ExtraHop

SECURITY ADVISORY

Insecure Protocol Usage Exposes Organizations to Cybersecurity Risk

EXECUTIVE SUMMARY

As organizations around the world have learned the hard way in recent years, insecure protocols with known vulnerabilities expose your business to serious cyber risk. But even in the wake of costly events like WannaCry and NotPetya, many organizations still allow—knowingly or unknowingly—insecure and deprecated protocols to run in their environments. In this report, we explore some of the most common insecure protocols still in use, assess the risks associated with them, and provide guidance to security and IT operations teams about how to find and eliminate these vulnerabilities within their environment.

TABLE OF CONTENTS

INTRODUCTION

On May 12, 2017, the WannaCry ransomware variant spread like wildfire, infecting and encrypting over 230,000 computers at public- and private-sector organizations worldwide, and inflicting hundreds of millions, if not billions, of dollars in damage. Less than two short months later, another ransomware attack, NotPetya, again ripped its way through global organizations, temporarily crippling the shipping industry and costing Maersk $300 million alone.

What NotPetya and WannaCry have in common, beyond the size and scope of the damage they were able to inflict, is that they exploited the same vulnerabilities in the Microsoft Server Message Block version one (SMBv1) protocol, an exploit known as EternalBlue. What makes these attacks particularly poignant and painful is the fact that they could have been avoided.

It's common knowledge that EternalBlue was an exploit developed by the National Security Agency (NSA). The NSA used this exploit for the better part of five years before disclosing its existence to Microsoft, which promptly issued a patch in March of 2017, two months before WannaCry. Advanced threat actors are not looking for advanced technology. Quite the opposite, they are looking for the weakest link in an enterprise, making outdated, fundamentally-insecure protocols especially compelling.

If the issuance of the patch and corresponding alert from Microsoft wasn't enough to achieve widespread patching, the April 14, 2017 disclosure of the vulnerability by a group called Shadow Brokers should have made clear the severity of the vulnerability. And yet, a month after the vulnerability was made public, it was exploited to maximum effect by the WannaCry perpetrators. That it happened again just six weeks later makes it all the more painful.

And yet, today, four years after these devastating attacks took place, ExtraHop research found that SMBv1 is still surprisingly common in enterprise environments. Almost 90% had at least one device still running the protocol. And it's not just SMBv1. Other insecure protocols, including the Link-Local Multicast Name Resolution (LLMNR) protocol and the NT LAN Manager (NTLM) protocol, are still in use. And while not inherently insecure, HTTP, which is deeply problematic when used for transmission of sensitive data, is still widely used in enterprise environments.

In this report, we'll provide insight into how common these insecure protocols are within the enterprise, the risks associated with each, and provide recommendations for eliminating these weak points from your environment.

What's in a Protocol?

Protocols are the lingua franca of the network, allowing connected devices to communicate with each other regardless of differences in operating systems, hardware, and processes. Protocols work by establishing a set of rules for how data is transmitted between different network devices, and include everything from how that data is protected (encryption) to how domains are resolved. While protocols for network communication can be updated and refined, and new ones can be created, many protocols in common use today have been around for decades.

Different protocols are used in different ways, both by IT teams and malicious actors. For example, the Domain Name System (DNS) protocol makes navigating the internet easier for users by mapping IP addresses to human readable domain names, but is frequently misused by attackers for malicious purposes such as command and control (C2) and data exfiltration.

You can learn more about protocols and threat activities associated with them by visiting the ExtraHop Network Protocol Library.

This report focuses specifically on four of those protocols: SMBv1, LLMNR, NTLM, and HTTP, the prevalence of these protocols within enterprise IT environments, and the risks associated with each.

SMBv1

67% of environments use SMBv1 in 2021

INTRODUCED
1983

DEPRECATED
2013

DAMAGES
$1B+

Server Message Block (SMB) v1

Server Message Block (SMB) is a protocol that was developed in the 1980s and is used primarily for sharing things like files, printer services, and communication between networked devices. During the 1980s and 1990s, Microsoft attempted to rebrand SMBv1 under the name Common Internet File System (CIFS) and expanded its capabilities to include the transmission of large files and the establishment of direct connections over port 445.

