ELSEVIER

network SECURITY

ISSN 1353-4858 March 2018 www.networksecuritynewsletter.com

Featured in this issue:

How a zero trust approach can help to secure your AWS environment

Today's organisations are increasingly turning to cloud computing providers to offer them the scalability and IT agility necessary to fuel a fast-moving digital and app-based business.

In reality, this means operating complex hybrid IT environments featuring public cloud infrastructure as a service (IaaS) platforms. Yet the notion of shared responsibility is still misunderstood by some, leading to potentially fatal cyber-security gaps. To help ensure maximum protection for your data, systems and users, you need to take an identity-centric, 'zero trust' approach, argues Barry Scott of Centrify.

Full story on page 5…

We need to talk about IDS signatures

Many companies install an IDS to control traffic inside the corporate network. The deep packet inspection (DPI) mechanism lets them collect traffic streams and identify activity by malware.

At the heart of the most common systems are signature sets used for detecting known attacks, developed by network security experts and companies worldwide. However, as Kirill Shipulin of Positive Technology demonstrates, there are ways of disrupting the operation of some IDS systems and then hiding all traces of such activity.

Full story on page 8…

Modelling cyber-attacks: a survey study

To defeat cyber-attacks it's important to understand their characteristics and how they come about. It's also important to comprehend the attackers' objectives.

Understanding the characteristics of attacks is paramount in creating a good security strategy, so attack modelling is important in gaining a perspective on how attacks can be stopped in a co-ordinated manner. Yassine Ayrour, Amine Raji and Mahmoud Nassar provide a comparative study of the state of the art in attack modelling techniques and show how these approaches can help identify attack vectors before they are used.

Full story on page 13…

Cyber-breaches hit twice as hard in past year, says Cisco

Breached businesses lost an average of $500,000 each as a result of cyber-attacks in 2017, according to research from Cisco. This financial hit was the result of lost revenue, customers and opportunities as well as mitigation and remediation costs, the firm said.

Many of the breaches occurred as a result of the increased sophistication of malware, which is weaponising cloud services and evading detection with encryption.

Cisco's 'Annual Cybersecurity Report' reveals that nearly a fifth of UK firms are having to deal with 250,000-500,000

Continued on page 2...

Contents

NEWS

FEATURES

REGULARS

Editorial Office:
Elsevier Ltd
The Boulevard, Langford Lane, Kidlington,
Oxford, OX5 1GB, United Kingdom
Tel: +44 1865 843239
Web: www.networksecuritynewsletter.com

Publishing Director: Bethan Keall

Editor: Steve Mansfield-Devine
E-mail: smd@contrarisk.com

Senior Editor: Sarah Gordon

Columnists: Tim Erridge, Karen Renaud, Colin Tankard

Production Support Manager: Lin Lucas
E-mail: l.lucas@elsevier.com

Subscription Information
An annual subscription to Network Security includes 12 issues and online access for up to 5 users.
Subscriptions run for 12 months, from the date payment is received.

More information: www.elsevier.com/journals/institutional/network-security/1353-4858

Notice
No responsibility is assumed by the Publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made. Although all advertising material is expected to conform to ethical (medical) standards, inclusion in this publication does not constitute a guarantee or endorsement of the quality or value of such product or of the claims made of it by its manufacturer.

12987
Digitally Produced by
Mayfield Press (Oxford) Limited

...Continued from front page

security alerts a day. Many of them are confronting the problem by turning to artificial intelligence, machine learning and other means of automation to bolster their defences, with around a third of companies engaging in at least one of these areas. The technologies are being deployed to learn and detect unusual patterns in encrypted web traffic, cloud services and Internet of Things (IoT) environments. But while many CISOs report that they are now reliant on such systems and are eager to pursue their potential more, they also report frustration at the high level of false positives they generate.

IoT became a hot topic over the past year in other ways, too, according to the report. We saw a massive rise in the number of IoT-based botnets, many of them used for launching distributed denial of service (DDoS) attacks. Strangely – and perhaps worryingly – in spite of the high-profile, headline-grabbing nature of some of these DDoS campaigns, Cisco's study found that only around 13% of organisations consider IoT botnets as an imminent threat.

The past year has seen a significant uptake in encryption technologies; sadly, this has been the case with malicious actors too. Cisco found more than a threefold increase in the use of encrypted communications channels by malware.

The research also found that security is getting more complex, with the scope of breaches expanding. In 2017, a quarter of security professionals said they used products from 11 to 20 vendors, compared with 18% the year before. And around a third of breaches affected more than half of each organisation's systems, compared with 15% in 2016.

The report is available here: http://bit.ly/2oWWPN0.

UK Government launches IoT code

The UK Government has released a code of practice for consumer Internet of Things (IoT) products which, it's hoped, will make them less liable to hacking.

The 'Security by Design' report, published by the Department of Digital, Culture, Media and Sport (DCMS), details 13 steps manufacturers should take to ensure the security of their devices. These include not having default passwords, implementing a vulnerability disclosure policy and ensuring that software or firmware can be updated – all areas where IoT vendors are regularly failing.

The report analyses the current growth in the IoT market, but also charts the rise in associated risks due to the failure of many vendors to consider security as part of the product development lifecycle. As the report points out, the Mirai attack was a wake-up call as to how these devices can be hijacked for malicious purposes.

"Security by design is a fantastic concept when delivered correctly," said Mark James, security specialist at ESET. "It helps the user understand the requirements and encourages them to make the right decisions to ensure that their safety and the safety of others is maintained at all times. One of the biggest issues for the consumer is knowing they need protecting and just as important, understanding what they need protecting from. It's not always easy to get this across, so, if we can implement measures from the ground up to take some of the decisions away from the user and have them 'auto' or 'default', then achieving that security will certainly be much easier."

He added: "For all this to work we have to still maintain the 'plug and play' culture and that could be a stumbling block – ensuring that something is easy to install, reasonably priced and secure at the same time may not be as simple as it sounds."

Other recommendations of the code of practice are that credentials should be stored securely and communications should be encrypted.

"We need a mindset change from consumers to shift their purchasing habits from selecting the cheapest device to choosing the most trusted device," said Matthias Maier, security evangelist at Splunk. "This change will happen as consumers become more educated and savvy about what they select and it's great to see the UK Government pushing understanding further with the launch of this report."

There are more details available here: http://bit.ly/2DbP0qN.

In brief

Was the Olympic attack a false flag?

A recent attack against the Winter Olympics in South Korea may have been a false flag operation, according to several security specialists, although it failed to work. Security firms noted that malware was being directed against individuals within organisations linked to the games, wifi systems were disrupted, media systems were crippled and the official website was taken down for around 12 hours. One piece of malware, dubbed 'Olympic Destroyer', became a key focus for researchers. The origin of the malware was variously attributed to Russia, Iran and China as well as the Lazarus group that is believed to operate under the orders of the North Korea regime, although it appears to be based on Chinese territory. The malware itself contained code strongly associated with Lazarus.

However, fairly quickly all suspicions began to settle on Russia – in particular, the Fancy Bear group – with one anonymous source within the US Government telling the *Washington Post* that the hackers most likely work for the Main Center for Special Technology (GTsST) group within Russia's military intelligence service, the GRU. This is the same group thought to be responsible for the NotPetya campaign. Now Kaspersky Lab has issued a report that claims the Lazarus code was deliberately inserted into the malware as a 'false flag' operation designed to divert attention towards North Korea. There's more information on Kaspersky's SecureList blog: http://bit.ly/2Ij1ocw.

Homeland insecurity

An audit of the US Department of Homeland Security (DHS) has turned up numerous security issues, including the use of old and unpatched software. The department's own Office of Inspector General found that many of the agency's systems – both unclassified systems and those employed in serious national security applications, with classifications as high as 'top secret' – are running old and unsupported operating systems, some of which have not been patched for five years. Three servers – one at DHS headquarters, and two others run by the Coast Guard and the Secret Service – were still running Windows Server 2003.

A total of 64 systems fell short of the standards required for them to be allowed to operate on DHS networks. In many cases, even computers running current versions of operating systems were found to have up to five unpatched vulnerabilities rated as 'critical'. These included two systems that were missing patches dating back to July 2013 and other systems that hadn't been patched following the WannaCry ransomware outbreak last year. Aside from a lack of patching, other problems found included: cached emails that could have been exposed if the machines were compromised; lack of auditing of the Windows Registry, allowing unattributed changes; and anonymous access to shared network drives. The report is vague in places about which agencies fell short, although it mentioned that the Federal Emergency Management Agency FEMA) had 15 unclassified systems that lost their authority to operate and DHS headquarters had the second-highest number of vulnerable unclassified systems – a total of seven – on its network. The report is here: http://bit.ly/2p3jLKp.

Biggest-ever DDoS attacks…

Late in February, GitHub achieved the dubious honour of being on the receiving end of the biggest-ever distributed denial of service (DDoS) attack, measured at 1.3Tbps – beating the previous record of 620Gbps using the Mirai botnet, which was aimed at security researcher and blogger Brian Krebs. However, the new record was broken five days later when Arbor Networks reported that a US service provider had come under a 1.7Tbps attacks. Neither attack caused much disruption because the firms had implemented DDoS mitigation protections.

Both attacks exploited memcached server amplification techniques. Memcached is a database server technology using memory-based caching of data to increase performance. There are no security features in memcached because it was never meant to be Internet-facing. However, it turns out that a number of organisations have been using the technology with web-connection servers. And a flaw has been discovered in which a small, carefully crafted UDP packet can elicit a large response from the server. By faking the source IP address of the UDP packet – to match that of the target – an attacker can generate huge amounts of data to flood a victim's systems. A 203-byte request can result in a 10MB response – an amplification factor of over 50,000. Security firms providing DDoS protection, including Arbor, Akamai and Cloudflare, say there has been a sharp increase recently in memcached server attacks and they estimate there are around 88,000 misconfigured servers available to attackers. Security firm Corero has warned that this vulnerability could also lead to attackers being able to retrieve, modify and delete data on the servers. As well as locking down servers using memcached technology, one simple mitigation technique is to block UDP traffic on port 11211.

…and the first IPv6 attack

Security firm Neustar has intercepted what it claims is the first distributed denial of service (DDoS) attack launched entirely using IPv6 protocols. The attack came from around 1,900 native hosts on more than 650 networks and targeted Neustar's authoritative DNS service. Barrett Lyon, head of research and development at Neustar, said: "We've been monitoring the increasing deployment of IPv6 for a while now and have seen certain indicators of it hitting critical mass. This attack was, however, the first actionable attempt from hackers. Businesses now need to treat IPv6 as a first-class citizen, as well as an important part of their security profile." Organisations implementing software that uses network connectivity have been advised to write code with the ability to call protocol-agnostic networking libraries, which means that in cases where the software doesn't need to consider whether it is on an IPv4 or IPv6 network, it will use whatever is available and preferred by the network. This, claims Neustar, has encouraged those that write bots and worms to follow the same practice.

Ransomware remains a major menace

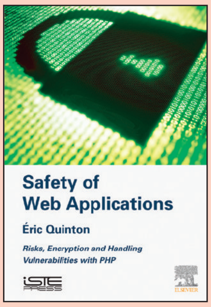
Research by CyberEdge reveals that more than half (55%) of organisations worldwide were the victims of ransomware in the past year. Of these, more than half (53%) refused to pay up and recovered their data anyway (presumably from back-ups), and 8% didn't pay and lost data. Of the roughly two-fifths who paid the ransom, only half actually got their data back. Contrary to the popular image, it's not UK and US firms that are the major targets – Spain, China and Mexico are the most affected; the US comes ninth. Mid-sized enterprises are the most common targets and the key industries affected are education, telecoms and technology, and manufacturing (with about 60% of organisations impacted), followed by retail (51%), finance (50%), government (50%) and healthcare (44%). The report is here: http://bit.ly/2HqwBZY.

