Open source software: determining the real risk posed by vulnerabilities

Steve Mansfield-Devine

Steve Mansfield-Devine, editor, *Network Security*

Open source software (OSS) still has a little trouble in shrugging off its 'alternative' image and that could be making many organisations vulnerable. The fact is that, far from being the concern only of sandal-wearing digital activists, OSS underpins a high proportion of most corporate software development. And, as Chris Fearon, head of Black Duck Software's Open Source Security Research Group, points out in this interview, if organisations don't take OSS security seriously, they could be making the task of an attacker easier.

Some corporate executives – even those in the IT department – might be surprised to find to what degree their solutions depend on OSS. Fearon explains that this is primarily because the use of OSS isn't so much in the software that people actually see and with which they interact but in the tools and platforms that were used to create that code.

"We often see that when people attempt to track the open source components in their applications, there's twice the amount of open source components in those applications as they think"

"The obvious open source use that we see, especially on front-end applications, would be the development frameworks in various languages – the bootstrap-type languages," he says. "However, the one thing that is often overlooked is the use of third-party libraries – for example, those that are bundled with components. I come from a development background and understand that when using particular components in an application there are various dependencies that are pulled into your products and you don't necessarily understand or review what those open source components are.

"It's very much aligned to the supply model and understanding what software you're actually using. It's all too easy to assume that a third-party library is comprised of that one code base. It's easy to overlook those additional open source components that are required: we often see that when people attempt to track the open source components in their applications, there's twice the amount of open source components in those applications as they think."

Deep within the business

Arguably, it's fairly easy to understand how OSS is being employed for web-based solutions – with languages such as PHP and Ruby, content management systems, databases (MySQL, PostgreSQL) and so on. There have been some well-publicised scares in this arena – indeed, an exploit involving Wordpress plugins or a CMS such as Drupal or Joomla seems to be a weekly occurrence. On the plus side, the security community has become adept and highly active in seeking out and correcting such flaws.

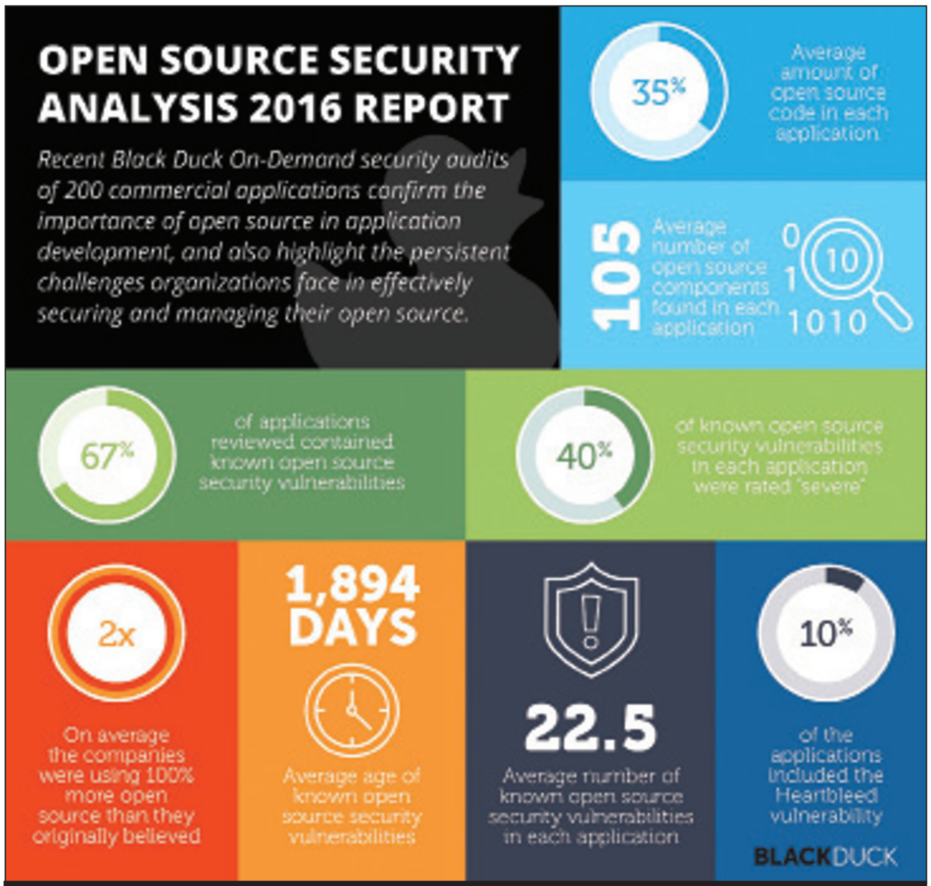
But Fearon points out that the use of OSS in corporate environments is across the board and therefore penetrates much further into the business IT landscape.