As it turned out, SMBv1 (CIFS) was notoriously buggy, chatty, and difficult to use, and had major security deficiencies. When Microsoft introduced SMBv2 in 2006 they abandoned the CIFS nomenclature altogether. Six years later, in 2012, Microsoft introduced SMBv3, and in 2013 the company officially deprecated SMBv1.

Despite the deprecation, Microsoft continued to install SMBv1 on its Windows Servers until 2016. By that time, Microsoft was actively urging the Microsoft user community to stop using SMBv1, but with millions of machines using the protocol, many of the warnings, including those from within the Windows Server engineering group, went unheeded. This is why, when EternalBlue and related exploits—known collectively as Eternal(x)—came to light in 2017, SMBv1 was still pervasive in IT environments around the world. The Eternal(x) vulnerabilities exploit a buffer overflow vulnerability in SMBv1.

Anatomy of Eternal Blue

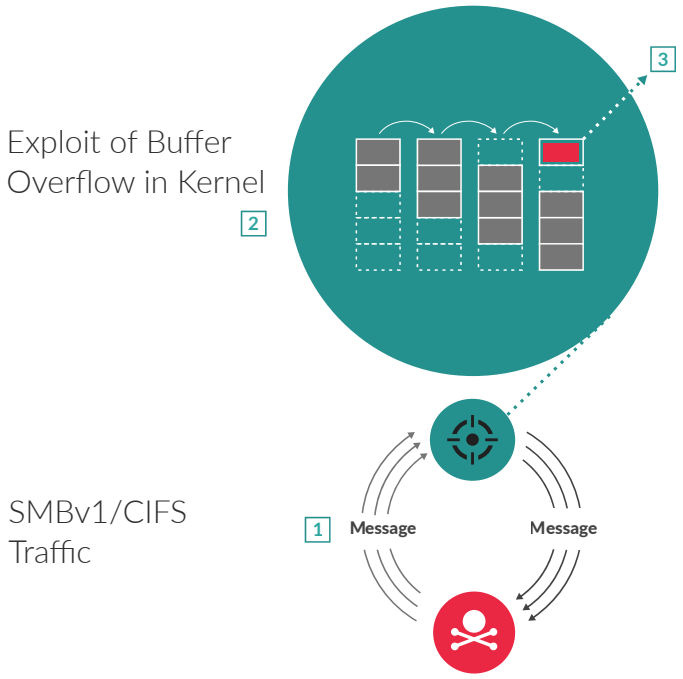

Fig. 1 Server Message Block 1.0 (SMBv1) is a version of the file sharing and transaction protocol that contains a memory calculation vulnerability, which is the target of the EternalBlue exploit. First, the attacker sends multiple requests, or messages, to a file server over SMBv1 (1). Each SMBv1 message is specially designed to manipulate the file server memory in a way that eventually causes a buffer overflow (2), which enables the attacker to deliver a malicious payload (3), such as ransomware, to the kernel of the server. Several well-known security attacks, such as the WannaCry ransomware, are associated with the EternalBlue exploit.

> SMBv1 was designed for a world that no longer exists. A world without malicious actors, without vast sets of important data, without near-universal computer usage. Frankly, its naivete is staggering when viewed through modern eyes.
>
> **- Ned Pyle, 15 year Microsoft veteran and a Principal Program Manager in the Windows Server engineering group**

How Common is SMBv1?

Based on research conducted by ExtraHop in the first three months of 2021, 88% of environments have at least one device running SMBv1. While that might seem like a staggering number, the good news is that in the cases where it's just one device, this is probably intentional. Red teams still use SMBv1 as a tool in penetration testing exercises, which probably accounts for those environments with just one instance.

The problem is that 67% of environments have at least 10 devices running SMBv1 and that's likely not intentional. While 10 devices might seem like a relatively small number, the remote code execution enabled by Eternal(x) exploits makes any device running SMBv1 an easy pivot point from which to launch a large-scale attack. These 10 devices might be a tiny fraction of the assets in an environment, but defense is a zero-fail mission. SMBv1 doesn't need to be installed on every device in the environment to be used to launch a catastrophic attack. It only needs to be on one.

The numbers only get scarier from there. According to ExtraHop data, 37% of environments have at least 50 devices running SMBv1, and 31% of environments have 100 or more devices still using the protocol four years after WannaCry and NotPetya rocked the world.

