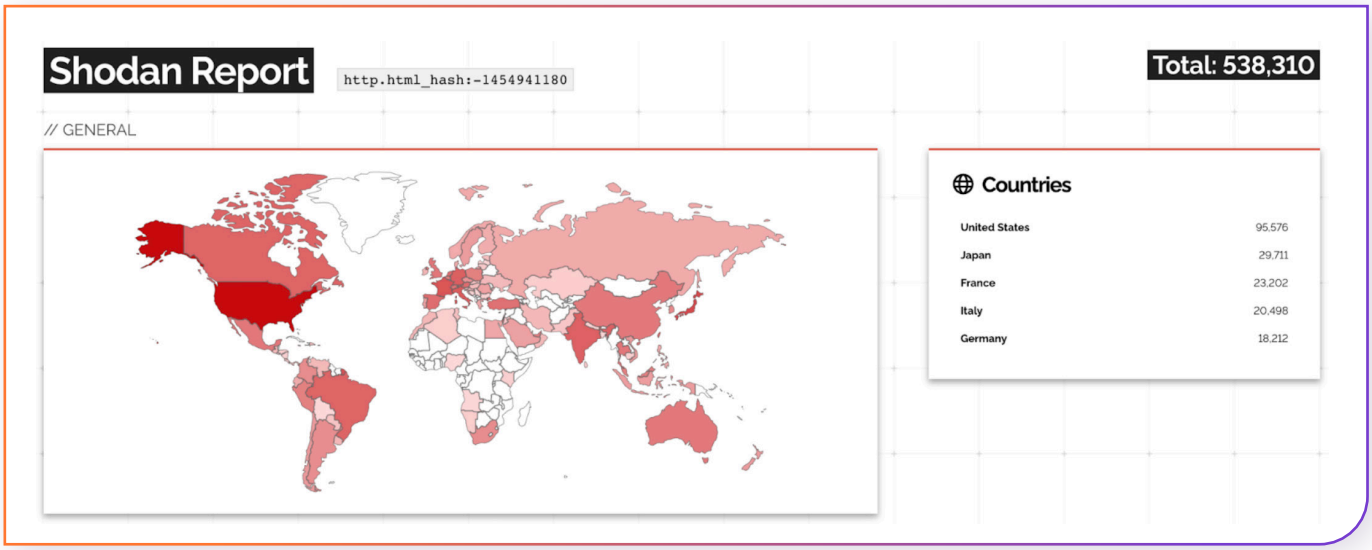
What is it?

Over four years in the wild and still a recurring member of the top most routinely exploited vulnerabilities, CVE-2018-13379 is a path traversal vulnerability in the FortiProxy SSL VPN web portal. Once exploited, the bug may allow a non-authenticated, remote attacker to download FortiProxy system files through specially crafted HTTP resource requests.

It has a long and storied history, as expected from a vulnerability that has been exploited for over 4 years, and has been utilized to steal data and deploy ransomware. CISA has released several advisories over the years detailing its use by both Iranian and Russian state actors to gain access to multiple government, commercial, and technology services networks.

As recently as February 2022, SentinelLabs reported Iranian-aligned threat actor TunnelVision making use of CVE-2018-13379, along with other more recent vulnerabilities such as Log4Shell and ProxyShell, to target organizations.

Potential Vulnerable Internet-Facing Applications According to Shodan.io — over 538,000



Active exploitation attempts in the last 30 days according to Greynoise.io: **162** (unique IPs)

