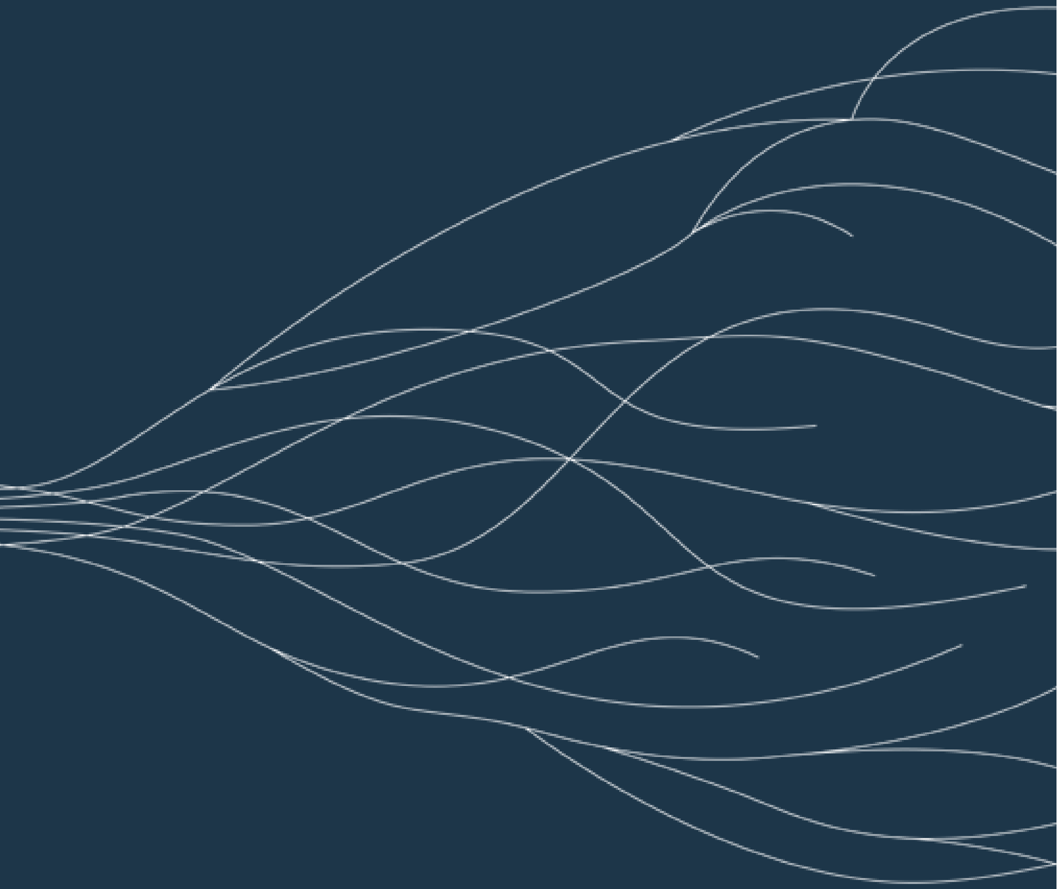
IBM Security



The importance of thwarting command injection attacks

How cybercriminals use this common attack vector to invade systems and data

IBM X-Force® Research
Managed Security Services Report

IBM

Click here to start ▶

Contents

Executive overview
1 • 2

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

Executive overview

Looking across the threat landscape at cybercriminals' go-to attack vectors, we see SQL injection high on the list, but there's another injection method that also poses a serious threat: command injection.

Command injection attacks are among the most common attacks on the Internet, where numerous entry points offer openings for a system compromise. The chances of success are enhanced by the complexity of measures needed to defend against them. IBM Managed Security Services observed over 84,000 such attacks across our data set for the thirteen months ending 31 May 2016.

The difference between SQL injection and command injection is that while commands in an SQL injection attack are injected into the targeted database, command injection lets attackers inject shell commands into the host operating system (OS) running the website. For example, an attacker might figure out the directory where an app is installed and run a script in that directory.

About this report

This IBM® X-Force® Research report was created by the IBM Managed Security Services Threat Research group, a team of experienced and skilled security analysts working diligently to keep IBM clients informed and prepared for the latest cybersecurity threats. This research team analyzes security data from many internal and external sources, including event data, activity and trends sourced from thousands of endpoints managed and monitored by IBM.

Contents

Executive overview
1 • 2

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

A successful command injection attack allows an attacker to issue arbitrary commands within a vulnerable web application environment. This happens when an application passes malicious user-supplied input—via, for example, forms, cookies, or HTTP headers—to a system shell. If the data input is not validated properly, the attacker can "inject" additional shell commands and have them executed with the permission of the vulnerable application. Simply put, this means that a critical web server and its entire back-end database can be completely compromised.