Environments with devices running SMBv1

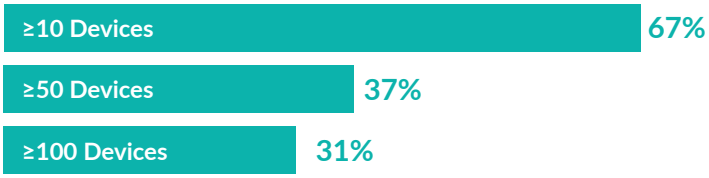

Fig. 2 This chart shows the prevalence of devices running SMBv1 across environments.

In the Wild: SMBv1

A High Stakes Gamble

In early 2019, almost two years after the EternalBlue vulnerability came to light, ExtraHop was working with a branch of a US Federal Agency. The agency was aware that it still had machines running SMBv1. What they didn't know was which machines, or how many.

> "SMBv1 doesn't need to be installed on every device in the environment to be used to launch a catastrophic attack. It only needs to be on one.

Their challenge was two-fold. First, the reason they were still running SMBv1 is because so many of their legacy systems used the protocol. Getting rid of it was a massive logistical undertaking, as many of their systems still ran on Windows XP, which couldn't use later versions of the protocol. However, they'd recently been advised by their NAS vendor that the vendor was going to stop supporting the SMBv1 protocol.

Second, while they had continued using SMBv1 long after its vulnerabilities (and the damage that could be wrought through their exploitation) had come to light, they were well aware of the risks associated with the protocol.

The security team at the agency used their SMB/CIFS Dashboard in Reveal(x) to pull up all devices running SMBv1. They quickly found that they still had a large number of devices using the protocol, including their entire VDI deployment. This protocol exposed the branch of the agency to huge risk of a fast-moving ransomware attack, among other things.

While the transition away from SMBv1 took effort, the move ultimately meant a more secure environment for the agency. Unfortunately, the ongoing use of legacy systems that don't support later versions of the SMB protocol are a major reason that many organizations remain exposed to Eternal(x) exploitation.

A Brush with Ransomware Four Years After WannaCry

In March 2021, news broke that Taiwanese computing giant Acer had been severely compromised by ransomware. The $50 million ransom demand was the largest in history, and REvil, the cybercriminals responsible for the attack, had architected a trump card. Not only had their attack encrypted a large percentage of the company's files, they had also exfiltrated huge quantities of data. In the event Acer had a back-up, they could still extract the ransom by threatening to leak the stolen data.

The exfiltrate-then-encrypt play is becoming increasingly common in ransomware attacks as cybercriminals try to maximize their leverage. Just a few months before the disclosure of the Acer attack, an ExtraHop customer experienced a very similar attack.

In late 2020, a large ExtraHop customer based in North America was alerted to a Ransomware Activity detection in Reveal(x) 360. The same devices were also seeing alerts for detections on SMB data staging and suspicious file reads. By looking at Related Detections, the customer's security team determined that the attackers were also in the process of exfiltrating data before they encrypted it in an effort to inflict maximum damage.

The team was able to quickly identify and quarantine affected assets and accounts, and as a result, the attackers were only able to encrypt a small percentage of targeted files.

While the customer averted disaster in this case, it serves as a cautionary tale, not only about the new ransomware playbook, but about the ongoing risks associated with SMBv1. Devices running this protocol remain an easy target for attackers, especially ransomware gangs more interested in money than in information.

You can read more about this ransomware trend here.

> "An attacker can quickly spread malware to other unpatched servers across a network.

The Risks of SMBv1

The reason that SMBv1 has been so successfully exploited for attacks like WannaCry and NotPetya is that, if an attacker can successfully access a server with SMBv1 enabled, they can quickly spread malware to other unpatched servers across a network. In an SMBv1-based attack, the attacker sends multiple requests, or messages, to a file server over SMBv1. Each SMBv1 message is specially designed to manipulate the file server memory in a way that eventually causes a buffer overflow, which enables the attacker to deliver a malicious payload, such as ransomware, to the kernel of the server.

Buffer Overflow attacks of system level services are particularly dangerous because they provide attackers with a wide latitude for follow-on activities such as injecting malicious payloads, scraping user credentials from memory, and setting up hidden persistence mechanisms that reside only in memory.