In getting to grips with the security implications, it's key to dig deeper and consider the business context in which this software is being used. "There are internal applications that may have a lower threat landscape, as opposed to applications that reside on your DMZ, on your Internet edge," he explains.

"The one thing that's currently missing, or misaligned, is understanding the business implications of the applications, application servers or components that are actually being used"

"While we have become more mature in the open source community in identifying the usage of open source components and understanding the security risks associated, the one thing that's currently missing, or misaligned, is understanding the business implications of the applications,

Chris Fearon, Black Duck Software: "When reviewing these open source applications or components in terms of business risk and risk tolerance within the organisation, that's when we really start to mature the use of open source."



The results of research carried out by Black Duck Software.

application servers or components that are actually being used," he says. You need to understand the role of the solution within the business, he adds, and in gaining that knowledge you will also gain a deeper insight into which applications represent your critical assets.

"When reviewing these open source applications or components in terms of business risk and risk tolerance within the organisation, that's when we really start to mature the use of open source," he says.

Extent of use

So just how extensively is OSS used within businesses? And what kind of security risk does this represent?

Black Duck regularly undertakes audits of business applications, often as part of merger and acquisition activities. Companies need to make sure, as part of their due diligence, that any software they are taking on board as the result of a merger doesn't bring with it an unacceptable level of risk or create licensing problems. The firms may themselves undertake static and dynamic testing of code but that rarely identifies the full range of potential problems.

In its 'Open Source Security Analysis' (OSSA) report for 2016, Black Duck reviewed 200 business applications.[1] Some 95% of these contained OSS components of some kind. The average number of components in each application was 105, 67% of the applications had vulnerabilities in those open source components and 40% of those vulnerabilities were rated as 'severe'. Perhaps most puzzling was the fact that the average age of the OSS vulnerabilities was 1,894 days. In other words, these are likely to be well-known flaws for which potential attackers have had plenty of time to develop exploits. Indeed, 10% of the applications were vulnerable to the infamous Heartbleed bug in the OpenSSL cryptographic library.[2]

Fearon reiterates the fact that even those organisations that make an effort to track what software they're using will typically find, as the result of an audit, that they're employing double the amount of OSS than they assumed. That's a lot more bugs, so having a more accurate picture of your software usage is critical. "Knowing is half the battle in this world," he says.

How did we get here?

It's reasonable to ask how this situation arose. For example, are developers increasingly turning to OSS platforms and libraries to solve their problems?

"Obviously the benefits are hard to ignore with open source software," says Fearon. "With business driving revenue and the time pressures that are on development houses at the moment, with quicker lead times for development and the competitive nature of applications these days, using open source is an absolute requirement. Obviously there are cost benefits associated with it and there's also an opportunity for learning. Because these are open source applications, we can review others' work – it's more open and there's more room for improvement. Generally it just drives improvement across the board."