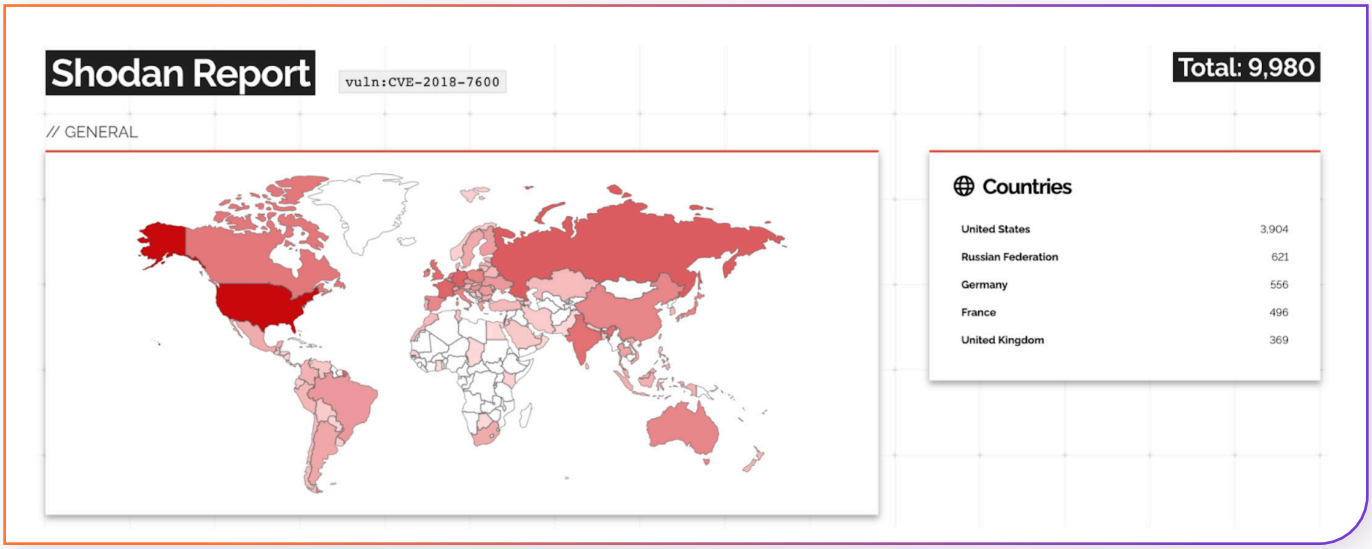
CVE-2018-7600 ("Drupalgeddon2") — Drupal remote code execution vulnerability
CVSS3— 9.8
Known to be exploited in the wild — Yes
Vulnerability Age: **4.5 Years**

What is it?

Drupalgeddon 2 is a remote code execution vulnerability affecting Drupal versions 7.x before 7.58, 8.3.x versions before 8.3.9, 8.4.x versions before 8.4.6, and 8.5.x before 8.5.1.
The vulnerability, which does not require authenticated access, can be used by an attacker to force the server running Drupal to execute malicious code that could completely compromise the Drupal installation and in specific configurations potentially compromise the host machine as well.

Upon discovery of the vulnerability, it was estimated that over 1 million sites were vulnerable to CVE-2018-7600.

Potential Vulnerable Internet-Facing Applications According to Shodan.io — over 9980



Active exploitation attempts in the last 30 days according to Greynoise.io: **86** (unique IPs)