LLMNR

70% of environments use LLMNR in 2021

INTRODUCED
2007

DEPRECATED
N/A

DAMAGES
Unknown

Link-Local Multicast Name Resolution (LLMNR)

Link-Local Multicast Name Resolution (LLMNR) is a protocol that allows name resolution without a DNS server. Essentially, LLMNR is a layer 2 protocol that provides a hostname-to-IP resolution on the basis of a network packet that's transmitted via Port UDP 5355 to the multicast network address (224.0.0.0 through 239.255.255.255). The multicast packet queries all network interfaces looking for any that can self-identify authoritatively as the hostname in the query.

LLMNR is distinct from other protocols described in this report, in that it was never adopted as an IETF standard protocol (although it was defined in RFC 4795).

LLMNR was originally created as a workaround to enable name resolution in environments in which DNS servers would be impractical, such as small private networks. LLMNR was created as a way to achieve name resolution without the onerous requirements of DNS. The protocol has been (and still is) used by operating systems, including Microsoft Windows, to identify networked devices like file servers.

How Common is LLMNR?

For a protocol that's not quite a protocol—and a very risky one at that—it's usage is still surprisingly common. De-identified network telemetry from ExtraHop Reveal(x) shows that 70% of enterprise environments have at least 10 clients still running LLMNR. Unfortunately, as the device counts go up, the number of environments with vulnerable clients doesn't decline by much. Fifty-five percent of environments have fifty or more LLMNR clients, while 46% have more than 100 such clients.

Environments with devices running LLMNR

Fig. 3 This chart shows the prevalence of devices running LLMNR across environments.

The Risks of LLMNR

While LLMNR provides a DNS-free mechanism for host-name resolution within a local environment, it also provides an avenue of attack for malicious actors. An attacker can use the protocol to trick a victim into revealing user credentials. This is done by leveraging LLMNR to gain access to the user credential hashes, which can then be cracked to reveal actual credentials, especially if older MS password techniques like LANMAN are not disabled.

How does this work if you're the attacker? The first step is to configure a node on the network to answer that it is the hostname associated with any query. For the requesting client, this creates a race condition in which the client will not only accept, but trust whichever device answers first. This is because the protocol specifications of LLMNR state that all client responses are trustworthy.

This is when credential hashes come into play. When the client receives a definitive “It’s me!” from a compromised device, it automatically sends a hashed copy of the current user's credentials as part of the response. This provides the attacker with a hashed copy of the credentials which can be decrypted, or leveraged via pass-the-hash attacks to begin escalating privileges within the broader network.

While DNS is not without its challenges, it’s a far more secure way to accurately identify host names. With that said, DNS should be carefully monitored to ensure that it is not itself being utilized for nefarious purposes.

Learn how the SolarWinds SUNBURST attackers used DNS ›

LLMNR Broadcast

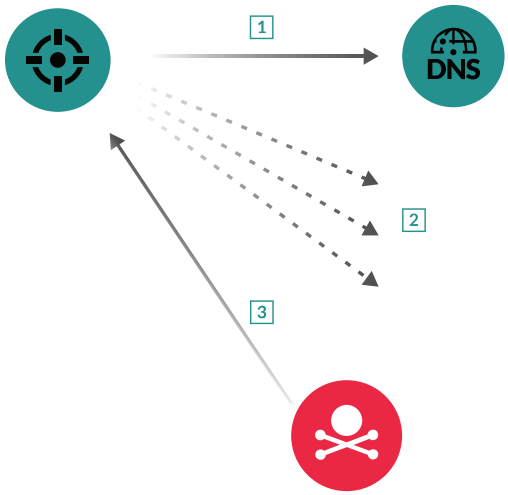

Fig. 4 When a DNS request for a hostname cannot be resolved or a DNS server is unavailable (1), clients with LLMNR enabled will broadcast a query to all local devices over UDP 5355 (2). If an attacker is listening, they can respond and impersonate the requested host (3). If the host is an authenticated resource, the client response will contain user credentials.

NTLMv1

34% of environments use NTLMv1 in 2021

INTRODUCED
1993

DEPRECATED
2010

DAMAGES
Unknown

New Technology LAN Manager (NTLM) v1

New Technology LAN Manager (NTLM) is a proprietary Microsoft protocol introduced in 1993 to replace Microsoft LAN Manager (LANMAN). NTLM is part of a cohort of Microsoft security protocols designed to collectively provide authentication, integrity, and confidentiality to users.