Consumer tech in the workplace

As enterprises continue to adopt an increasing number of cloud applications, many are considering allowing employees to use consumer-focused authentication methods in the workplace for ease of access to company resources. That's according to Gemalto's '2018 Authentication and Identity Management Index' report, which claims that while 92% of businesses are concerned about employees using personal credentials to access corporate resources, 70% believe that consumer authentication could be applied in the workplace. Over half (54%) of IT leaders believe that the authentication methods they implement in their businesses are not as good as those found on popular sites, including Amazon and Facebook. In this rush to adopt this technology, though, security is taking a back seat; 61% of organisations admit that they are failing to implement two-factor authentication. The report is here: http://bit.ly/2HmOhFR.

Reviews

BOOK REVIEW



Safety of Web Applications

Eric Quinton. Published by Iste Press. ISBN: 9781785482281. Price: $130, 224pgs, hardback.
E-book editions also available.

The web represents the biggest attack surface you could possibly imagine. Now that it is so completely integral to our way of life, organisations and individuals can find themselves exposed in all manner of unsavoury ways thanks to the reach and ubiquity of web technology.

It's not just the humble website itself that is at the root of so many vulnerabilities – technologies such as REST APIs, FTP servers and other web-oriented services have the potential to result in data breaches or network intrusion.

An early problem with the web that persists to this day is that many of the people involved in the development side are not necessarily trained programmers, let alone people skilled in security. It's not uncommon for websites to be thrown together by graphic designers who have picked up a few Apache, PHP and MySQL skills – perhaps from a 'how to' book. And while such books are starting to get better at acknowledging security risks – such as not simply concatenating 'username' and 'password' inputs from users in a 'SELECT * FROM' MySQL query – they rarely have the space for a real evaluation of the risks.

Even where a web developer is a professional coder, security awareness is often lacking. Security is a skill in itself and unless a trained practitioner is involved in the development and/or the resulting web application is subject to rigorous penetration testing, it's all too easy for security vulnerabilities to creep in purely out of ignorance of the problem.

You could argue that it's just too easy to create a web application. PHP is one of the most popular technologies for doing this, yet it had very humble origins. Created by Rasmus Lerdorf in 1994, it was originally intended just as a way of easily updating his personal web page (the acronym initially stood for 'personal home page', later becoming the recursive 'PHP: Hypertext Preprocessor'). Lerdorf released it as open source in 1995, but it was always intended as a simple CGI preprocessor for web pages – not as a full-blown programming language. However, its simplicity and adaptability meant that it quickly grew beyond Lerdorf's control.

Today, PHP is a powerful, fully-featured language that, nevertheless, is easy to learn and deploy. Even a newcomer to programming, such as a graphic designer, can quickly have a public-facing website interrogating back-end databases and presenting custom pages on the fly. Its power and ubiquity also mean that PHP remains a firm favourite for the creation of Ajax servers and REST APIs.

With web application technology so readily available (much of it is open source and therefore free) and simple to deploy (hosting services will do that for you), the time and effort you have to expend to have your app running on the Internet is very modest. Now add to that the competitive, 'first to market' ethos of the web and the limited training in and awareness of security among developers and you have a powerful mix that seems designed to create vulnerabilities.

This book is about taking a step back from that 'throw up a website' attitude in that it details a more considered, formal approach in which security is firmly entrenched.

The book – and the process it describes – begins with risk analysis, the proper starting point for any security-related activity. Alas, this isn't a simple process. It depends greatly on who you are and what you're doing – as author Eric Quinton points out: "A banking application is more sensitive than a system for booking meeting rooms". And risk is affected greatly by the technologies you choose to use. This means that risk analysis can end up being an iterative process that continues right through development.

Encryption plays a major role in securing web applications these days – witness Google's push to make every website adopt SSL encryption. Quinton spends an entire chapter on getting your website's encryption properly configured, including how to prevent the site falling back to less-secure encryption standards such as SSL/TLS 1.0.

Another major focus of the book is user authorisation and identity management. The humble login is frequently one of the weakest points of any web application, in spite of the ready availability of technologies such as OAuth and LDAP, which help deal with these issues.

With security, the devil is often in the detail, with vulnerabilities arising in the gaps between technologies or in weird corner cases not foreseen by the developers. But Quinton also points out that you need to look at the overall structure of your solution and he gives a brief but clear overview of the now-ubiquitous model-view-controller (MVC) paradigm for applications.

While PHP is highlighted in the book's sub-heading – 'Risks, encryption and handling vulnerabilities with PHP' – it's only once you're about a third of the way into the book that you really start to see PHP code. Still, there are plenty of code examples of both good and bad practices that will help you understand how pitfalls emerge and how to avoid them.

Although fairly short, this book offers good insights into how security issues arise in web applications. And it presents a strong framework for addressing them, not just in terms of coding practice and the technologies you should deploy, but also in terms of process. This makes the book a valuable source for a wide variety of readers.

Anyone with overall responsibility for a web applications project can use it to define the overall approach and what aspects need to be addressed during the development process. The book will also help them formulate the questions they should be asking of their developers.

Developers themselves could use this book to discover where they may be lacking in security awareness – which parts of their skillbase are incomplete. And graphics designers might learn that they need to hire skilled developers if they want to avoid getting into trouble.

There's more information available here: http://bit.ly/2iVaAZ7.

– SM-D

How a zero trust approach can help to secure your AWS environment

Barry Scott

Barry Scott, Centrify

Today's organisations are increasingly turning to cloud computing providers to offer them the scalability and IT agility necessary to fuel a fast-moving digital and app-based business. In reality, this means operating complex hybrid IT environments featuring public cloud Infrastructure as a Service (IaaS) platforms from providers such as Amazon Web Services (AWS). Yet the notion of shared responsibility is still misunderstood by some, leading to potentially fatal cyber-security gaps. To help ensure maximum protection for your data, systems and users, you need to take an identity-centric, 'zero trust' approach.

Shared responsibility

There's no doubt that IaaS is becoming a more attractive option to global organisations. That's why worldwide spending on the sector was projected to grow by nearly 37% last year to reach $34.6bn.[1] Barracuda Networks estimates that the portion of infrastructure that EMEA organisations are putting in the cloud will rise from 35% in 2017 to 63% by 2022.[2] The benefits are now well understood. Running workloads in the public cloud offers the kind of IT efficiency and flexibility needed to support rapid development and consumption of the applications which increasingly drive the modern enterprise and its customers. Organisations understand that to be competitive today they need to have the agility to respond fast to changing business demands. That means investing in IaaS.

Yet security remains a major perceived barrier to doing so. Is this fair? Well, yes and no. Providers such as Amazon Web Services have advanced tremendously in terms of the security capabilities they're willing and able to provide. But organisations that think they can outsource most or all of their security wholesale to the cloud provider are sorely mistaken.

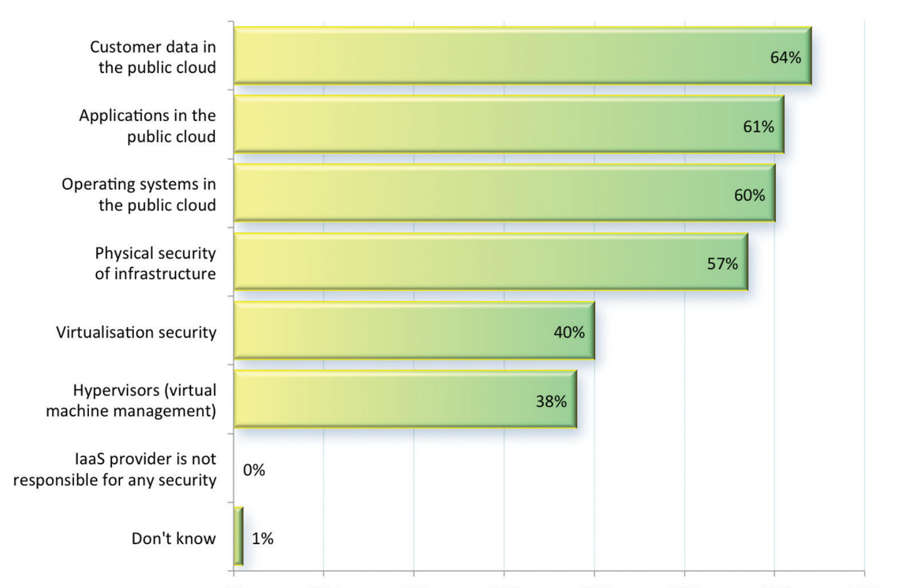


Figure 1: What organisations believe public cloud service providers are responsible for securing. Source: Barracuda.

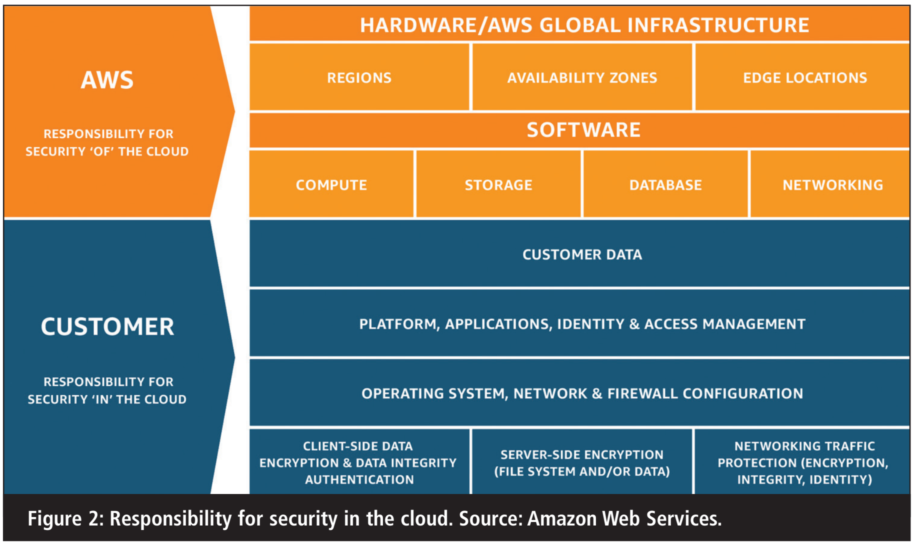
"Customers must address 'security in the cloud' – that is, the data, operating systems, network configuration, applications and the identity and access management"

In fact, AWS spells out very clearly that customers must agree to a shared responsibility model when it comes to security in the cloud.[3] That means AWS will take care of what it describes as 'security of the cloud' – the hardware, software, networking and facilities that comprise the infrastructure which runs AWS services. But customers must address 'security in the cloud' – that is, the data, operating systems, network configuration, applications and the identity and access management (IAM).

Unfortunately, awareness of these responsibilities can be lacking. That same Barracuda Networks poll of EMEA IT leaders last year found that the vast majority thought – erroneously – that their public IaaS provider is responsible for securing customer data (64%), applications (61%) and operating systems (60%).

Identity at the frontline

The security implications are stark. Cybercrime gangs and nation-state operatives are increasingly adept at finding and exposing the holes in your hybrid cloud set-up. The old model of traditional perimeter security – which depended on firewalls, VPNs and



AWS

RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD

HARDWARE/AWS GLOBAL INFRASTRUCTURE

REGIONS | AVAILABILITY ZONES | EDGE LOCATIONS

SOFTWARE

COMPUTE | STORAGE | DATABASE | NETWORKING

CUSTOMER

RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD

CUSTOMER DATA

PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT

OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION

CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION | SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA) | NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)

Figure 2: Responsibility for security in the cloud. Source: Amazon Web Services.

web gateways to separate trusted from untrusted users – is gone. The new cloud and mobile-first world – where employees access the network via their own devices and IoT endpoints proliferate – is a more complex and challenging environment to secure.

This has pushed IAM to the fore: it's now the front line against IaaS threats. Yet attempts to secure AWS environments must take account of the challenges that exist. Static password-based log-ins can be easily cracked, stolen or guessed. Verizon's much-quoted 'Data Breach Investigations Report 2017' reveals that 81% of hacking-linked breaches over the reporting period used stolen and/or weak passwords.[4] Phishing is now used on an industrial scale to unlock corporate accounts *en route* to sensitive data – it was present in over 90% of security incidents and breaches studied in the report.