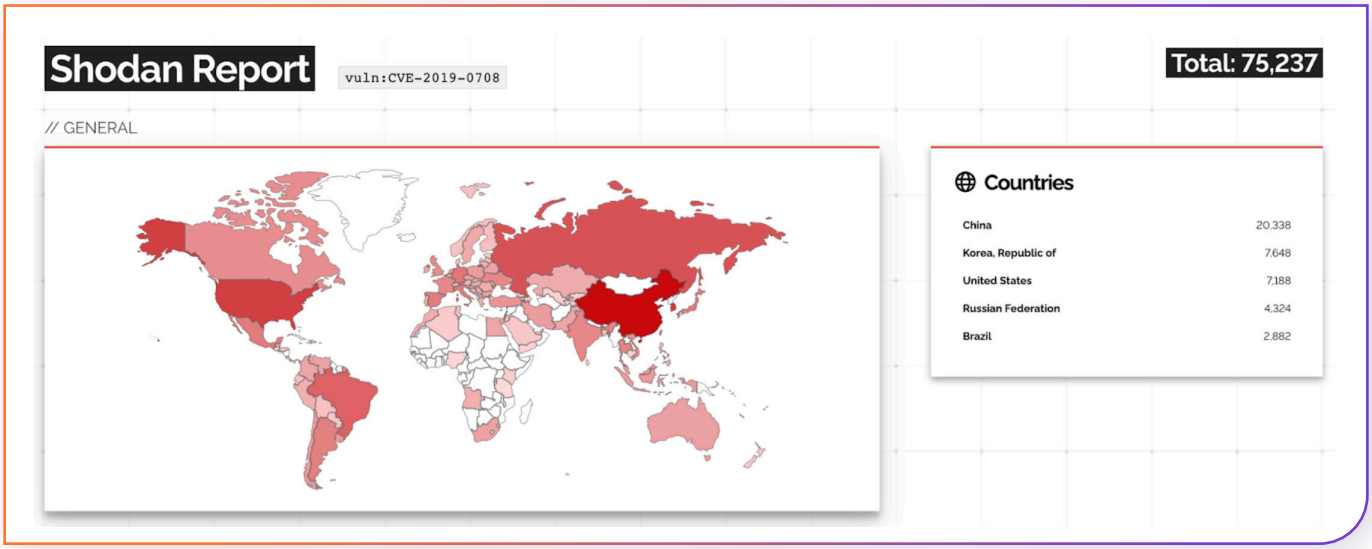
CVE-2019-0708 ("BlueKeep") — A remote code execution vulnerability in Remote Desktop Services (RDP)
CVSS3— 9.8
Known to be exploited in the wild — Yes
Vulnerability Age: **3 Years**

What is it?

BlueKeep is a critical RCE vulnerability in RDP discovered by the UK National Cyber Security Center in May 2019. The vulnerability affected all unpatched Windows NT-based versions of Microsoft Windows from Windows XP, Windows 2000 through Windows Server 2008 R2 and Windows 7. Windows XP has received an out-of-band update from Microsoft due to its potential severity, evn though it was already in an End of Life state at the time.

The potentially "wormable" vulnerability was even addressed by a special NSA advisory urging organizations to patch vulnerable versions as soon as possible.

Potential Vulnerable Internet-Facing Applications According to Shodan.io — over 75,000



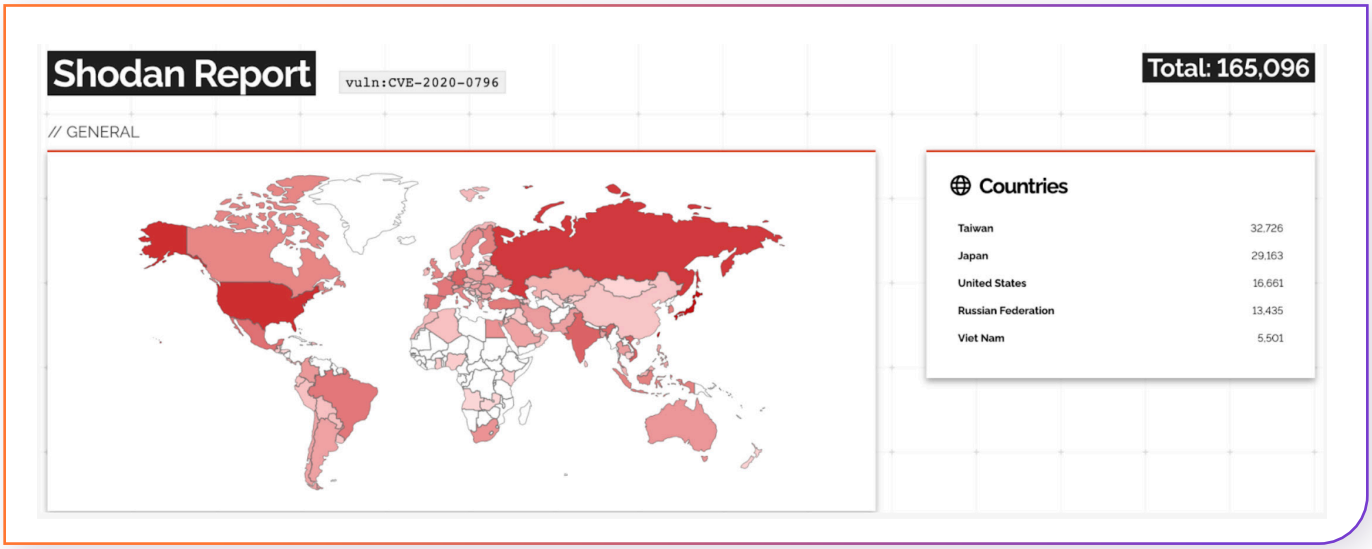
CVE-2020-0796 ("SMBGhost") — A remote code execution vulnerability in Microsoft Server Message Block 3.1.1 (SMBv3)
CVSS3— 10
Known to be exploited in the wild — Yes
Vulnerability Age: **2.5 Years**