The current most common command injection assault kicked off in late September 2014, when a more than 20-year-old vulnerability in the GNU Bash shell widely used on Linux, Solaris and OS X systems sparked the mobilization of attacks known as Shellshock. That first vulnerability quickly gave way to the disclosure of several additional vulnerabilities affecting the UNIX shell. IBM Managed Security Services (MSS) observed a significant increase in focused attacks targeting these vulnerabilities within 24 hours of their disclosure. The attacks came in waves, from different source IPs and originating countries. Today, almost two years later, we're still seeing a significant number of Shellshock attacks.

The still widely used Shellshock command injection takes advantage of a vulnerability in a shell widely used on Linux, Solaris and OS X operating systems.

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot
1 • 2 • 3

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

Trends: Shellshock, WordPress and Perlbot

Although attackers have many command injection vulnerabilities at their disposal, three notable trends stand out in our data.

First, Shellshock is still wreaking havoc on vulnerable systems. To put its continued prominence in perspective, consider that among command injection attacks over the past year, some months featured twelve Shellshock attacks for one attack of any other kind (see Figure 1).

The second notable trend in our data concerns WordPress, the popular content management system (CMS) platform. The activity spikes for non-Shellshock command injection attacks we see in April, May and November of 2015 are tied directly to increased attacks on WordPress installations (see Figure 1). Cybercriminals know there are large numbers of unpatched installations on the web, so they focus heavily on CMS-based sites. The IBM report Understanding the risks of content management systems highlights the weaknesses associated with CMS platforms.

Shellshock and all other shell command injection attacks compared

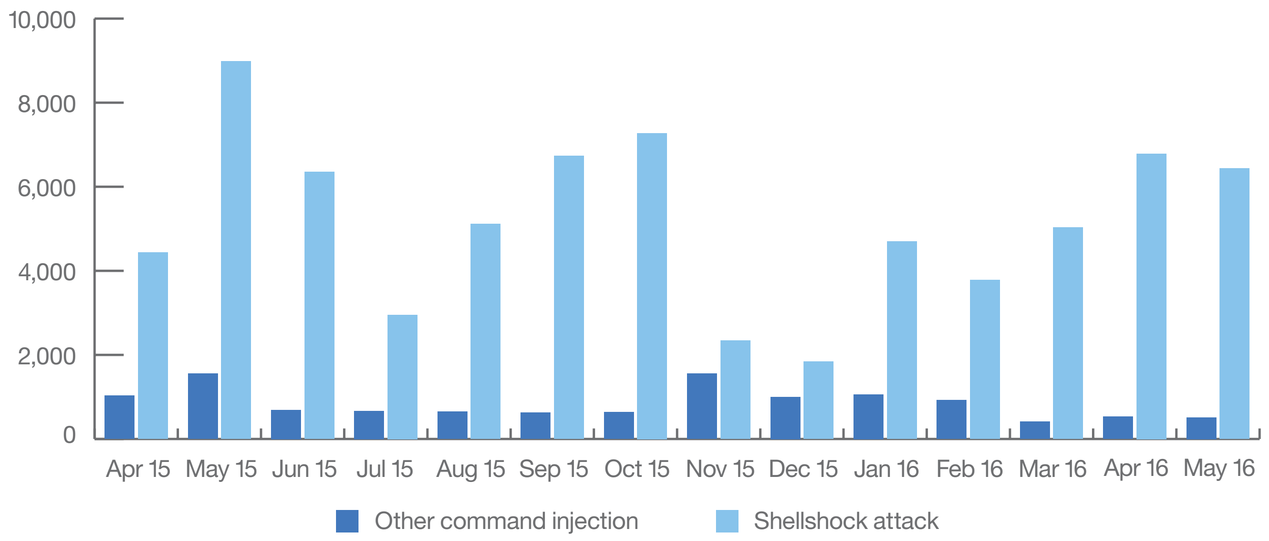


Figure 1. Shellshock attacks continue to significantly outnumber all other forms of shell command injection attacks. Source: IBM Managed Security Services data (April 1, 2015 – May 30, 2016).

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot
1 • 2 • 3

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

Content management systems' open source code makes them an attractive target, and security flaws are being found in them on a regular basis. As with other applications, they also face the risk of improper deployment, configuration and overall maintenance by system administrators.