"The combination of malicious insiders, negligent staff, third-party attackers and complex systems is a perfect storm that creates a high degree of information security risk in AWS environments"

Next up is the problem of too much privilege. Hackers are now increasingly targeting admin accounts in the knowledge that by cracking these open they can get to the organisation's crown jewels – its most sensitive data – quicker and more efficiently. It doesn't help that privileged users are also guilty of poor password management practices such as sharing credentials. In fact, users are at the heart of our story here and are the reason organisations must adopt a zero-trust approach. The combination of malicious insiders, negligent staff, third-party attackers and complex systems is a perfect storm that creates a high degree of information security risk in AWS environments. Siloed identity systems, a fluid user base with employees leaving and joining the organisation on a regular basis, third-party risk in the form of contractors and identity sprawl all serve to further ramp up the risk.

These are no longer theoretical risks. A case in point is Uber, the $6.5bn ride-hailing service which was breached to the tune of 57 million users. The cause? Hackers managed to access a private GitHub site managed by the company where they found and then stole the log-ins for a key AWS account.[5]

The fall-out from that breach and others like it should be well understood by now. The financial and reputational damage facing firms that don't properly secure their IaaS environments could be catastrophic. These penalties will get even more severe when the EU General Data Protection Regulation (GDPR) comes into force on 25 May, bringing with it fines of up to 4% of global annual turnover or £17m, whichever is higher.

A zero-trust approach

The zero-trust approach to security is widely regarded as best practice by analysts and governments. So what is it? Following the devastating breach of federal employee data at the Office of Personnel Management (OPM) – again via stolen passwords – a key committee report explained: "The Zero Trust model centres on the concept that users inside a network are no more trustworthy than users outside a network."[6]

To make it a reality, organisations must focus on four key areas: verifying the user; verifying the device; restricting access and privilege; and ensuring that systems are intelligent enough to learn and adapt over time. In practice, this translates to using multi-factor authentication (MFA) everywhere; extending identity controls to the endpoint via a risk score-based system; enforcing privileged access management; and behavioural analytics to constantly update those risk scores.

Some quick wins

So how can a zero-trust model be applied in AWS environments? The good news is that AWS helps a great deal, by providing a 'Security Best Practices' document, as well as tools and security bootstrapping – such as automatically creating an administrator account with encrypted password when you create a new Windows instance.

In fact, there are aspects of your existing security approach which you can extend into the cloud. The key areas we need to focus on here are securing access to AWS management services, EC2 instances and enterprise apps hosted on EC2.

Best practices could include creating a common security and compliance model across on-premises and cloud resources. Also think about consolidating identities to reduce the siloes and identity sprawl that can increase your attack surface. Rather than local AWS IAM accounts and access keys, use centralised identities like Active Directory and enable federated login.

Accountability is another best practice that can be extended out from the datacentre to the cloud. Shared privileged accounts such as 'ec2-user' and 'administrator' are typically anonymous. Instead, you should demand that users log-in with their individual accounts and elevate privilege as required. Then manage entitlements centrally from Active Directory, mapping roles and groups to AWS roles.

Least privilege should be a well understood quick win. Grant users just enough to complete the relevant task in the AWS Management Console, AWS services, on EC2 instances and for access to hosted apps. Bolster this by implementing cross-platform privilege management for AWS Management Console, Windows and Linux instances.

Auditing is another best practice that can extend to IaaS. Log and monitor authorised and unauthorised user sessions to EC2 instances, associating all activity to an individual and reporting on both privileged activity and access rights. AWS CloudTrails and CloudWatch tools can help with session recording.

"You should demand users log-in with their individual accounts and elevate privilege as required. Then manage entitlements centrally from Active Directory, mapping roles and groups to AWS roles"

The final best practice you should be extending from your datacentre to the cloud is MFA. It's a sure-fire way to thwart phishing and in-progress attacks and drive higher levels of user assurance. Ensure you implement it for AWS service management, on login and privilege elevation for EC2 instances, when checking out vaulted passwords, and when accessing enterprise apps.

Service management access

Aside from these six best practices, there are specific use cases in a hybrid cloud AWS environment that may require closer inspection. One particular area of risk lies with the 'root' AWS billing account, which is secured only with a password and email by default. Given its absolute power, AWS recommends you don't use this for everyday access. But you should also bolster security by applying MFA from a supported provider at the point of check-out. You can also configure AWS MFA to prompt for a second factor of authentication after entering the checked-out password on the AWS login page – this will help guard against brute force attacks.

A final step to mitigate the risk of hackers accessing this powerful root account is to delegate a subset of privileges to accounts tied to individuals. AWS IAM federation is a better option here than creating local AWS IAM user accounts because it enables you to grant existing user identities within your enterprise directory the requisite rights to access any AWS service. This helps prevent identity sprawl and challenges associated with identity duplication and synchronisation.

A federated approach could also mitigate the risk of attackers using Lambda scripts to automatically create a second access key for every IAM user. AWS permits up to two such keys to be enabled simultaneously, but many users are unaware of the second key unless they check. Because many never do, an attacker could log in from anywhere at any time, reusing the same access key over and over.

Privileged access

So vital is EC2 to your IaaS environment that you should consider authentication to specific instances as well as authorisation to perform certain tasks during a log-in session. Avoid yet more identity siloes by extending out your existing enterprise authentication. Third-party broker services can broker AD identities quickly and easily to AWS and even hide the complexity involved when managing multiple sources of identity. Governance and role management should be managed centrally, with admins logging in as themselves via a least privilege model and elevating privileges as required to perform actions relevant to their job function.

For authentication, AWS bootstraps a default admin account for your new instance. But again, these are highly privileged accounts so it's best not to share or use them frequently. Instead, consider using individual enterprise identities to log into EC2 instances. Admins can login directly via SSH with their individual, low-privileged accounts, or via the Shared Password Management portal using their enterprise credentials. Their activities can then be audited via session-recording either at the proxy or host level. The end result is to minimise your attack surface across the entire hybrid Windows and Linux infrastructure.

"Avoid yet more identity siloes by extending out your existing enterprise authentication. Third-party broker services can broker AD identities quickly and easily to AWS and even hide the complexity involved when managing multiple sources of identity"

Once logged-in, user access rights can be managed centrally via a third-party service. Follow best practices by forcing them to log in as themselves with minimal privileges, then you can assign granular privileges as required. If any actions are considered more sensitive than others, MFA can be applied based on an appraisal of risk – eg, where the user is logging in from, how secure his or her device is – and even behavioural profiles learned over time. This dynamic approach will significantly reduce your attack surface.

Access to hosted apps

If you are developing hosted apps in AWS, SAML-enable them for federated SSO via one of the many toolkits out there – eg, C#, Ruby on Rails, Python, PHP. It will improve your security over traditional passwords and allow you to benefit from not having to move or replicate identities. Once again, good third-party IAM providers will then allow you to establish a trust relationship between the app and their centralised identity services, ensuring simple access and management of each.

MFA can also be applied to provide extra identity assurance and combat in-progress attacks, and some services may also support PKI-based authentication for smartcards.

Thanks to the shared responsibility model, there's plenty for organisations to think about from a cyber-security standpoint when investing in IaaS to accelerate digital transformation. Although it's not as simple as flicking a switch, the good news is that there are some quick wins to be had by following industry best practices and adopting a zero-trust approach to identity security.

About the author

Barry Scott is CTO for Centrify EMEA. He has over 25 years of Unix, Windows and Linux experience working for many major organisations across industry verticals in various infrastructure operations and architecture roles. For the past 11 years, he has been helping organisations manage their identity management and auditing challenges, focusing on security, regulatory compliance and operational efficiency, especially using 'AD bridge' technology. Scott's current role is focused around enabling Centrify's customers to use infrastructure they already own – Microsoft's Active Directory – to control, secure and audit heterogeneous systems, mobile devices and applications, and also providing them with a unified identity service across datacentre, cloud and mobile using Centrify's on-premises and cloud-based solutions.

References

1. 'Gartner Says Worldwide Public Cloud Services Market to Grow 18% in 2017'. Gartner, 22 Feb 2017. Accessed Feb 2018. www.gartner.com/newsroom/id/3616417
2. 'Steps to Secure Your Journey to the Public Cloud'. Barracuda, Accessed Mar 2018. http://content.barracuda.com/discover-your-cloud.
3. 'Shared Responsibility Model'. Amazon. Accessed February 2018. https://aws.amazon.com/compliance/shared-responsibility-model/.
4. 'Data Breach Investigations Report 2017'. Verizon. Accessed Mar 2018. www.verizonenterprise.com/verizon-insights-lab/dbir/2017/.
5. Newcomer, Eric. 'Uber Paid Hackers to Delete Stolen Data on 57 Million People'. Bloomberg, 21 Nov 2017. Accessed 1 February 2018. www.bloomberg.com/news/articles/2017-11-21/uber-concealed-cyber-attack-that-exposed-57-million-people-s-data.
6. Mello, John. 'Congress to Bureaucrats: Trust No One'. Tech News World, 20 Sep 2016. Accessed Mar 2018. www.technewsworld.com/story/83910.html.

We need to talk about IDS signatures

Kirill Shipulin, Positive Technology

Kirill Shipulin

The names Snort and Suricata are known to all who work in the field of network security. Web application firewall (WAF) and intrusion detection system (IDS) are two classes of security systems that analyse network traffic, parse top-level protocols and signal the presence of malicious or unwanted network activity. Whereas WAF helps web servers detect and avoid attacks targeted only at them, IDS detects attacks in all network traffic.

Many companies install an IDS to control traffic inside the corporate network. The deep packet inspection (DPI) mechanism lets them collect traffic streams, peer inside packets at the IP, HTTP, DCE/RPC and other levels and identify both the exploitation of vulnerabilities and network activity by malware.

At the heart of both systems are signature sets used for detecting known attacks, developed by network security experts and companies worldwide. The signature sets are the product of numerous individual researchers and companies. Among the vendors are such names as Cisco Talos and Emerging Threats and the open set of rules currently counts more than 20,000 active signatures.

However, as we'll see later in the article, there's a new approach that disrupts the operation of Suricata IDS systems and then hides all traces of such activity.

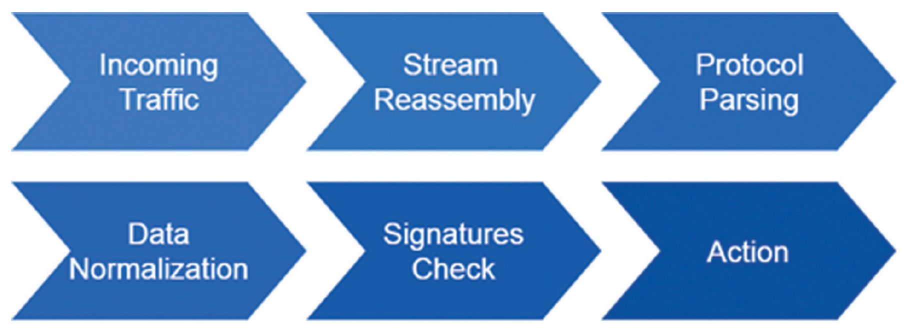


Figure 1: The operating principle behind intrusion detection systems.

How does IDS work?

Before plunging into the technical details of this IDS bypass technique and the stage at which it is applied, let's refresh our concept of the operating principle behind IDS technology.

First, incoming traffic is divided into TCP, UDP or other traffic streams, after which the parsers mark and break them down into high-level protocols and their related fields, normalising them if required. The decoded, decompressed and normalised protocol fields are then checked against the signature sets that detect network attack attempts or malicious packets in the network traffic.

Common evasion methods

IDS flaws and software errors sometimes mean that attacks go unspotted in network traffic. The following are fairly well-known bypass techniques at the stream-parsing stage:

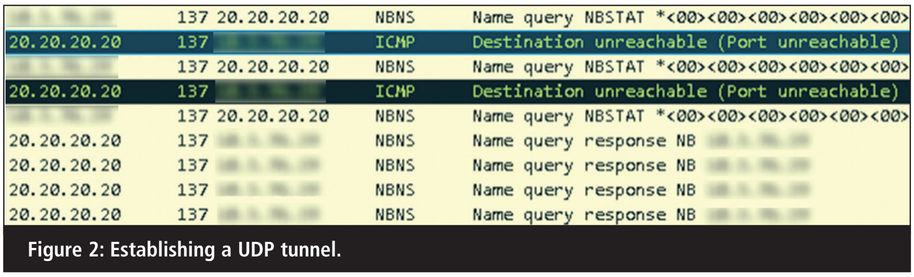
- Non-standard fragmentation of packets, including at the IP, TCP and DCERPC levels, with which the IDS is sometimes unable to cope.
- Packages with boundary or invalid TTL or MTU values can also be incorrectly processed by the IDS.
- Ambiguous overlapping of TCP fragments (TCP SYN numbers) can be handled differently by the IDS than on the server or client for which the TCP traffic was intended. For instance, instead of ignoring it, a TCP FIN dummy packet with an invalid checksum (so-called TCP un-sync) can be interpreted as the end of the session.
- A different timeout time for the TCP session between the IDS and the client can also serve as a tool for hiding attacks.