NTLM is what is known as a challenge-response protocol used by servers to authenticate clients using password hashes. In its original incarnation NTLMv1, used a fairly simple (and easily compromised) authentication method. The process by which NTLM authenticates users is described by Microsoft below:

> ***NTLM credentials are based on data obtained during the interactive logon process and consist of a domain name, a user name, and a one-way hash of the user's password. NTLM uses an encrypted challenge/response protocol to authenticate a user without sending the user's password over the wire. Instead, the system requesting authentication must perform a calculation that proves it has access to the secured NTLM credentials.***

While NTLMv1 was eventually replaced by NTLMv2, the next generation of the protocol didn't fare much better when it came to actually protecting passwords from being intercepted by malicious actors. NTLMv2 added a few features, including a time stamp and a usernames appended to the hash. While this helps mitigate offline relay attacks, it introduced other vulnerabilities and did little to improve the overall security of the protocol.

The problem with this cryptography scheme is that it's incredibly easy to crack. In 2012, it was demonstrated that every possible permutation of NTLM's eight-byte hash could be cracked in under six hours. In 2019, an open-source password recovery tool known as HashCat demonstrated that it could crack any eight-byte hash in under two and a half hours, a fact the *The Register* noted—with its trademark snark—was less time than it took to watch Avengers: Endgame.

How Common is NTLMv1?

Despite the recommendation from Microsoft that organizations cease use of NTLM in favor of the much more secure Kerberos authentication protocol, NTLM is still quite common in enterprise environments. While ExtraHop did not look at the ubiquity of NTLMv2, the less secure of the two versions, NTLMv1 is still in broad use.

Environments with devices running NTLMv1

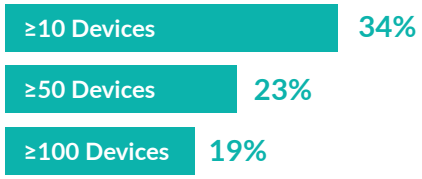

Fig. 5 This chart shows the prevalence of devices running LLMNR across environments.

According to network telemetry from Reveal(x), 34% of environments have at least 10 clients running NTLMv1, less than the number of organizations with the same number of devices running SMBv1 and LLMNR, but still a significant number for a protocol that has been deprecated for over a decade, with 23% of organizations having fifty or more clients using NTLMv1, while 19% have one hundred or more clients using the protocol for authentication.

> “
> A skilled attacker can easily intercept NTLM hashes that are equivalent to passwords or crack NTLMv1 passwords offline.

The Risks of NTLM

Using NTLM for authentication exposes organizations to a number of risks. A skilled attacker can easily intercept NTLM hashes that are equivalent to passwords or crack NTLMv1 passwords offline. A successful exploit against NTLMv1 authentication can enable an attacker to launch machine-in-the-middle (MITM) attacks or take complete control of a domain.

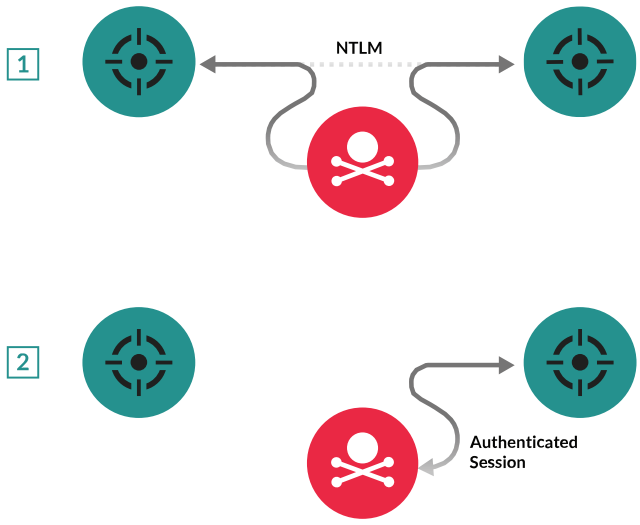

Fig. 6 During an NTLM relay attack, the attacker acts as a machine-in-the-middle (MITM), receiving and then forwarding the NTLM messages [1]. The attacker then creates an authenticated session with the server [2]. With this technique, the attacker only needs the NTLM hash to move laterally across the network or access sensitive information stored on servers.