Fortunately, the number of publicly disclosed command injection vulnerabilities has been trending downward since 2013 (see Figure 2). Vulnerabilities identified in 2015 dropped by more than 50 percent compared to 2014. Exploit code was made publicly available for 40 percent of the vulnerabilities disclosed in 2014, but for only 22 percent in 2015, possibly because in 2015, attackers were still successfully exploiting unpatched older vulnerabilities released in 2014.

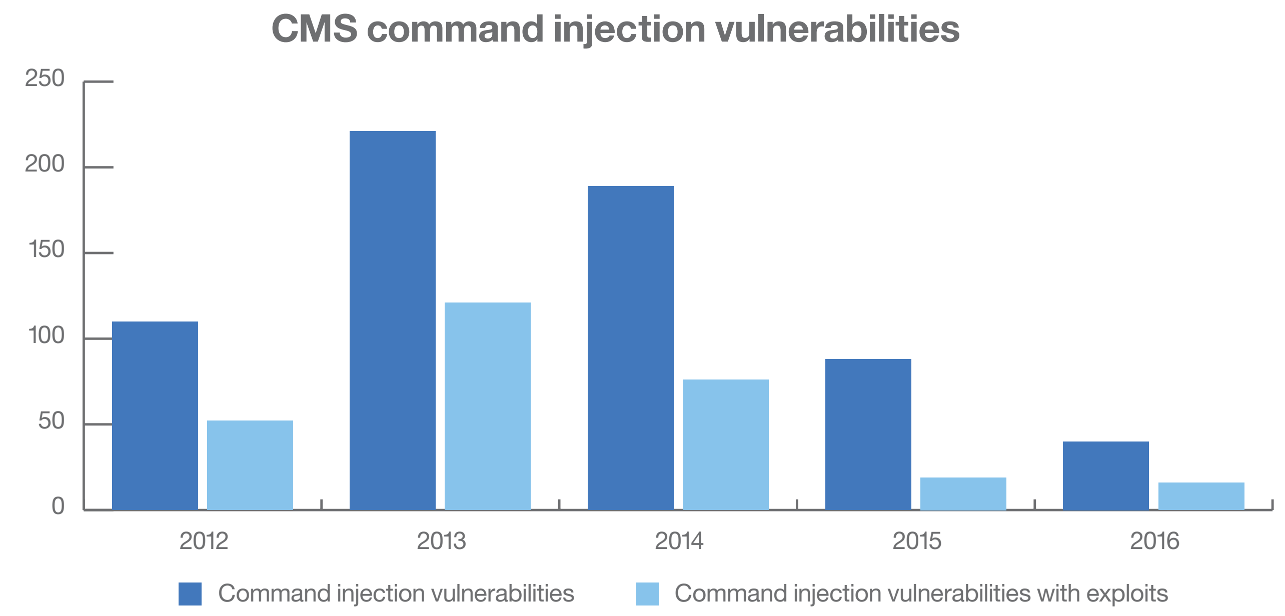


Figure 2. The number of publicly disclosed WordPress vulnerabilities has declined since 2013. Sources: MITRE and IBM X-Force Vulnerability Database data (January 1, 2012 – May 1, 2016).

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot
1 • 2 • 3

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

The third notable trend seen in our data was a large influx of Perlbot traffic. Based on our observations, Perlbot injections typically are focused on vulnerable Plesk Desk installations, a web-based administration system for setting up and managing hosted sites. The Perlbot injections establish a persistent internet relay chat (IRC) connection back to a command-and-control host, with the goal of performing vulnerability scans and performing denial of service (DoS) attacks. In the months of May and November 2015, our data showed a large spike of activity aimed at using these infected hosts to drain bandwidth, impacting availability. That was not the case in many of the other attacks on vulnerable web applications, where the only aim was to gain complete control of the target server.

Perlbot injection attacks often establish a persistent connection back to a command-and-control host that will in turn perform vulnerability scans or denial of service attacks.

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends
1 • 2

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

Geography and industry trends

According to IBM's data, attackers in the United States are responsible for fully 45 percent of all command injection attacks from April 2015 through May 2016 (see Figure 3). IBM X-Force researchers believe that most attacks are the work of cybercriminal organizations seeking sensitive information to be used for monetary gain.

The industry sectors most heavily targeted for command injection attack were manufacturing and financial services, together receiving 57 percent of all attacks (see Figure 4). The IBM report Security trends in the manufacturing industry revealed that in 2015, Shellshock accounted for the vast majority of all attacks, over 26 percent. In the financial industry that figure was almost as high: nearly 19 percent.