What is it?

The youngest member of our list, CVE-2020-0796, is a bug in the compression mechanism of SMBv3.1.1. The vulnerability, first publicly reported in March 2020, affects Windows 10 versions 1903 and 1909.

Initial analysis has classified the vulnerability as an unauthenticated remote DoS vulnerability, though an exploit code released by a security researcher three months later has demonstrated how the vulnerability can be exploited to achieve unauthenticated Remote Code Execution.

Potential Vulnerable Internet-Facing Applications According to Shodan.io — over 165,000



Unique IPs scanning for the SMBGhost vulnerability in the last 30 days according to Greynoise.io: **31**

Conclusion

ALL OF THE VULNERABILITIES ANALYZED IN THIS RESEARCH HAVE BEEN AROUND FOR YEARS, they all have patches released, and they are all known to be exploited in the wild.

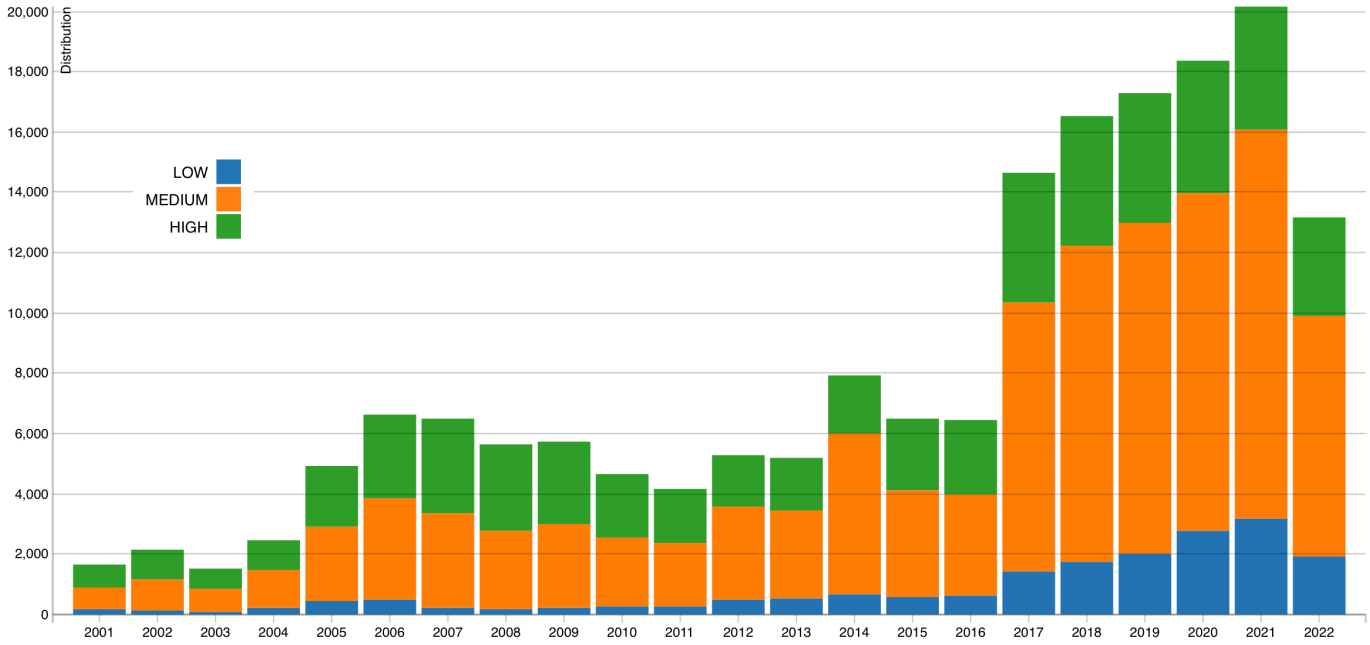
Yet, as we have demonstrated, their attack surface remains huge. Why is that?