In an MITM attack, malicious actors insert themselves between the client and the server, allowing them to intercept all data transmitted between those devices. If the communications are not encrypted, or the ciphers are weak, as in the case of NTLM, MITM attacks can result in compromised credentials, scrapped PII, stolen trade secrets, the injection of fake or manipulated data, and much more. In fact, recently patched flaws in current NTLM versions allow attackers to force a downgrade of the security features within the NTLM protocol. Attackers can then leverage stolen credentials to pivot between machines until they identify domain admin accounts that can be used for direct AD Server access and a full domain takeover.

HTTP

81% of environments use HTTP in 2021

INTRODUCED
1991

DEPRECATED
N/A

DAMAGES
> $1B

Plaintext Credentials over Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is almost certainly the most commonly known application protocol. Whether you know anything about networking or not, chances are you've seen it at the beginning of every web address you've ever visited. A distributed, collaborative, hypermedia information system, it allows users to communicate data on the World Wide Web. Alongside its sister protocol, HTML, it's the foundation of the internet we know today.

In 1995, four years after the introduction of HTTP, it's more secure version, HTTPS, arrived on the scene. This was in response to demand for companies to be able to process payment information over the internet—something that couldn't be done over HTTP without blatantly exposing payment card information. Unlike HTTP, HTTPS uses TLS to encrypt the communications between clients and servers, preventing people from intercepting and reading your data in flight. It also preserves the integrity of data, helping to prevent it from being broken or corrupted.

While HTTP has never been officially deprecated, in 2017, Google took a major step toward phasing out the use of insecure HTTP. In January of 2017, Google Chrome started marking all non-HTTPS websites as "insecure." Since that time, any website that stores information such as login credentials, credit card information, or PII must use HTTPS in order to function effectively in Chrome. Google also gives search priority to HTTPS sites.

While HTTP is not inherently problematic, its use for transmission of sensitive data is definitely a major risk. When plaintext credentials are transmitted over HTTP, those credentials are left exposed, the internet equivalent of shouting passwords across a crowded room, making it trivial for anyone to intercept and steal those credentials.

How Common is Plaintext over HTTP?

Detections data from ExtraHop Reveal(x) 360 shows that 81 of 100 enterprise environments still use insecure HTTP credentials:

81%
Use Unencrypted HTTP Credentials

Fig. 7 This graph shows the prevalence of unsecured HTTP across enterprise environments.

In the Wild: Plaintext Credentials over HTTP

In March 2021, ExtraHop experts were working with a major metropolitan law enforcement agency when Reveal(x) encountered a "Credentials Sent over HTTP" detection. The detection showed that the device sending credentials was a law enforcement agency laptop. The hostname matched the naming scheme for agency laptops, the IP address was within agency domain space, and the user associated with the laptop was a law enforcement officer with the agency.

The security team then pivoted to look at the target of the HTTP credentials, which happened to be an internet address. Looking at related records, the team identified the address as the domain for a common law enforcement research forum that's also available to the general public. The domain was HTTP, not HTTPS.

Even more concerning was the fact that the officer was accessing a portion of the site for verified members (typically law enforcement agents and other investigators) that requires a login. Because the domain was HTTP and not HTTPS, usernames and passwords were both sent in the clear, making it easy for malicious actors to find and steal those credentials directly from the web.

Further investigation found that the site in question had a proper SSL certificate and had SSL enabled. The site was just not configured to make encryption mandatory, potentially exposing the login credentials of thousands of law enforcement officers and other investigators who are verified site users.

The Risks of Plaintext Credentials over HTTP

As illustrated in the example above, sending plaintext credentials over HTTP exposes users and the organizations for which they work to a number of risks. In addition to credentials, HTTP websites can easily expose sensitive customer data such as credit card information and PII.

> "HTTP websites can easily expose sensitive customer data such as credit card information and PII.

Of course, even HTTPS isn't foolproof. Heartbleed, a serious vulnerability in OpenSSL that first came to light in 2014, is a classic example of how HTTPS can be exploited. Under normal conditions, SSL/TLS encryption protects information—such as logins and credit card numbers—being transmitted over the internet. The Heartbleed vulnerability inadvertently exposed the memory of systems protected by OpenSSL, compromising the secret keys used to encrypt the traffic and giving attackers access to users names, passwords, and other sensitive information.