As for the protocol-parsing and field-normalisation stage, many WAF bypass techniques can be applied to an IDS. Here are just some of them:

HTTP double-encoding.

- A Gzip-compressed HTTP packet without a corresponding Content-Encoding header might remain uncompressed at the normalisation stage; this technique can sometimes be detected in malware traffic.
- The use of rare encodings, such as Quoted-Printable for POP3/IMAP, can also render some signatures useless.

And don't forget about bugs specific to every vendor of IDS systems or third-party libraries inside them, which are available on public bug trackers. One of these specific bugs used to disable signature checks in certain conditions was discovered in Suricata; this error could be exploited to conceal attacks such as BadTunnel.



```
[redacted]      137 20.20.20.20   NBNS   Name query NBSTAT *<00><00><00><00><00><00>
20.20.20.20     137 [redacted]    ICMP   Destination unreachable (Port unreachable)
[redacted]      137 20.20.20.20   NBNS   Name query NBSTAT *<00><00><00><00><00><00>
20.20.20.20     137 [redacted]    ICMP   Destination unreachable (Port unreachable)
[redacted]      137 20.20.20.20   NBNS   Name query NBSTAT *<00><00><00><00><00><00>
20.20.20.20     137 [redacted]    NBNS   Name query response NB [redacted]
20.20.20.20     137 [redacted]    NBNS   Name query response NB [redacted]
20.20.20.20     137 [redacted]    NBNS   Name query response NB [redacted]
20.20.20.20     137 [redacted]    NBNS   Name query response NB [redacted]
```

Figure 2: Establishing a UDP tunnel.

During this attack, the vulnerable client opens an HTML page generated by the attacker, establishing a UDP tunnel through the network perimeter to the attacker's server for ports 137 on both sides. Once the tunnel is established, the attacker is able to spoof names inside the network of the vulnerable client by sending fake responses to NBNS requests. Although three packets went to the attacker's server, it was sufficient to respond to just one of them to establish the tunnel (see Figure 2).

"Researchers investigating network security and network attacks, and developing and testing network signatures first hand, couldn't fail to notice the emergence of bypassing techniques linked to the signatures themselves"

The error was due to the fact that since the response to the first UDP packet from the client was an ICMP packet, for example ICMP Destination Unreachable, the imprecise algorithm meant that the stream was verified with signatures only for ICMP. Any further attacks, including name spoofing, remained unspotted by the IDS, as they were carried out on top of the UDP tunnel. Despite the lack of a CVE identifier for this vulnerability, it led to the evasion of IDS security functions.

The above-mentioned bypass techniques are well known and have been eliminated in modern and long-developed IDS systems, while specific bugs and vulnerabilities work only for unpatched versions. However, researchers investigating network security and network attacks, and developing and testing network signatures first hand, couldn't fail to notice the emergence of bypassing techniques linked to the signatures themselves and their flaws.

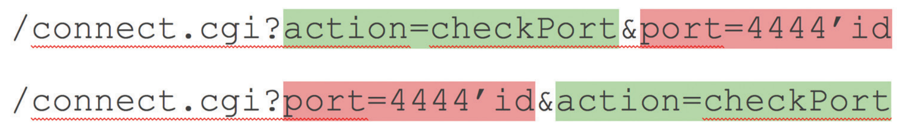
Bypassing signatures

But how can signatures be a problem? Researchers study emerging threats and form an understanding of how an attack can be detected at the network level on the basis of operational features or other network artefacts and then translate the resulting picture into one or more signatures in an IDS-friendly language. Due to the limited capabilities of the system, or researcher error, some methods of exploiting vulnerabilities remain undetected.

If the protocol and message format of a particular family or generation of malware remain unchanged and the signatures for them work just fine, then when it comes to exploiting vulnerabilities, the more complex the protocol and its variability, the simpler it is for the attacker to change the exploit with no loss of functionality – and bypass the signatures.

"To cover the signatures of all attack variations and develop not only high-quality but speedy signatures, the developer must possess wide-ranging skills and a solid knowledge of network protocols"

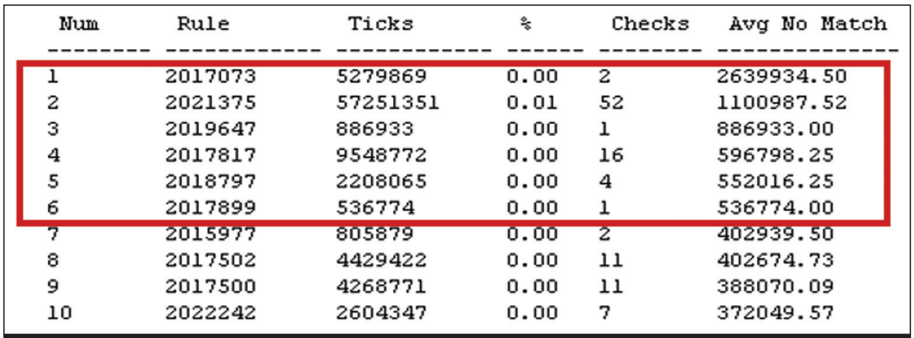
Although you can find many decent signatures from different vendors for the most dangerous and high-profile vulnerabilities, other signatures can be evaded by simple methods. Figure 3 shows an example of a very common signature error for HTTP: at times it's enough just to change the order of the HTTP GET arguments to bypass a signature check.



```
/connect.cgi?action=checkPort&port=4444'id

/connect.cgi?port=4444'id&action=checkPort
```

Figure 3: Changing the order of GET arguments to bypass an HTTP signature check.



```
   Num       Rule          Ticks         %       Checks    Avg No Match
  --------  ------------  ------------  ------  --------  --------------
  1         2017073       5279869       0.00    2         2639934.50
  2         2021375       57251351      0.01    52        1100987.52
  3         2019647       886933        0.00    1         886933.00
  4         2017817       9548772       0.00    16        596798.25
  5         2018797       2208065       0.00    4         552016.25
  6         2017899       536774        0.00    1         536774.00
  7         2015977       805879        0.00    2         402939.50
  8         2017502       4429422       0.00    11        402674.73
  9         2017500       4268771       0.00    11        388070.09
  10        2022242       2604347       0.00    7         372049.57
```

Figure 4: Suricata log showing the slowest signatures at the top.

And you'd be right to think that substring checks with a fixed order of arguments are encountered in signatures – for example, '?action=checkPort' or 'action=checkPort&port='. All that's needed is to carefully study the signature and check whether it contains such hardcode.

Some other equally complex checking protocols and formats are DNS, HTML and DCERPC, which all have extremely high variability. Therefore, to cover the signatures of all attack variations and develop not only high-quality but speedy signatures, the developer must possess wide-ranging skills and a solid knowledge of network protocols.

The inadequacy of IDS signatures is old hat, and you can find plenty of other opinions in various reports.[1-3]

How much does a signature weigh?

As already mentioned, signature speed is the developer's responsibility and, naturally, the more signatures, the more scanning resources are required. The 'golden mean' rule recommends adding one CPU per thousand signatures or 500Mbps network traffic in the case of Suricata.[4] It depends on the number of signatures and volume of network traffic. Although this formula looks good, it leaves out the fact that signatures can be fast or slow and traffic can be extremely diverse. So what happens if a slow signature encounters bad traffic?

Suricata is able to log data on the performance of signatures. The log gathers data on the slowest signatures and generates a list specifying execution time in ticks – CPU time and number of checks performed. In Figure 4, the slowest signatures are at the top.

The highlighted signatures are described as slow. The list is constantly updated and so different traffic profiles would be sure to list other signatures. This is because signatures generally consist of a subset of simple checks, such as searching for a substring or regular expression arranged in a certain order. When checking a network packet or stream, the signature checks its entire contents for all valid combinations. As such, the tree of checks for one and the same signature can have more or fewer branches and the execution time will vary depending on the traffic analysed. One of the developer's tasks, therefore, is to optimise the signature to operate on any kind of traffic.

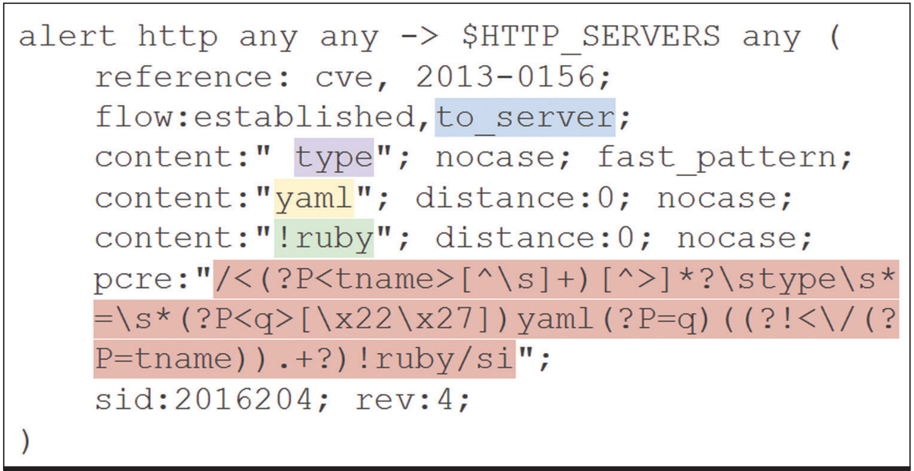
What happens if the IDS is not properly implemented and not capable of checking all network traffic? Generally, if the load on CPU cores is on average more than 80%, it means the IDS is already starting to skip some packet checks. The higher the load on the cores, the more network traffic checks are skipped and the greater the chances that malicious activity will go unnoticed.

> ***"What if an attempt is made to increase this effect when the signature spends too much time checking network packets? Such an operating scheme would sideline the IDS by forcing it to skip packets and attacks"***

What if an attempt is made to increase this effect when the signature spends too much time checking network packets? Such an operating scheme would sideline the IDS by forcing it to skip packets and attacks. So, we already have a top list of hot signatures on live traffic, and we'll try to amplify the effect.

Let's operate

In Figure 5, you can see that one of these signatures reveals an attempt in the traffic to exploit the vulnerability CVE-2013-0156 RoR YAML Deserialisation Code Execution. All HTTP traffic directed to corporate web servers is checked for the presence of three strings in the strict sequence – 'type', 'yaml',

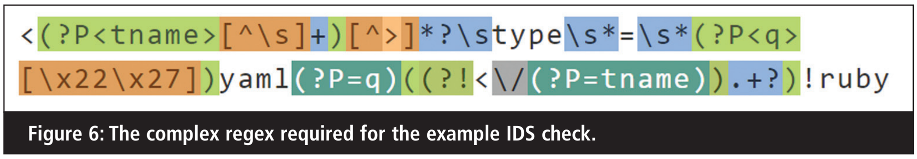


```
alert http any any -> $HTTP_SERVERS any (
    reference: cve, 2013-0156;
    flow:established,to_server;
    content:" type"; nocase; fast_pattern;
    content:"yaml"; distance:0; nocase;
    content:"!ruby"; distance:0; nocase;
    pcre:"/<(?P<tname>[^\s]+)[^>]*?\stype\s*
    =\s*(?P<q>[\x22\x27])yaml(?P=q)((?!<\/(?
    P=tname)).+?)!ruby/si";
    sid:2016204; rev:4;
)
```

Figure 5: An attempt to exploit a YAML deserialisation code execution vulnerability.

rule_perf.log	Num	Rule	Average Ticks	
	1	2016204	57630.00	
keyword_perf.log	Keyword	Ticks	Checks	Matches
	content	18,765	4	3
	pcre	18,985	1	0

Table 1: Results of an IDS check run using a regular expression.



```
<(?P<tname>[^\s]+)[^>]*?\stype\s*=\s*(?P<q>[\x22\x27])yaml(?P=q)((?!<\/(?P=tname)).+?)!ruby
```

Figure 6: The complex regex required for the example IDS check.