"It's one thing saying we are allowed, or we approve a particular open source library. You need to understand what burdens those libraries impose"

Of all those, cost and fast delivery are arguably the biggest drivers and many software and development houses would be hard-pushed to survive without using open source. But while the coders themselves almost certainly understand that the library packages they're including in their source code, or the development frameworks in which they're building the project, are open source (and potentially subject to licensing restrictions such as the GPL), does this awareness

Research found that 10% of corporate applications were vulnerable to the Heartbleed flaw.

extend up the management chain, especially once you get to the C level?[3]

"Essentially, I think there's a lack of education, the higher you go," says Fearon. "Obviously the developers are well aware. However, the management team above the developers are really responsible for understanding and creating the open source usage policies that are required. It's one thing saying we are allowed, or we approve a particular open source library. You need to understand what burdens those libraries impose. For example, what is the approval chain for understanding which open source components are applicable to the software that you're building? And obviously there's a business aspect. Once we start using open source, we get faster turnaround times, especially as more businesses adopt agile methodologies and the continuous integration process. Higher up, at the board level, I think there's a lack of awareness of how much open source is actually required in order to deliver that continuous integration environment and the continuous release schedule."

"The supply chain via cloud environments is now a potential attack vector that may not have existed previously. So the way that we're doing business and deploying is really changing the face of application security"

The good news is that there are signs of change – albeit for bad reasons. Breaches are hitting the mainstream news headlines and so now are critical vulnerabilities. Indeed, major flaws are deemed so media-worthy that they get their own branding, such as Heartbleed and Shellshock and sometimes rate having their own websites and logos.[4,5] Fearon believes this is helping to get the message through to the board level that there's something important going on and that there's real business risk associated with the use of open source – as profitable as it might be. This is starting to drive the adoption of tools to get a grip on the situation.

The Dirty Cow flaw in Linux lingered for years before being discovered.

No safety in the cloud

If all of this is making OSS sound insecure, that's unfair. Few, if any, computing platforms, environments, operating systems or applications can be considered completely free of vulnerabilities. The Linux community often likes to portray its favoured OS as the secure alternative to, say, Microsoft Windows. But Linux exploits exist. And OSS aficionados are prone to point to the 'many eyes' principle under which code is purged of flaws because so many people have access to the source code, yet it turns out that much OSS code is maintained by a handful of people – sometimes one person and sometimes no-one, in the case of 'abandonware'. Those who do peruse and contribute to the code often have similar coding backgrounds and so are prone to similar mistakes. And so bugs do slip through, some of which turn out to be exploitable.

"The 'many eyes' argument has been raging for years now," says Fearon. He points to the recent Dirty Cow privilege escalation vulnerability in the Linux kernel.[6] "On the one hand, the original committer made clear with Dirty Cow that he was responsible for the commit that actually resulted in the weakness being discovered years later, but it's surprising to see how long it went without being identified," he says. "Like with Heartbleed etc, the bug had been in the system for a lengthy period of time. But what should be noted is, with the adoption of open source security, researchers are really paying more attention to open source. They are looking for those vulnerabilities in open source components – it's not just developers who may not be fully aware of the impact of certain commits."

Even if a platform itself is reasonably bug-free and secure it's possible for chinks in the armour to open up as a result of how it's delivered. One recent example of this is a flaw found in Microsoft's Azure cloud platform.[7] A software engineer by the name of Ian Duffy discovered a weakness in how instances of Red Hat Enterprise Linux (RHEL) were configured on Azure. This potentially gave him the ability to inject malicious code via RHEL's Yum update service.