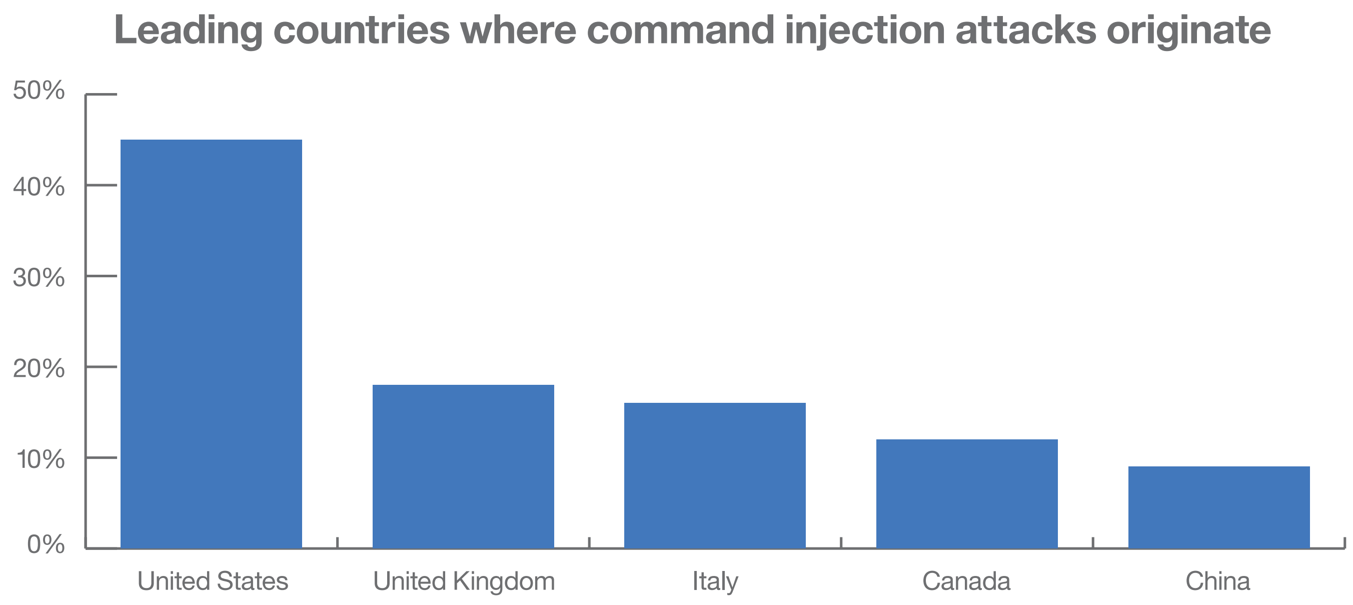


Figure 3. The United States was the leading source of command injection attacks. Source: IBM Managed Security Services data (April 1, 2015 – May 30, 2016).

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends
1 • 2

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

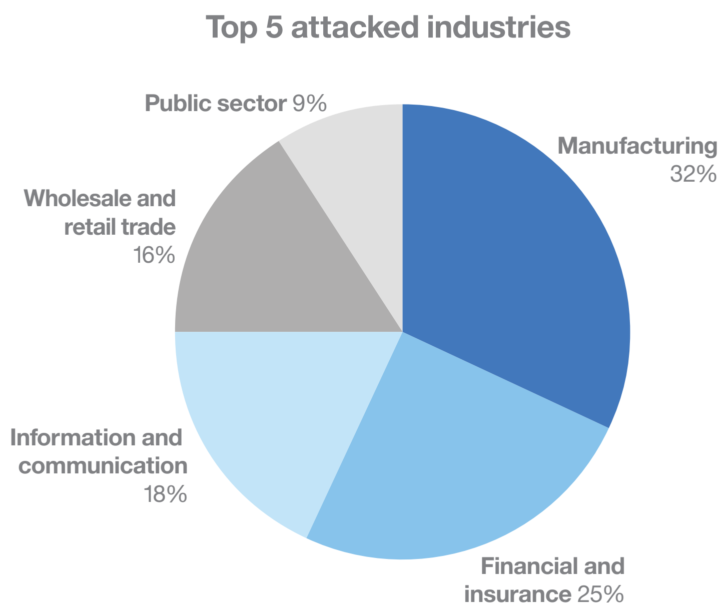


Figure 4. Leading industries experiencing command injection attacks. Source: IBM Managed Security Services data (April 1, 2015 – May 30, 2016).

Anatomy of a command injection attack

Once attackers have found the vulnerability in an application, they place a custom shell script on the target server in order to allow further access. In most cases, IBM MSS sees PHP pages dedicated to obtaining shell access pushed to the target server once the vulnerability has been exploited. Having gained shell access, the attacker can monitor all processes in order to find additional attack vectors.