Because HTTP or HTTPS are often used to transmit user input from websites and web applications, the protocols are sometimes abused to transmit malicious content from the public internet into a private environment. For example, an attacker using the SQL injection tactic sends SQL statements hidden in HTTP headers or other user-manipulatable fields in the HTTP protocol. The encryption used by HTTPS can actually make it more challenging to detect SQL injection attacks.

Even with vulnerabilities like Heartbleed, HTTPS is still far more secure than HTTP for transmission of sensitive information.

A Note on TLS 1.0/1.1

Recently IETF announced the formal deprecation of the TLS protocol version 1.0 and 1.1. In the coming months, ExtraHop will carefully monitor the ongoing use of these protocol versions across enterprise environments to understand how and if it is being phased out.

How to Determine Whether You're Running Insecure Protocols

There are many ways you can end up running insecure or deprecated protocols or protocol versions in your environment, but only a few ways to find them and get them out.

When it comes to the introduction of insecure protocols, the "tyranny of default" is often to blame. Devices and software that communicate across the network are configured with default settings that may go out of date over time. If a new device or solution is introduced into the environment, but left to its default configuration, it may run protocols that are no longer considered secure.

Similarly, cloud systems and workloads use configuration templates to determine their protocol usage, and over time, as new protocols are developed and old versions deprecated, these configuration templates may go out of date and need to be updated. Any new workloads created with an older template may introduce insecure protocols into the environment. Because of the often short-lived and ephemeral nature of cloud workloads, it can be very challenging to catch these instances of insecure protocol usage and know how to get them out of your system.

> "Maintaining an inventory of software and hardware in your environment is a fundamental necessity for security hygiene.

So, how do you do it?

Manual, Point-in-Time Audit

Maintaining an inventory of software and hardware in your environment is a fundamental necessity for security hygiene, and is recommended in the first and second security controls in the CIS Top 20. Despite being a vital security practice, maintaining this inventory is a challenge for many organizations. One way to gather such an inventory is through a manual audit using a combination of scanning tools, ranging from the free, open source tool Nmap to costlier commercial products.

While scanning the network for devices and protocols in use can provide a valuable snapshot of a point in time, it doesn't protect you against new occurrences of insecure protocols. Neither does a point-in-time scan offer any guarantee that you've discovered every instance of an insecure protocol actively in use. Particularly in a large and heavily segmented network, scanning tools may simply not uncover what you're looking for. As one security analyst once remarked, "The network is dark and full of terrors."

So how do you find insecure protocols in an enterprise-scale network?

Continuous Network Monitoring

By passively monitoring and analyzing network traffic continuously, you can discover each protocol that is in use on the network at any given moment. This addresses two of the primary challenges:

1. Manual audits only give you a point-in-time snapshot. There are many well-documented ways for insecure protocols to be reintroduced to your environment between audits, which are usually conducted quarterly at best.
2. Manual audits are extremely time- and energy-consuming, taking away time from other more critical security functions like responding to actual threats.

With the increase in remote and distributed workforces and hybrid environments with both on-premises and cloud components, the number of ways that insecure protocols may be introduced into your environment have multiplied at the same time that maintaining an accurate inventory has grown massively more difficult. Continuous monitoring of network traffic for protocol identification and threat detection and response has gone from a nice-to-have to a must-have.

A Note on Data Sourcing

The ExtraHop platform was engineered from the beginning for privacy and security. Our products passively monitor and analyze over four petabytes of network traffic per day to understand the communications between all devices and applications. We then extract de-identified metadata and send it to the cloud where we can leverage the scale and compute resources to apply advanced machine learning to more than 75 protocols for high-fidelity threat detection.

But while ExtraHop Reveal(x) can see every communication, device, and workload within a customer environment—from the cloud, to the data center, to the IoT device—that doesn't mean ExtraHop security researchers can. Our platform contains layers of security and privacy controls designed to protect customers. If you've ever wondered why ExtraHop issues relatively few reports that mention our own customers' data, it's because we made it very hard for anyone but the customer to whom it belongs to access it. We firmly believe that's how it should be.

Please note that statistics in this report were gathered from large samples, and may have small error margins.

What we can extract from our platform is information aimed at making sure that our customers have the very best visibility and insight. Take the device data, for example. ExtraHop uses de-identified (anonymized) aggregate data to catalog device models, ensuring that all customer systems get smarter whenever a new model is seen by a Reveal(x) sensor. It puts data to work for everyone without compromising privacy or security—and that's a win-win.