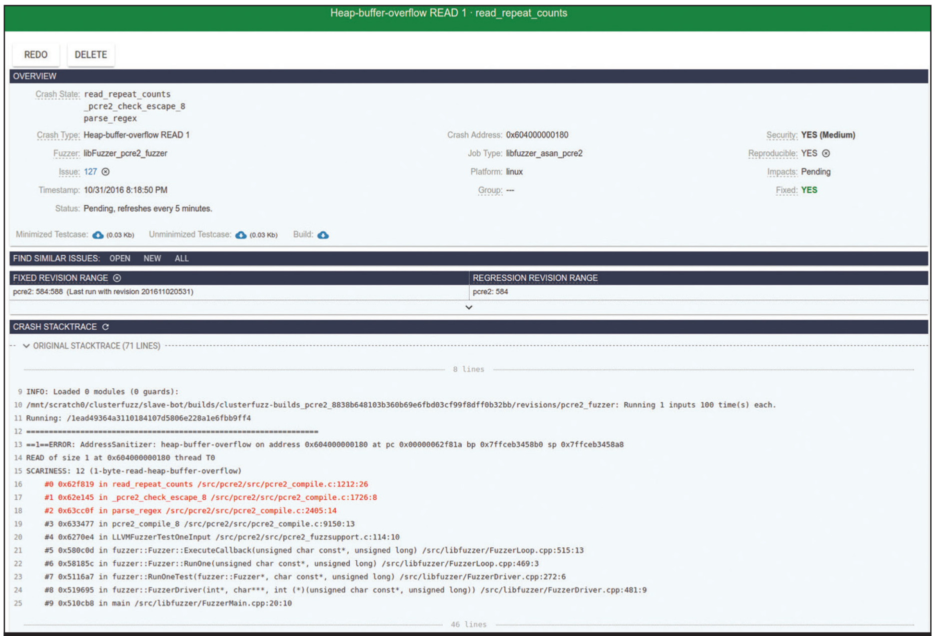
"It boils down to the supply chain," says Fearon. "The supply chain via cloud environments is now a potential attack vector that may not have existed previously. So the way that we're doing business and deploying is really changing the face of application security. We need to take that into the delivery chain and supply model and to build a great appreciation for that in terms of software."

Hot Fuzz

Aside from the attention being paid by security researchers, another promising sign, says Fearon, is the appearance of Google's OSS-Fuzz project.[8] The aim is to continuously test certain OSS projects for security vulnerabilities. It's based on techniques and tools Google developed that revealed hundreds of security vulnerabilities and stability bugs in its own code for the Chrome browser.

"If you look at the life cycle of continuous integration, we're really trying to preempt these weaknesses and get ahead of the game, move ahead of the SDLC phase further up the chain"

Essentially, OSS-Fuzz is a library of tools that a developer can include in a project's build and test system. The code is then submitted to the OSS-Fuzz service and put under scrutiny by ClusterFuzz, a distributed fuzzing environment in which many people can contribute to testing the software.



Heap-buffer-overflow READ 1 · read_repeat_counts

REDO DELETE

OVERVIEW

Crash State: read_repeat_counts
_pcre2_check_escape_8
parse_regex

Crash Type: Heap-buffer-overflow READ 1
Fuzzer: libFuzzer_pcre2_fuzzer
Issue: 127
Timestamp: 10/31/2016 8:18:50 PM
Status: Pending, refreshes every 5 minutes.

Crash Address: 0x604000000180
Job Type: libfuzzer_asan_pcre2
Platform: linux
Group: ---

Security: YES (Medium)
Reproducible: YES
Impacts: Pending
Fixed: YES

Minimized Testcase: (0.03 Kb) Unminimized Testcase: (0.03 Kb) Build:

FIND SIMILAR ISSUES: OPEN NEW ALL

FIXED REVISION RANGE	REGRESSION REVISION RANGE
pcre2: 584:588 (Last run with revision 201611020531)	pcre2: 584

CRASH STACKTRACE

ORIGINAL STACKTRACE (71 LINES)

```
8 lines

INFO: Loaded 0 modules (0 guards):
/mnt/scratch0/clusterfuzz/slave-bot/builds/clusterfuzz-builds_pcre2_8838b648103b360b69e6fbd03cf99f8dff0b32bb/revisions/pcre2_fuzzer: Running 1 inputs 100 time(s) each.
Running: /1ead49364a3110184107d5806e228a1e6fbb9ff4
=================================================================
==1==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x604000000180 at pc 0x00000062f81a bp 0x7ffceb3458b0 sp 0x7ffceb3458a8
READ of size 1 at 0x604000000180 thread T0
SCARINESS: 12 (1-byte-read-heap-buffer-overflow)
    #0 0x62f819 in read_repeat_counts /src/pcre2/src/pcre2_compile.c:1212:26
    #1 0x62e145 in _pcre2_check_escape_8 /src/pcre2/src/pcre2_compile.c:1726:8
    #2 0x63cc0f in parse_regex /src/pcre2/src/pcre2_compile.c:2405:14
    #3 0x633477 in pcre2_compile_8 /src/pcre2/src/pcre2_compile.c:9150:13
    #4 0x6270e4 in LLVMFuzzerTestOneInput /src/pcre2/src/pcre2_fuzzsupport.c:114:10
    #5 0x580c0d in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) /src/libfuzzer/FuzzerLoop.cpp:515:13
    #6 0x58185c in fuzzer::Fuzzer::RunOne(unsigned char const*, unsigned long) /src/libfuzzer/FuzzerLoop.cpp:469:3
    #7 0x5116a7 in fuzzer::RunOneTest(fuzzer::Fuzzer*, char const*, unsigned long) /src/libfuzzer/FuzzerDriver.cpp:272:6
    #8 0x519695 in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) /src/libfuzzer/FuzzerDriver.cpp:481:9
    #9 0x510cb8 in main /src/libfuzzer/FuzzerMain.cpp:20:10

46 lines
```