'!ruby' – and checked with a regular expression.

Before we set about generating 'bad' traffic, let's present some hypotheses that might help our investigation:

- It's easier to find a matching substring than to prove there is no such match.
- For Suricata, checking with a regular expression is slower than searching for a substring.

This means that if we want long checks from a signature, these checks should be unsuccessful and use regular expressions. In order to get to the regex check, there must be three substrings in the packet, one after the other. Let's try combining them in the order 'typeyaml!ruby' and running the IDS to perform a check. To construct files with HTTP traffic in Pcap format from the text, we used the Cisco Talos file2pcap tool.[5] The results are shown in Table 1.

Another log, keyword_perf.log, helps us see that the chain of checks successfully made it (content matches = 3) to the regular expression (PCRE) and then failed (PCRE matches = 0). If later we want to benefit from resource-intensive PCRE checks, we need to completely parse it and pick out some effective traffic. The complex regex is shown in Figure 6.

The task of reverse parsing a regular expression, although easy to do manually, is poorly automated due to such constructions as back references or named capture groups: we didn't find any methods at all to automatically select a string for successfully passing a regular expression. The following construction was the minimum string required for such an expression.

```
<a type="yaml" !ruby
```

To test the theory that an unsuccessful search is more resource-intensive than a successful one, we'll trim the rightmost character from the string and run the regex again.

```
<a type="yaml" !ruby : 32 steps, match
<a type="yaml" !rub : 57 steps, no match
```

It turns out that the same principle also applies to regular expressions: the unsuccessful check took more steps than its successful counterpart. In this case, the difference was greater than 50%. You can see this for yourself at https://regex101.com/r/51ukhR/1.

Further study of this regular expression produced another eye-opener. If we repeatedly duplicate the minimum required string without the last character, it is reasonable to expect an increase in the number of steps taken to complete the check, but the growth curve is explosive:

```
2 x (<a type="yaml" !rub) : 209 steps
10 x (<a type="yaml" !rub) : 9885 steps
100 x (<a type="yaml" !rub) : timeout
```

The scan time for several dozen such strings is already around one second and increasing their number risks a timeout error. This effect in regular expressions is called 'catastrophic backtracking' and there are many articles devoted to it.[6] Such errors are still encountered in common products; for example, one was recently found in the Apache Struts framework.[7]

Let's take the strings obtained and check them with Suricata. The result is shown in Table 2.

However, instead of catastrophic backtracking, the IDS barely notices the load – only 1 million ticks. This is the story of how after debugging and examining the Suricata IDS source code and the libpcre library used inside it, we stumbled on these PCRE limits:

MATCH_LIMIT DEFAULT = 3500
MATCH_LIMIT_RECURSION_
DEFAULT = 1500

These limits save regular expressions from catastrophic backtracking in many regex libraries. The same limits can be found in WAF, where regex checks predominate. Sure, these limits can be changed in the IDS configuration, but they are propagated by default and changing them isn't recommended.

Network packet

Using only a regular expression won't help us achieve the desired result. But what if we use the IDS to check a network packet with the content 'typeyaml!ruby typeyaml!ruby'? In this case, we get the log values shown in Table 3.

There were four checks, which became seven only because of duplication of the initial string. Although the mechanism remains unclear, we should expect the number of checks to snowball if we further duplicate the strings. In the end, we got the 1,508 checks and 1,507 matches for content and 1,492 with no matches for PCRE.

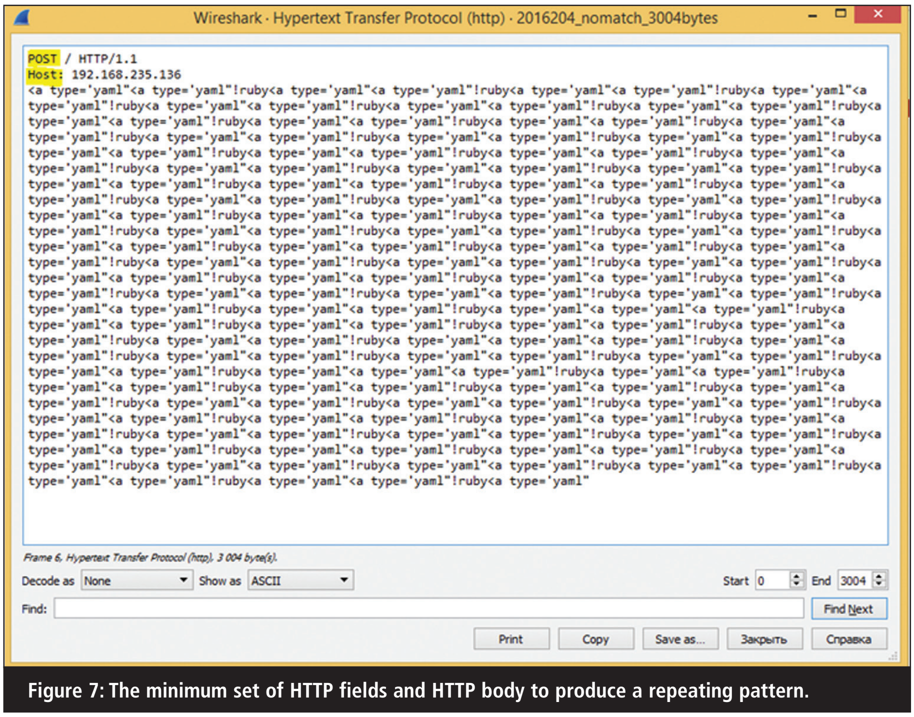
In total, the number of checks of substrings and regular expressions does not exceed 3,000, no matter what content is checked by the signature. Clearly, the IDS itself also has an internal limiter, which goes by the name of the 'inspection-recursion limit', set by default to that

Keyword	Ticks	Checks	Matches
content	19,135	4	3
pcre	1,180,797	1	0

Table 2: Result of a check with Suricata.

Keyword	Average Ticks	Checks	Matches
content	3,338	7	6
pcre	12,052	3	0

Table 3: Log values from a network check.



Wireshark · Hypertext Transfer Protocol (http) · 2016204_nomatch_3004bytes

```
POST / HTTP/1.1
Host: 192.168.235.136
<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"
```

Frame 6, Hypertext Transfer Protocol (http), 3 004 byte(s).

Decode as: None; Show as: ASCII; Start: 0; End: 3004

Find:

Figure 7: The minimum set of HTTP fields and HTTP body to produce a repeating pattern.

same figure of 3,000. With all the PCRE and IDS limits and restrictions on the one-time size of content being checked, by modifying the content and using snowballing regex checks, you get the result you're after, with 1,587,144 ticks producing 1,492 checks and no matches for PCRE.

Although the complexity of one regex check has not changed, the number of such checks has shot up. Multiplying the number of checks by the average number of clock cycles spent on each check, we get the coveted figure of three billion ticks. That's more than a thousand-fold increase! The operation requires only the curl utility for generating the minimum HTTP POST request, which looks something like what's down in Figure 7.

Such content cannot be infinitely large so as to cause the IDS to spend vast resources on checking it, since although inside it the TCP segments are joined in a single stream, the stream and the collected HTTP packets are not checked entirely, no matter how big they are. Instead, they are checked in small chunks of about 3-4KB in size. The size of the segments to be checked, as well as the depth of the checks, is set in config (like everything in the IDS). The segment size 'wobbles' slightly from launch to launch to avoid fragmentation attacks on such segments – when the attacker, knowing the default segment size, splits the network packets so that the attack is divided into two neighbouring segments and cannot be detected by the signature.

Powerful weapon

So, we just got our hands on a powerful weapon that loads the IDS in excess of 3,000,000,000 CPU ticks per utilisation. What does that even mean? The actual figure obtained is roughly 1 second of average CPU operation. Basically, by sending an HTTP request of size 3KB, we load the IDS for a full second. The more cores in the IDS, the more data streams it can process simultaneously.

> ***"The constant flow of malicious traffic can disable the IDS until the traffic stops bombarding the internal network, while for short-term attacks the attacker can send a short spike from such packets and also blind the detection system"***

Remember that the IDS does not sit idle and generally spends some resources on monitoring background network traffic, thereby lowering the attack threshold. Taking metrics on a working IDS configuration with 8/40 Intel Xeon E5-2650 v3 CPU cores (2.3GHz) without background traffic, where all eight CPU cores are 100% loaded, the threshold value turns out to be only 250Kbps. And that's for a system designed to process a multi-gigabit network stream – ie, thousands of times greater.

To exploit this particular signature, the attacker need only send about 10 HTTP requests per second to the protected web server to gradually fill the network packet queue of the IDS. When the buffer is full up, the packets start to bypass the IDS, which is when the attacker can use any tools or carry out arbitrary attacks while remaining unnoticed by the detection systems. The constant flow of malicious traffic can disable the IDS until the traffic stops bombarding the internal network, while for short-term attacks the attacker can send a short spike from such packets and also blind the detection system for a brief period.

Current mechanisms are unable to detect slow signatures: although IDS has a profiling code, the system cannot distinguish a signature that is merely slow from one that is catastrophically slow and automatically signal it. Note that signature triggering is not signalled either, due to the lack of relevant content.

Do you remember the unexplained rise in the number of checks? There was indeed an IDS error that led to an increase in the number of superfluous checks. The vulnerability was given the name CVE-2017-15377 and has now been fixed in Suricata IDS 3.2 and 4.0.[8]

Specific instance

The above approach works well for one specific instance of the signature. It is distributed as part of an open signature set and is usually enabled by default. But new examples keep emerging at the top of the list of hot signatures, while others continue waiting for their traffic.

The signature description language for Snort and Suricata supplies the developer with many handy tools, such as base64 decoding, content jumping and mathematical operations. Other combinations of checks can also cause explosive growth in the consumption of resources.

Careful monitoring of performance data can be a springboard for exploitation. After the CVE-2017-15377 problem was remedied, we again launched Suricata to check our network traffic and saw exactly the same picture: a list of the hottest signatures at the top of the log, but this time with different numbers. This suggests that such signatures – and ways to exploit them – are numerous.

Not only IDS, but also anti-virus, WAF and many other systems are based on signature-search methods. As a result, this approach can be applied to search for weaknesses in their operation. It can stealthily prevent detection systems from doing their job of detecting malicious activity. Related network activity cannot be detected by security tools or anomaly detectors. As an experiment, enable the profiling setting in your detection system – and keep an eye on the top of the performance log.

About the author

Kirill Shipulin is a security researcher at Positive Technologies where he studies malware, network attacks and exploitation techniques to build strong intrusion detection system capabilities to combat modern threats. During his time at Positive Technologies, he has analysed hundreds of security vulnerabilities and malware attacks and has produced IDS detection rules for the majority of famous open source and proprietary products.

References

1. Gula, Ron. 'Bypassing intrusion detection systems'. Black Hat, 2000. Accessed Mar 2018. www.blackhat.com/presentations/bh-usa-00/Ron-Gula/ron_gula.ppt.
2. Coty, Stephen. 'IDS/IPS Signature Bypassing (Snort)'. Alert Logic, 27 Sep 2012. Accessed Mar 2018. www.alertlogic.com/blog/ids/ips-signature-bypassing-(snort)/.
3. 'IDS-Evasion'. Github. Accessed Mar 2018. https://github.com/ahm3d-hany/IDS-Evasion.
4. 'Open-Source Security Tools'. OSsectools, 11 Apr 2011. Accessed Mar 2018. http://ossectools.blogspot.ru/2011/04/network-intrusion-detection-systems.html.
5. 'file2pcap'. Cisco-Talos. Accessed Mar 2018. https://github.com/Cisco-Talos/file2pcap.
6. 'Catastrophic backtracking in regular expressions'. Habra, 4 Nov 2011 (in Russian). Accessed Mar 2018. https://habrahabr.ru/post/131915/.
7. 'S2-050'. Apache, 25 Aug 2017. Accessed Mar 2018. https://cwiki.apache.org/confluence/display/WW/S2-050.
8. 'CVE-2017-15377'. CVE, Mitre, 16 Oct 2017. Accessed Mar 2018. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-15377.