It is no secret that effective patching takes time, and a considerable amount of effort.

With the constant rise in the number of vulnerabilities discovered each year and the lack of skilled professionals, organizations simply struggle to keep up.

CVSS SEVERITY DISTRIBUTION OVER TIME

This visualization is a simple graph which shows the distribution of vulnerabilities by severity over time. The choice of LOW, MEDIUM and HIGH is based upon the CVSS V2 Base score. For more information on how this data was constructed please see the NVD CVSS page.



CVSS Severity Distribution Over Time (Source: NVD)

While 2021 showed a record-breaking 20,157 new vulnerabilities added to the National Vulnerability Database, according to the CyberSeek Global Security Heatmap there are currently over 700,000 unfilled cybersecurity positions in the US alone.

Another factor that can explain this patching gap is the fact that patching often introduces instability. While vendors try to check their software to the best of their ability, not all interoperability edge cases can be tested. Often security updates are coupled with product functionality which at times could lead to unforeseen patch compatibility issues.

This leads to a mindset of "don't fix it unless it's broken", especially with less security-mature organizations.

Lastly, less mature organizations often lack visibility into their actual attack surface. Without the proper tooling and vulnerability management processes in place, they are basically blind to the risk, and an asset that you don't know is vulnerable isn't likely to get patched.

So What Can Be Done?

FIRST, AT THE RISK OF STATING THE OBVIOUS, a vulnerability management function is a must!

Make sure you have visibility into your existing attack surface (CVEs) as well as the ability to identify vulnerable components in your environment associated with the vulnerabilities which require patching. In this context, the importance of having an actionable Software Bill of Materials (SBOM) cannot be stressed enough.

Effective patching requires having supporting processes in place. Change control, testing, and quality assurance are all needed to account for possible compatibility issues.

Once you cover the basics, make sure your vulnerability management efforts can scale.
As the upward trend in the number of vulnerabilities discovered isn't expected to change, to be able to scale your patching efforts automation and intelligent prioritization are needed.

Given all of the above, and especially the bigger and more complex the organization, assuming all systems are going to be up-to-date all the time is impractical. Since you can't patch everything, you need to be able to prioritize the patches that matter most.

Prioritizing by CVSS alone doesn't tell the whole story. You should strive to take a risk-based approach (identify and prioritize high-risk vulnerabilities over minor flaws) in deciding what are the critical patches for your environment. This can be done by looking at exploitation in the wild data such as the CISA known exploited vulnerabilities list, or other threat intelligence sources, by determining whether the vulnerable software elements are even running in your environment, or by taking into account other mitigation or compensating controls you have in place.

Lastly, put an emphasis on continuous monitoring and assessment. In today's environment in which reliance on third-party code (whether commercial or open-source) is inevitable, we have seen multiple occasions where vulnerable code that was already patched is being pulled back into production environments by CI/CD processes.

When you rip a pair of jeans, you can adhere a patch to the rip with a little time and sewing skills. The same applies to software patching, with the proper tools, time, and research, a patch can be applied to prevent cybercriminals from attacking. Vintage vulnerabilities are still in style, but if we as an industry make patching cool again, we can make vintage vulnerabilities last season's fashion trend.