Example of a report for software that has been tested using Google's OSS-Fuzz. Source: Google.

"Obviously it's just been released and it's specific to specific projects," says Fearon. "However, if you look at the life cycle of continuous integration, we're really trying to pre-empt these weaknesses and get ahead of the game, move ahead of the SDLC [software development life cycle] phase further up the chain. We really want to be more proactive rather than reactive, which is usually the case. So it's absolutely fantastic that we're moving in this direction, especially for the developers. This sort of tool being used by the developers themselves will instil confidence in the consumer base and that will drive the open source community to be more secure in development: that will be a huge benefit going forward."

The closed alternative

Any C-level executive having read this far might be forgiven for thinking that employing OSS is like stepping into a minefield. But is closed source software any better, or does it simply offer up a different class of problem? Fearon points out that even when flaws are identified in closed-source software and have made their way into the Common Vulnerabilities and Exposures (CVE) database, it's hard to know the true scale of the risk they present.[9]

"With closed source, the issue is that we are unable to have any reasonable access to the code base," says Fearon. "Therefore we're unable to review the weakness itself and understand the implications of that weakness. The vulnerabilities that may be identified may not truly reflect the impact of that weakness, purely because there's no access to that software. So where vulnerability researchers today are getting better with responsible disclosure, we're in a situation where, if vendors do not respond appropriately, this information is then released to the general public, as the researchers believe that individuals running this piece of software should be alerted and make reasonable steps to at least mitigate, if they can't remediate, the issue."

"It really comes down to what threat-modelling processes you have internally when selecting open source or closed source. You need to take that into account when understanding the trust boundaries associated with this open source component that is being used compared to this commercial application"

In the past, this has led to situations where security researchers, having responsibly disclosed vulnerabilities to a vendor but frustrated with lack of a response, have revealed flaws that have resulted in organisations and individuals being exposed to attack. Security companies and individual researchers can put only so much pressure on vendors of closed-source software to fix their bugs. And this is true even when it isn't a case of a software Goliath facing down a security researcher David. There have been at least two incidents in which Google released details of flaws in Microsoft software before the latter had been able to fix them, much to Microsoft's fury. For its part, Google claimed a form of public interest defence.

"In the open source world, what we find is that, with many eyes, we can fully understand the vulnerability and the weakness that was introduced and quickly respond to it," adds Fearon. "It allows us to have greater confidence in the actions that we're taking in our operational environment and at least we have awareness of what that risk is."

In isn't a black-and-white picture, though. Fearon points out that, even among closed-source software vendors, their responsiveness, their support and the information they provide vary greatly from firm to firm.

"It really comes down to what threat-modelling processes you have internally when selecting open source or closed source," he says. "You need to take that into account when understanding the trust boundaries associated with this open source component that is being used compared to this third-party or commercial application. You need to really understand your risk tolerance and that's something that the higher level executives should understand. You may have more risk tolerance in your internal network, where it can be tightly controlled. However, your external network is where you need to understand the role of open source and how that is introduced into your applications."

Do it yourself

One option that's always open with open source is to fix the problem yourself, assuming you have the coding skills in-house. Or you can more easily roll back to an earlier version of a library or application than you can with closed source software. With the latter, the reasons for and nature of updates are often somewhat obscure.