Modelling cyber-attacks: a survey study

Yassine Ayrour

Amine Raji

Mahmoud Nassar

Yassine Ayrour, Amine Raji and Mahmoud Nassar

Computer networks play an important role in many areas of life, which makes them attractive targets for hackers looking to steal sensitive data and cause critical damage. Every day the situation is getting worse; there has been an increase in the number of attacks and their complexity.[1,2] Cyber-attacks are varied and sophisticated, making security analysis more complex. And so research is currently focusing on cyberthreat analysis to learn an attacker's behavioral model and predict the model of attack for any network.[3]

To defeat cyber-attacks it's important to understand their characteristics and how they come about. It's also important to comprehend the attackers' objectives and their means. For example, if attackers are seeking personal satisfaction, they may employ a denial of service to show they can affect a target network. An employee in an accounts department may secretly use legitimate access to issue cheques to collaborators for personal financial gain. So, the possibilities are vast and as varied as the people involved. Understanding the characteristics of attacks is paramount in creating a good security strategy, so attack modelling is important in gaining a perspective on how attacks can be stopped in a co-ordinated manner.

Attack modelling techniques

Knowing a cyber-attack is going to occur before it actually happens is very useful. However, it is very difficult to predict an attack without understanding the vulnerability of the network. In this section, we present attack modelling techniques that can help identify attack vectors before they are used. These techniques can automatically discover all the possible ways an attacker can compromise a network.

Several approaches are proposed, such as attack graph, attack vector, the diamond model, attack surface, OWASP's threat model and kill chain.[4-12] We focus on reviewing the four attack modelling techniques that are mainly used by the research community, namely the attack graph, the diamond model, the kill chain and the attack surface, in order

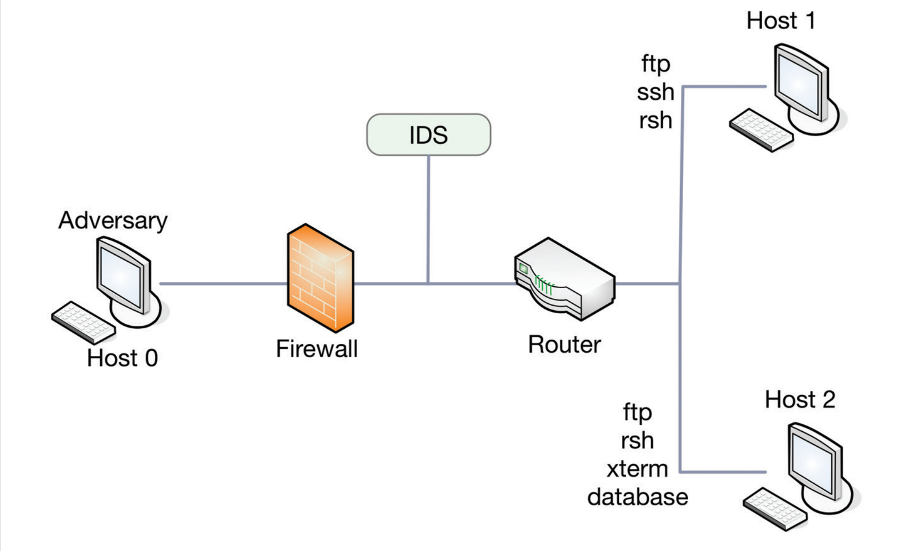


Figure 1: An example of a network from Ghosh.

to understand the process of modelling cyber-attacks.

Attack graph

Attack graphs are abstract representations of the different scenarios and paths an attacker may use to compromise the security of a system by using multiple vulnerabilities. An attack graph uncovers all the possible ways an attacker can compromise an enterprise network. It is a very effective tool for security and risk analysis as it takes into account the relationships between multiple vulnerabilities.[13] The graph represents all possible sequences of the attacker's actions that lead him to certain established goals.[14]

As defined in Kaynar, an attack graph represents possible ways via which a potential attacker can intrude into the target network by exploiting a series of vulnerabilities on various network hosts and gaining certain privileges at each step. In Ghosh, it consists of a number of attack paths that are a logical succession of exploits, each node in the graph representing an attack state and the edge representing a transition of state caused by an action by the attacker. The attacker may need to have a set of privileges on certain hosts in order to exploit a specific vulnerability on a network host. Nodes in the graph are states of the network and arcs are the atomic attacks. Each path from start state to attack state is a series of exploits that leads to the attack.

Attack graphs are mainly used to gather information about types of attacks to which the network is vulnerable and to make decisions such as the set of actions required to stop an adversary. Generating attack graphs is computationally complex and usually takes a long time. For a small network the attack graph can be created quickly; however, when the graph needs to be built for a network that includes hundreds or even thousands of hosts and the result is required in a limited time, the graph-based algorithms require a very large amount of computational resources.

There are a number of attack graph-generating tools and techniques such as Mulval, Tva and Netspa, and a framework was proposed for designing the cyber-attack modelling and impact assessment component that implements the attack graph generation.[17,18] As described, attack graphs help to identify any potential attacks on the network and they predict the various possible ways of penetrating a network to reach critical assets.

Figure 1 shows a network that consists of two internal hosts (host 1 and host 2) that are separated from an external host (host 0) by a firewall. Host 1 is a file server behind the firewall that offers file transfer (ftp), secure shell (ssh) and remote shell (rsh) services. Host 2 is an internal database server that offers ftp and rsh services. The firewall allows the inbound ftp and ssh packets to communicate with host 1 and host 2 while it blocks other packets. Within the network, the firewall enables the internal hosts to connect to remote servers on any port.

Figure 2 shows the attack graph for an example network. It depicts three attack paths. On the right, the attack path starts with sshd_bof(0,1). This indicates a buffer overflow exploit executed from host 0 against host 1. A buffer overflow attack is an attack in which a program overwrites memory adjacent to a buffer that should not have been modified intentionally or unintentionally. The result of the sshd_bof(0,1) exploit is that the attacker has the capability of executing arbitrary codes on host 1 as a normal user. The ftp_rhosts(1,2) exploit means that the attacker exploits the ftp vulnerability on host 2 to anonymously upload a list of trusted hosts from host 1 to host 2. Such a trust relationship enables the attacker to remotely execute shell commands on host 2 without providing a password, which is indicated by the rsh(1,2) exploit. This exploit establishes the attacker's control over the database server as a user with privileges. Consequently, a local buffer overflow (local_bof(2)) exploit on host 2 escalates the attacker's privilege to be the root of that machine. The result is that the attacker can execute code on the database server with full privileges.

Diamond model

The diamond model of intrusion analysis provides a formalised way to characterise network intrusion. It gets its name from the fundamental data structure it uses to describe intrusion events. The model describes that an adversary deploys a capability over the infrastructure against a victim. These activities are called events and are the atomic features. An event defines a discrete time-bound activity restricted to a specific phase where an adversary, requiring external resources, uses a capability and methodology over the infrastructure against a victim, with a given result.

Figure 3 shows the diamond model of intrusion analysis, comprising the core features of an intrusion event: adversary, capability, infrastructure and victim. The core features are linked via edges to represent the fundamental relationships between the features that can be exploit-

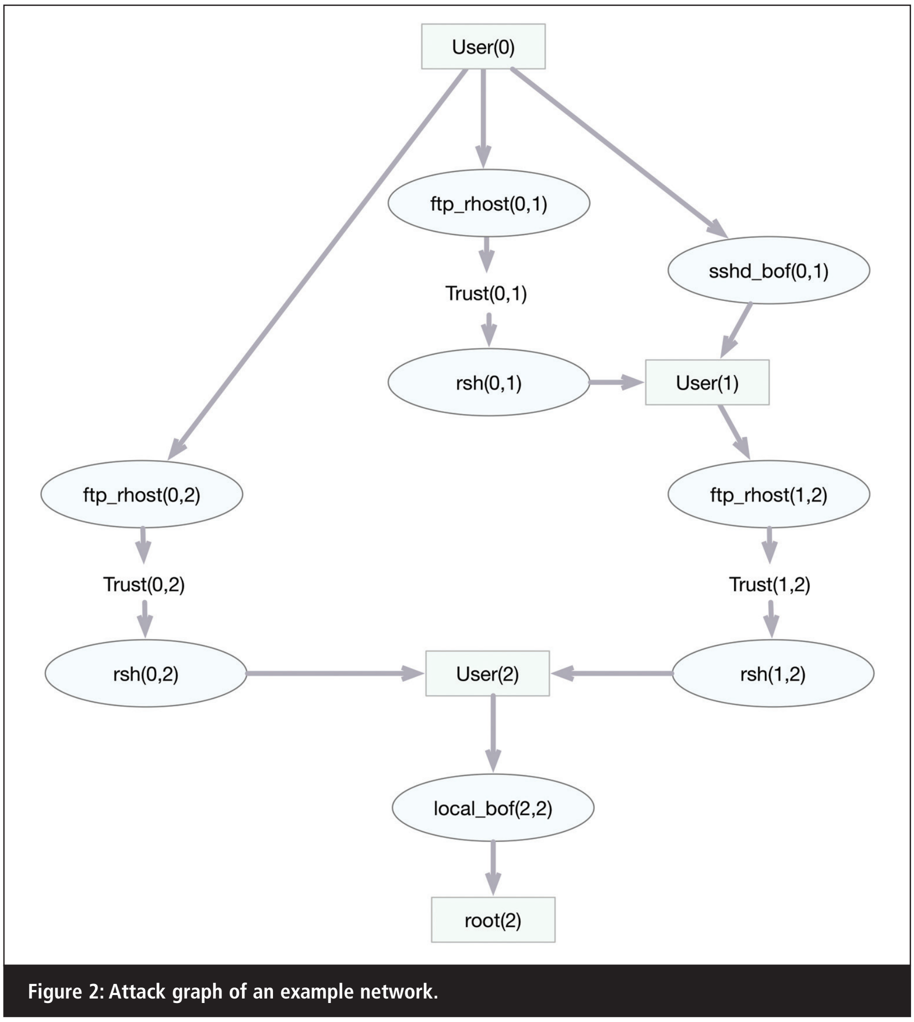


Figure 2: Attack graph of an example network.

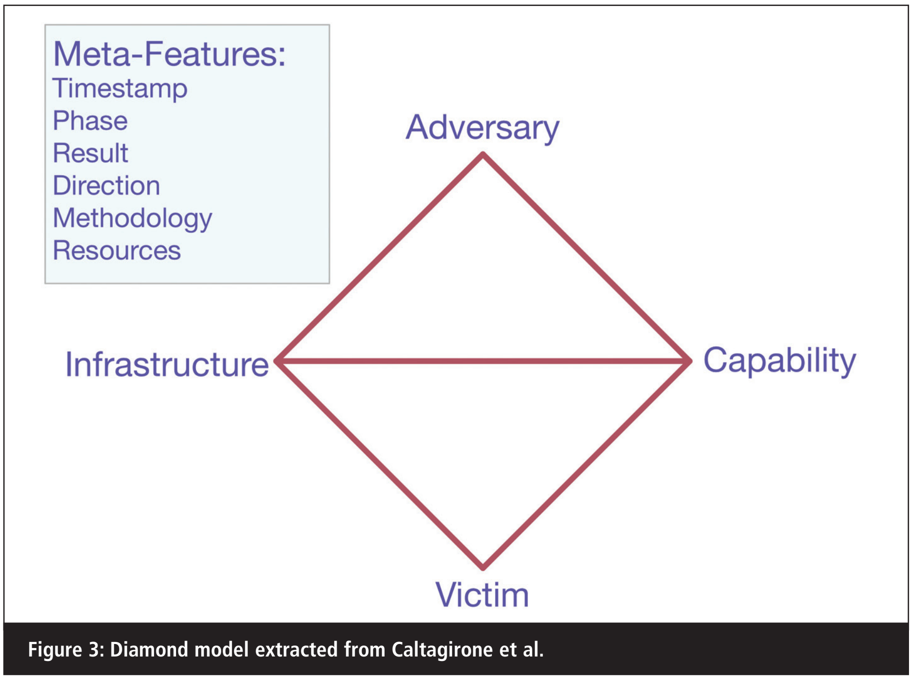


Figure 3: Diamond model extracted from Caltagirone et al.

ed analytically to further discover and develop knowledge of malicious activity. The definitions for these features are as follows:

- An adversary is the actor/organisation responsible for utilising a capability against the victim to achieve his/its intent.
- The capability feature describes the tools and/or techniques of the adversary used in the event. The flexibility of the model allows the capability to be described with sufficient fidelity. We intend for capability to be broadly understood and to include all means for affecting the victim from the most manual 'unsophisticated' methods (eg, manual password guessing) to the most sophisticated automated techniques.
- The infrastructure feature describes the physical and/or logical communication structures the adversary uses to deliver a capability, maintain control of capabilities (eg, command and control, or C2) and effect results from the victim.
- A victim is the target of the adversary against whom vulnerabilities and exposures are exploited and capabilities used. As with other features, a victim can be described in whichever way is appropriate – organisation, person, target email address, IP address, domain, etc.