"It's entirely dependent on the nature of the application in which the component is being used," says Fearon. "It's also about understanding the vulnerability itself."

He draws a comparison with network management, where practitioners used to fall into the trap of judging their security based on the number of firewall rules they had. With software, it's not about how many patches you implement but the nature of the threat posed by each flaw.

"What we want to understand is what knowledge lies behind the vulnerability and if this vulnerability is something, in terms of the context of our application, that is actually exploitable," he says. "Is the risk that is associated with that vulnerability something we are willing to accept? That knowledge of the business context shapes our understanding of what we really need to do. While we are good at determining the vulnerabilities associated with a component, it doesn't necessarily mean you have to fix it. It's entirely dependent on the mitigating controls that you have in your environment and the risk tolerance level within your business. Not every vulnerability can be reached through normal means and not every vulnerability may require fixing in the context of your application."

Spotting vulnerabilities

To fix a vulnerability, you first have to know it exists. There are easily available vulnerability databases, such as CVE and the US National Vulnerability Database (NVD), which help track the better-understood issues.[10] However, they don't encompass every threat that's out there. Many vulnerabilities are discussed and even traded via less official channels, including the dark web. Fearon believes it's important to develop the capability to monitor all such sources of information.

"You need to begin to build abilities within your organisation to understand the threats that may exist to your business," he says, "and you really need to extend your skill set and understanding as to what vulnerability feeds exist and how we can correlate these vulnerability feeds."

Just as critical, you have to correlate this information with the software you're actually using.

"What you need to understand is your process for determining whether these vulnerabilities are actually associated with open source components that you use," explains Fearon. "Once you understand which vulnerabilities are of interest, you then need to understand the vulnerability at a technical level. There are vulnerability entries [in the databases] with limited information, but in the open source world we have the ability to go and view that code. We really need to become more mature in the publishing of vulnerabilities and understanding the technical details. That will allow the security operation people to quickly identify what they need to do in order to mitigate the issue, but it will also give the DevOps people an understanding of how to remedy this issue quicker. We need to get better, as a community, at communicating the vulnerabilities at a technical level, as opposed to a high-level introduction."

"We really need to become more mature in the publishing of vulnerabilities and understanding the technical details. That will allow the security operation people to quickly identify what they need to do in order to mitigate the issue"

A lot of this will involved skilled people – for example, in integrating tools such as OSS-Fuzz into the software development life cycle. It will also require skilled people with an advanced understanding of the business context who understand that not all vulnerabilities are exploitable. Fearon sees a shift happening from the quantitative to the qualitative.

"We no longer want to see metrics of how many vulnerabilities are in an environment – we want to know what the business risk is to the environment," he says. "That essentially involves businesses like ourselves educating the open source community – and the consumers of that open source – in understanding how we can take steps to determine the business risk. In order to do that, we need to be able to provide actionable information that's associated with those vulnerabilities. For example, is this vulnerability actually accessible in certain configurations?"

Business investment

In achieving this level of insight into the business implications of software, would it help for the business people themselves to invest some effort in the security programme? Fearon says that, increasingly, the users of the applications are getting more involved in the development and delivery of the software, including the release schedule and maintenance. But he feels that more needs to be done to drive home the message that the business depends on this software and it's critical to the success and viability of the organisation.

"This is your application, this is your business," he says. "This is what keeps the lights on. It comes back to threat modelling that application. You want to first decompose the application, understand the trust boundaries and understand the data flow. This is more of an architectural review. You want to determine and rank the threat scenarios that can be used against that application and understand what countermeasures and mitigation strategies you'd have in your network, including firewalls, IDS, WAFs, etc. You start to understand which applications are critical, and that is something that the business really needs to understand at a greater level. It's not good enough just to have a solution for tracking the vulnerabilities. While it's a fantastic step in the right direction, it also needs to be configured and run properly."

It never ends

Just as new vulnerabilities are being discovered all the time and new exploits created for them, so understanding and mitigating against these problems is a never-ending task.