The meta-features are: timestamp (both start and end), phase, result, direction, methodology and resources. The meta-features are used to order events within an activity thread. The event can also be understood and represented as a graph, as illustrated in Figure 3.

Kill chain

In military parlance, a kill chain is a phase-based model to describe the stages of an attack, which also helps inform ways to prevent such attacks. It's defined with stages such as: find, fix, track, target, engage and assess. This is an integrated, end-to-end process described as a chain because any one deficiency will interrupt the entire process.[19]

In cyber-security, it is used to identify what the adversaries must complete in order to achieve their objective. This model has seven phases of attack, which can be described as:

- Phase 1 – Reconnaissance: Information gathering can be conducted by studying targets through their public websites, following their employees on social media and using other public information. It also includes technical tactics such as scanning ports for vulnerabilities, services and applications to exploit.

- Phase 2 – Weaponisation: Adversaries analyse the data collected on their targets to determine what attack methods to use. Attackers may target specific operating systems, firewalls and other technologies. In addition, they may target the endpoints of specific people within the organisation for phishing and drive-by download attacks.
- Phase 3 – Delivery: The attacker sends a malicious payload to the victim by email, websites, USB removable media or one of many other possible intrusion techniques.
- Phase 4 – Exploitation: If the victim has downloaded the payload on to his or her computer, the main exploitation starts.
- Phase 5 – Installation: Installing malware on the infected computer is relevant only if the attacker used malware as part of the attack: even when there is malware involved, the installation is a point in time within a much more elaborate attack process that takes months to complete.
- Phase 6 – Command and control: To communicate and pass data back and forth, attackers set up command and control channels to operate between infected devices and themselves. These channels increasingly use encryption to hide their tracks.
- Phase 7 – Actions on objectives: The attacker performs steps to achieve his actual goals inside the victim's network. This is the elaborate active attack process that takes months – and thousands of small steps – to achieve. This objective is data exfiltration, which involves collecting, encrypting and extracting information from the victim's environment: violations of data integrity or availability are potential objectives as well. Alternatively, the intruders may only desire access to the initial victim's machine for use as a hop point to compromise additional systems and move laterally inside the network.

As described in Figure 4, the cyber kill chain defines the flow of a cyber-attack, so detecting and preventing attacks early in the kill chain is critical in defending against cyberthreats. The closer to the beginning of the chain you can stop an attack, the less costly and time-consuming the cleanup will be. If you don't stop the attack until it's already in your network, you'll have to fix affected machines and carry out extensive forensics work to find out what information the attackers have compromised.

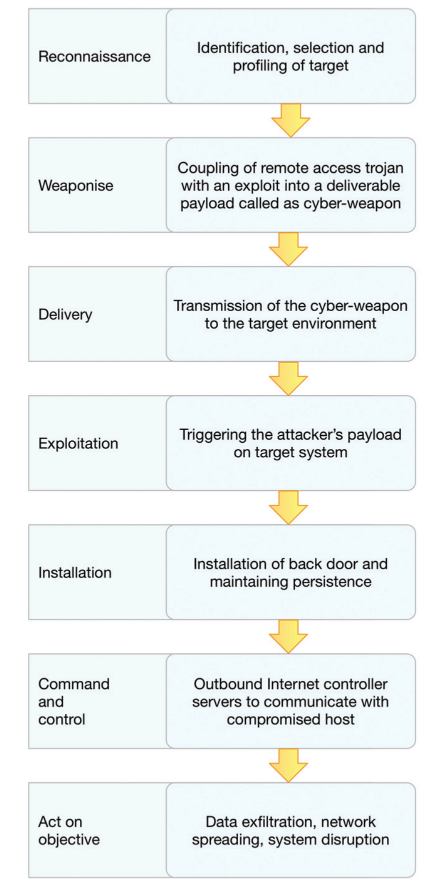

Reconnaissance
Identification, selection and profiling of target
Weaponise
Coupling of remote access trojan with an exploit into a deliverable payload called as cyber-weapon
Delivery
Transmission of the cyber-weapon to the target environment
Exploitation
Triggering the attacker's payload on target system
Installation
Installation of back door and maintaining persistence
Command and control
Outbound Internet controller servers to communicate with compromised host
Act on objective
Data exfiltration, network spreading, system disruption

Figure 4: Phases of cyber kill chain, based on Yadav and Mallari.

Breaking this cyber kill chain is a viable and efficient defence technique. However, it should be noted that the step at which we can break the chain will affect the cost and complexity of the defences required. For example, installing a firewall is easier compared to a complex log gathering and correlation solution that aims to uncover possible malicious movements within the network. This model suggests that instead of focusing on defending the organisation's perimeter alone, we should recognise the stages of an attack and incorporate controls at each level so as to protect from attacks. Understanding the different phases of the kill chain can then be used in identifying the courses of action that an incident responder may be able to use to try to put in defensive controls.

Attack surface

An attack surface represents any known, unknown or potential vulnerabilities across certain main areas of exposure – software, hardware, network and users. Reducing the attack surface can reduce risk. An attack is anything that can compromise the security of data. A specific set of conditions must be met in order to successfully perform an action.

A system's attack surface is the set of ways in which an adversary can enter the system and potentially cause damage.[20] Therefore, a larger attack surface measurement indicates that an attacker is likely to exploit the vulnerabilities present in the system with less effort and cause more damage to the system. An attacker connects to a system using the system's channels, invokes the system's methods and sends data items into the system or receives data items from the system. Hence, a system's attack surface is the subset of the system's resources (methods, channels, and data) potentially used in attacks on the system (Figure 5).

However, not all resources are part of the attack surface; a resource's contribution to the attack surface depends on the likelihood of the resource being used in attacks.[21] For example, a method running with root privileges is more likely to be used in attacks than a method running with non-root privileges and hence makes a larger contribution.

Manadhata and Wing use the entry point, exit point and channel framework to identify the resources that are part of a system's attack surface and thus formalise the notion of a system's attack surface using an I/O automaton model.

An I/O automaton A is a four-tuple consisting of an action signature, sig(A), a set of states, states(A), a set of start states, start(A), and a transition relation, step(A). An I/O automaton's environment generates input and transmits the input to the automaton using input actions.[22] In contrast, the automaton generates output actions and internal actions autonomously and transmits output to its environment. So, in Manadhata and Wing, they calculate the attack surface as the sum of all entry and exit points, channels and untrusted data elements: then they apply a damage potential and effort ratio to these attack surface elements to identify the risk areas. These elements are defined as follows:

- Entry points: Each system has a set of methods. A method receives arguments as input and returns results as output. A system's methods that receive data items from the system's environment are the system's entry points. For example, a method that receives input from a user or a method that reads a configuration file is an entry point.
- Exit points: A system's methods that send data items to the system's environment are the system's exit points. For example, a method that writes into a log file is an exit point.
- Channels: Each system also has a set of channels; the channels are the means by which users or other systems in the environment communicate with the system. Examples of channels are TCP/UDP sockets, RPC endpoints and named pipes. An attacker uses a system's channels to connect to the system and attack the system. Hence a system's channels act as another basis for attacks.

Attack modelling techniques	Advantages	Disadvantages	Security challenges
Attack graph	– Enables understanding of the system and awareness of the attack surface. – Allows user to measure the risk. – Shows the various ways an attacker can penetrate a network.	– Building an attack graph for a big network takes a long time. – It cannot be used to detect zero day attacks. – Only aware of known vulnerabilities. – Detection process is not efficient at runtime.	The model needs a more focused effort on modelling zero day attacks.
Diamond model	– Allows analysts to develop understanding to built the knowledge necessary to execute analysis. – Useful for analysing a larger threat. – Useful analyst's tool to predict adversary behaviour.	Too focused on malware-based intrusion.	The diamond model needs to be focused more on unknown attacks.
Kill chain	– Effective detection of low and slow attacks. – Very low performance requirements.	– Can't act much on early phases. – Too focused on malware-based intrusion. – Gives too little attention to threats from insiders and social engineering. – No attack prevention (yet).	The kill chain model needs to be adapted for the insider threat.
Attack surface	– Identifies and manages risks going forward as you make changes. – More practical and more useful for comparing the security of two versions of a system. – Provides a way to think about how to reduce the attack surface. – Practical metric for security measurement.	More focused on the resources that can be compromised rather than the ways an attack can be conducted.	The attack surface model should use multiple metrics to quantify different aspects of security.

Table 1: Comparison of attack modelling techniques.

Discussion

We have presented four attack modelling techniques. Each of the methods is unique and exposes the same attack in different ways. In this section, we discuss the insight achieved following this analysis.

The diamond model focuses much more on understanding the attackers, what tools and infrastucture they use and their motivations. For every intrusion event there exists an adversary taking a step towards an intended goal by using a capability over infrastructure against a victim to produce a result.

Instead of seeking to identify disruption points in single attacks, the diamond model is intended to enable better understanding of the nature of the threat. The more complete your understanding of the capabilities and technologies available to your attacker, the better placed you are to mitigate the majority of their attacks and to resist those that succeed.

The kill chain is a method for identifying and preventing intrusions. It identifies the seven essential steps that any intruder must go through to complete their objective. If the intrusion can be stopped at any step it will break the kill chain and prevent the intruder from completing their objective. It also shows that the risk and the cost to contain and remediate an attack increase at each step in the process. It is much safer and cheaper to defeat an attack at step one than at step seven. However, it is too focused on malware-based intrusion and gives too little attention to threats from insiders and social engineering. Further, it's worth considering that the first three steps are not relevant from an

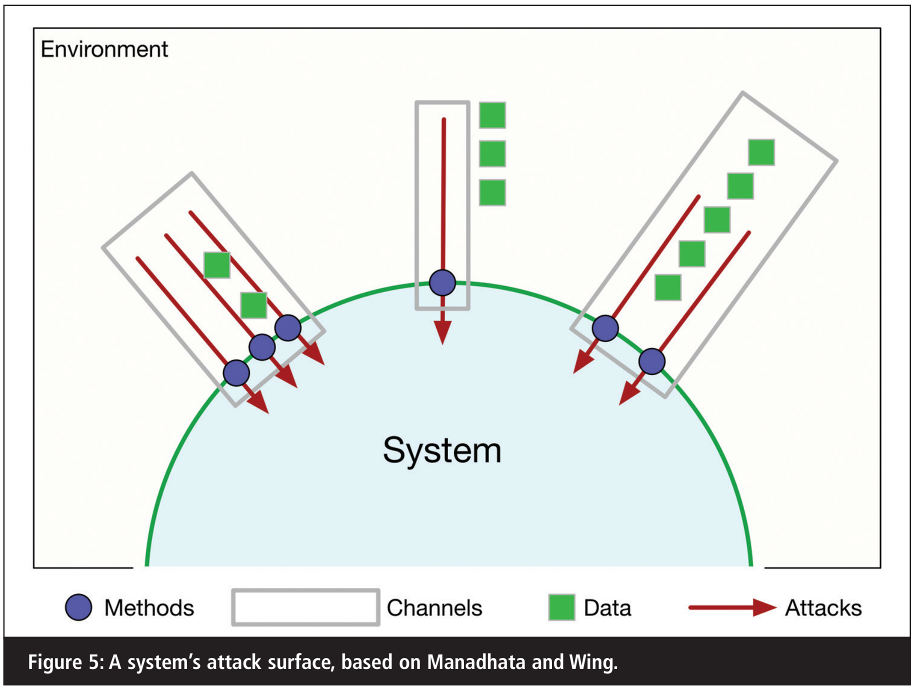


Figure 5: A system's attack surface, based on Manadhata and Wing.

operational point of view, because these are more about attacker planning and the defender can't control those. They happen outside the defended network, making it virtually impossible to identify or counter actions at these stages.