"It's not a silver bullet, it's a continuous process," says Fearon. "You need to understand, initially, the open source usage policy. But then you need to track usage and enforce those policies and monitor for new vulnerabilities. The whole process really aims to improve the efficiency of SOC [security operations centre] activities and the development organisation in terms of what actions should be taken: fix it yourself; do nothing, which may well be a valid case; or upgrade. It allows you to effectively plan your resources."

About the author

Steve Mansfield-Devine is a freelance journalist specialising in information security. He is the editor of Network Security *and its sister publication* Computer Fraud & Security. *He also blogs and podcasts about infosecurity issues at Contrarisk.com.*

References

1. 'Open Source Security Analysis'. Black Duck Software, 2016. Accessed Jan 2017. https://info.blackducksoftware.com/rs/872-OLS-526/images/OSSAReportFINAL.pdf.
2. Heartbleed. Home page. Accessed Apr 2016. http://heartbleed.com/.
3. 'GNU General Public Licence'. Free Software Foundation. Accessed Jan 2017. www.gnu.org/licenses/gpl-3.0.en.html.
4. 'Shellshock (software bug)'. Wikipedia. Accessed Jan 2017. https://en.wikipedia.org/wiki/Shellshock_(software_bug).
5. Möller, Bodo; Duong, Thai; Kotowicz, Krysztof. 'This POODLE bites: exploiting the SSL 3.0 fallback'. OpenSSL, Sept 2014. Accessed Jan 2017. www.openssl.org/~bodo/ssl-poodle.pdf.
6. Dirty Cow, home page. Accessed Jan 2017. https://dirtycow.ninja/.
7. Pauli, Darren. 'Microsoft update servers left all Azure RHEL instances hackable'. The Register, 28 Nov 2016. Accessed Jan 2017. www.theregister.co.uk/2016/11/28/microsoft_update_servers_left_all_azure_rhel_instances_hackable/.
8. OSS-Fuzz, GitHub page. Google. Accessed Jan 2017. https://github.com/google/oss-fuzz.
9. Common Vulnerabilities and Exposures (CVE), home page. Mitre. Accessed Jan 2017. http://cve.mitre.org/.
10. National Vulnerability Database, home page. Accessed Jan 2017. https://nvd.nist.gov/.

How to make SD-WAN secure

Michael Wood, VeloCloud Networks

Michael Wood

With the software-defined wide area network (SD-WAN), lower costs and increased efficiency are the promised payoffs.[1,2] But what about security concerns? Does this additional route into corporate systems open up new vulnerabilities and create fresh opportunities for attackers?

Perhaps there's a tendency to exaggerate, but IT professionals, especially those involved in telecommunications, should always beware of anything that's connected to the Internet, as well as services provided across the Internet. That includes websites, email, cloud-based applications and – of course – WANs.

Perfectly safe

The bad news is that the wild, unfettered Internet can indeed be a dangerous place – it's a good thing we have firewalls, universal threat defence, intrusion prevention systems, heavily encrypted VPNs and endpoint security to protect us. The good news is that the SD-WAN, one of the fastest-growing technologies for connecting branch offices and remote locations, is perfectly safe.

"You can trust SD-WAN to provide the same or even better security as traditional dedicated WAN services such as Multiprotocol Label Switching (MPLS) at a much lower total cost of ownership"

While SD-WAN often routes traffic over the Internet, the underlying technologies are hardened, armoured and fully protected. You can trust SD-WAN to provide the same or even better security as traditional dedicated WAN services such as Multiprotocol Label Switching (MPLS) at a much lower total cost of ownership (TCO).

An SD-WAN, in a nutshell, can be thought of as an overlay architecture that connects enterprise on-premises datacentres, infrastructure-as-a-service (such as those offerings hosted by Amazon Web Services or Microsoft Azure), cloud services (such as software-as-a-service) and remote locations and branch offices.

In some cases, those locations might be already linked by dedicated circuits using carrier-provided services such as MPLS. Those services are usually reliable and secure, offering guaranteed bandwidth and mostly high availability. On the flip side, they are extremely expensive, locked in by contracts and slow to provision new locations or