> ***"Understanding the ways in which a system can be attacked enables the development of countermeasures to prevent those attacks from achieving their goal. Using this method allows the user to model the probability that different attacks will succeed"***

The attack graph provides a method to model the threats against a system in a graphical manner. Understanding the ways in which a system can be attacked enables the development of countermeasures to prevent those attacks from achieving their goal. Using this method allows the user to model the probability that different attacks will succeed and to define indicators that quantify the cost of an attack, the operational difficulty in mounting the attack and any other relevant quantifiable measure that may be of interest. However, analysing network vulnerability using this technique is expensive, as generating a graph is difficult. For example, a network of only 10 hosts with five vulnerabilities per host takes about 15 minutes to generate and results in a graph of 10 million edges.

The attack surface is the subset of the system's resources (methods, channels and data) potentially used in attacks on the system. A model is used to identify these resources and estimate the contribution of each resource to the attack surface as a damage potential and effort ratio. This model can be applied to compare attack surface measurements and used to determine whether one system is more secure than another. It's more practical and more useful to compare the security of two versions of a system and it gives a way to think about how to reduce the attack surface.

In fact, attack modelling techniques can provide value to an organisation; they give insights about an attack and can lead to a complete understanding of the questions defenders need to ask. But these techniques are used only for known attacks and don't provide a complete and effective solution to the insider threat. The insider threat refers to harmful acts that trusted insiders might carry out, such as something that causes harm to the organisation. What we need, then, is a product that can quickly analyse data, predict breach scenarios and predict outcomes in order to decide on the best countermeasures and prevent security incidents before attacks happen.

Conclusion

We have presented four methods for modelling attack scenarios. We found that cyber-attacks can be modelled using different techniques and each of the techniques gives particular insights into a cyber-attack. For instance, the attack graph depicts ways in which an adversary can exploit vulnerabilities to break into a system; for each path it specifies how an attacker gains access to the victim computer; it tells us which vulnerabilities an attacker can take advantage of, and how; and it tells us what kind of damage may be done that can impact the organisation. This approach is useful for understanding where the system is vulnerable and to help decide which security measures should be taken.

On the other hand, the diamond model focuses on the adversaries themselves, what are their objectives, capabilities and the infrastructure they use. If the victim has strengths in infrastructure and services, the attack will not succeed. This model looks at relationships between features to help defenders better understand the threat.

The kill chain technique looks at the actions the adversary has taken; if the intrusion can be stopped at any step it will break the kill chain and prevent intruders from completing their objectives. In future work, we aim to propose an approach for modelling cyber-attacks that help to determine an attacker behavioral model to predict future and unknown attacks.

About the authors

Yassine Ayrour is a PhD student in the Model and Systems Engineering Laboratory at the National School of Computer Science and Systems Analysis (ENSIAS), Rabat, Morocco. He holds a bachelor's degree in science and technology, and a bachelor's degree in computer engineering from the National School of Mines in Rabat. He is a certified Cisco instructor and graduated in the deploy-

ment of optical fibre. His research focus is the modelling of cyber-attacks to control their impact on open information systems on the Internet.

Mahmoud Nassar is professor and head of ENSIAS. He is also head of the IMS (Models and Systems Engineering) Team of the SIME Laboratory. He received his PhD in computer science from the INPT Institute of Toulouse, France. His research interests are context-aware, service-oriented computing, component-based engineering and model-driven engineering. He leads numerous R&D projects related to the application of these domains in embedded systems, e-health, and e-tourism.

Amine Raji holds a PhD degree in software engineering from the University of Brittany. His PhD research focused on improving embedded systems reliability through the integration of formal methods during an embedded system's software development process. He worked at the Computer Science Laboratory of Bordeaux (LaBRI) as a post-doc on improving the quality and reliability of source code. Since 2014, he has applied his findings to the domain of cyber-security, proposing new methodologies and tools to improve the quality of software and software development processes to enhance the security and resilience of modern information systems.

References

1. Korzyk A, et al. 'A forecasting model for Internet security attacks'. Citeseer, 1998.
2. Simmons, C; Ellis, C; Shiva, S; Dasgupta, D; Wu, Q. 'Avoidit: A cyber-attack taxonomy'. 2009.
3. Jones, M; Salis, G; Shamma, JS. 'Cyber-attack forecast modelling and complexity reduction using a game-theoretic framework'. In Control of Cyber-Physical Systems, Springer, 2013, pp.65–84.
4. Ghosh N; Ghosh, SK. 'A planner-based approach to generate and analyse minimal attack graph'. Applied Intelligence, vol.36, no.2, pp.369–390, 2012.
5. Wang, L; Singhal, A; Jajodia, S. 'Toward measuring network security using attack graphs'. In Proceedings of the 2007 ACM workshop on Quality of protection, ACM, 2007, pp.49–54.
6. Mulazzani, M; Schrittwieser, S; Leithner, M; Huber, M; Weippl, ER. 'Dark clouds on the horizon: Using cloud storage as attack vector and online slack space'.
7. Caltagirone, S; Pendergast, A; Betz, C. 'The diamond model of intrusion analysis'. DTIC Document, Tech. Rep, 2013.
8. Manadhata, PK; Wing, JM. 'An attack surface metric'. IEEE Transactions on Software Engineering, vol.37, no.3, pp.371–386, 2011.
9. Manadhata P; Wing, JM. 'Measuring a system's attack surface'. 2004.
10. Howard, M; Pincus, J; Wing, JM. 'Measuring relative attack surfaces'. In 'Computer Security in the 21st Century'. Springer, 2005, pp.109–137.
11. Lin, X; Zavarsky, P; Ruhl, R; Lindskog, D. 'Threat modelling for CSRF attacks'. In Computational Science and Engineering, 2009. CSE'09. International Conference, vol.3, IEEE, 2009, pp.486–491.
12. Yadav T; Mallari, RA. 'Technical aspects of cyber kill chain'. ArXiv preprint, arXiv:1606.03184, 2016.
13. Ou, X; Singhal, A. 'Attack graph techniques'. In Quantitative Security Risk Assessment of Enterprise Networks, Springer, 2012, pp.5–8.
14. Kotenko I; Chechulin, A. 'A cyber-attack modelling and impact assessment framework'. In Cyber Conflict (CyCon), 2013 5th International Conference. IEEE, 2013, pp.1–24.
15. Kaynar, K. 'A taxonomy for attack graph generation and usage in network security'. Journal of Information Security and Applications, vol.29, pp.27–56, 2016.
16. Phillips, C; Swiler, LP. 'A graph-based system for network vulnerability analysis'. In Proceedings of the 1998 Workshop on New Security Paradigms. ACM, 1998, pp.71–79.
17. Barik, MS; Sengupta, A; Mazumdar, C. 'Attack graph generation and analysis techniques'. Defence Science Journal, vol.66, no.6, p.559, 2016.
18. Ou, X; Boyer, WF; McQueen, MA. 'A scalable approach to attack graph generation'. In Proceedings of the 13th ACM Conference on Computer and Communications Security. ACM, 2006, pp.336–345.
19. Hutchins, EM; Cloppert, MJ; Amin, RM. 'Intelligence driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains'. Leading Issues in Information Warfare & Security Research, vol.1, p.80, 2011.
20. Manadhata, PK; Karabulut, Y; Wing, JM. 'Report: Measuring the attack surfaces of enterprise software'. ESSoS, vol.9, pp.91–100, 2009.
21. Manadhata, PK; Tan, KM; Maxion, RA; Wing, JM. 'An approach to measuring a system's attack surface'. Carnegie-Mellon University, Pittsburgh PA, School of Computer Science, Tech. Rep. 2007.
22. Manadhata, PK; Kaynar, DK; Wing, JM. 'A formal model for a system's attack surface'. Carnegie-Mellon University, Pittsburgh PA, School of Computer Science, Tech. Rep. 2007.

The Firewall

The threat of fileless trojans

Colin Tankard, Digital Pathways

The growth in the use of fileless or zero-footprint attacks is alarming. And while they seem to have been targeted at corporate networks so far, they will spread wider.

Fileless attacks do not rely on installing new software but use legitimate applications in the OS. An advanced volatile threat (AVT) does not write itself onto the hard drive but stays in memory and is deleted once the system is reset. And it can be paired with other malware types to deliver multiple payloads.

All this means that regular anti-virus tools are less likely to detect them and hacks are more likely to become more successful. So what can be done to mitigate the likelihood of becoming a victim to a fileless attack?

The challenge is that the AVT lives in memory – it never touches the disk – which means that it is a very different type of threat. It can only steal information when the computer is running and the exposure ends when the user shuts down the machine. Without a payload file, anti-virus software can't generate a signature definition based on the malware's characteristics. This poses a problem, as the anti-virus simply does not know what to look for.

Adding to the detection difficulties, AVTs use the system's own commands to execute the attack. In Windows the netsh command is a normal, built-in function. If this script runs on a computer without a user's knowledge, the newly created network connection could be used as a path to exfiltrate data to another remote machine. The delivery of the script to run netsh would come from a PowerShell command, set up via the AVT.

It is not only Windows that can fall victim to AVTs; Linux and macOS are as vulnerable. But it tends to be that the more pervasive a technology is, the more popular a target it becomes, which is why we see so many attacks on Windows but are now starting to see an increase in malware for other OS systems.

The only way to deal with AVTs is with anomaly-based detection tools that live on each individual computer or server. These tools look at all system activity, even down to keystroke patterns, and distinguish normal from abnormal behaviour. In the case of an AVT, detection is likely because it will probably open a service to enable an external connection. It is through this service that data is sent. Hence, the behaviour would be deemed abnormal, detected and shut down.

On Windows machines a simple way to stop most AVT attacks is to disable PowerShell as most users do not need this function. To do this go to Control Panel, open Programs and Features and on the left-hand pane you will see 'Turn Windows Features On or Off'. This opens a second window, scroll down to 'Windows PowerShell' and untick.

On macOS the same can be said for switching off AppleScript and other scripting tools for most users; but administrators do need to proceed with caution as often some applications need these tools to update themselves.

With Linux the OS is generally hardened and many commands are disabled. This makes an AVT exploit that much harder to construct. Administrators should always review security release notes from the main Linux developer sites to stay up to date with security advisories and recommendations for kernel hardening.

As always, users need to be vigilant about what they open, as AVTs can be hidden in a PDF document that most users think is safe. Always check the source and, if suspicious, shut down your machine immediately, do a full reboot and then delete the attachment.

Remember, think before you click!

EVENTS CALENDAR

9–13 April 2018
HITB Security Conference
Amsterdam, Netherlands
https://conference.hitb.org

11–13 April 2018
CyberCentral
Prague, Czech Republic
http://cybercentral.eu

23–24 April 2018
Financial Services Information Security Network
Windsor, UK
http://bit.ly/2jSgLNx

24–26 April 2018
Industrial Control Systems Cyber Security Conference
Singapore
www.icscybersecurityconference.com

25–26 April 2018
IP Expo Manchester
Manchester, UK
www.ipexpomanchester.com

2–3 May 2018
InfoSecurity Denmark
Copenhagen, Denmark
www.infosecurity.dk

14–20 May 2018
NorthSec
Montréal, Canada
www.nsec.io

17–8 May 2018
RuhrSec
Bochum, Germany
www.ruhrsec.de

29 May – 1 June 2018
CyCon: International Conference on Cyber Conflict
Tallinn, Estonia
https://ccdcoe.org/cycon/