For example, the target could be a database user connection against which injection attacks can be performed directly because the attacker is inside the victim's network. By creating setuid files with global read privileges, the attacker can gain full access and control over the database. Applications are considered vulnerable to command injection attacks if system-level commands are executed by user input, for example a web app that passes unexpected input to the shell command because no input sanitization was used.

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

Protect applications by restricting common operators

Use of the following operators should be restricted or sanitized within the data input fields of any web application, especially search fields.

Redirection operators

Examples: `<, >>, >`

These operators redirect either input or output somewhere else on the server. `<` will make whatever comes after it standard input. Replacing the filename with `<` filename will not change the output, but could be used to avoid some filters. `>` and `>>` redirect command output, and can be used to modify files on the server or create new ones altogether. Combined with the `cat` command, either could easily be used to add Unix users to the system or to deface the website.

Pipes

Examples: `|`

Pipes allow the user to chain multiple commands. A pipe will redirect the output of one command into the next, so you can run unlimited commands by chaining them with multiple pipes, such as `cat file1 | grep "string"`.

Inline commands

Examples: `;, $`

Using a semicolon asks the command line to execute everything before the semicolon, and then execute everything else as if on a fresh command line.

Logical operators

Examples: `$, &&, ||`

These operators perform some logical operation against the data that appears on the command line, before and after the operator.

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

Common injection patterns and results

Here are the expected results from a number of common injection patterns that append an operator to a given input string, assuming all quotes are correctly paired:

- `` `shell_command` `` executes the command
- `$(shell_command)` executes the command
- `| shell_command` executes the command and returns the output of the command
- `|| shell_command` executes the command and returns the output of the command
- `; shell_command` executes the command and returns the output of the command
- `&& shell_command` executes the command and returns the output of the command
- `> target_file` overwrites the target file with the output of the previous command
- `>> target_file` appends the target file with the output of the previous command
- `< target_file` sends contents of target_file to the previous command
- `- operator` adds additional operations to target command

These are some examples of possible command injection vectors. The full breadth of attack possibilities depends on the underlying function calls. For example, if an underlying function is using a shell program such as AWK, more attack possibilities exist.

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations
1 • 2

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

Recommendations

Many network IPS appliances use a signature-based technology to detect network intrusions. However, relying on raw signature events to detect command injection—or any attack type, for that matter—is hardly a perfect solution. The reason is that raw events don't normally display contextual information in a first-view application unless the intrusion prevention device is instructed to do so—a labor-intensive process that is very demanding on the device resources and the reporting application. Therefore many organizations deploy a security information and event management (SIEM) solution.

One example is the IBM Security QRadar® SIEM, which uses correlation logic to perform analytics on raw signature events. It employs a separate log feed that pulls the full raw log details straight from the endpoint. It then performs the de-obfuscation needed so the SIEM product can recognize, analyze and alert upon real-time attack strings. New attack strings surface daily. We analyze them and either implement them into existing correlation rules or create new rules to improve detection. If your SIEM identifies a command injection attack, it's imperative that you inspect the target server(s) immediately.

Relying solely on raw signature events detected by network IPS appliances will not provide sufficient protection against command injection attacks.

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations
1 • 2

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

Following are some best practices to mitigate command injection attacks:

- Most attack strings are purposely meant to exploit an application first, so you MUST ensure your patch management solution is robust.
- As with SQL injection, sanitizing expected user input is a key prevention method. Form and URL data need to be validated for potentially malicious characters. A white list of characters allowed for use in passwords should be created to validate user input. Note, however, that limiting characters can diminish the strength of account credentials, so password fields should allow a full range of alpha numeric characters. Just NEVER allow shell code operators in commands or database queries that are concatenated using user input.
- Web applications and their components should run with strict permissions that never allow any operating system command execution.
- Vulnerabilities may exist within some web development languages, including PHP and Perl. A more secure solution would be to avoid the use of a shell interpreter by using `pcntl_fork` and `pcntl_exec` within PHP, and similar calls within other languages.
- Implement Python as a web framework for application development instead of PHP. Python directs developers towards using modules that will not invoke the shell. This creates a “secure by default” environment. We recommend frameworks that default to the more secure design pattern of avoiding the shell interpreter altogether.
- Test your web servers for command injection vulnerabilities and your applications for input validation errors on a regular basis using tools such as IBM Security AppScan® software.

It’s true that there has been a downward trend in the number of publicly disclosed command injection vulnerabilities. Nonetheless, attackers continue to exploit existing vulnerabilities in unpatched web servers and applications. Implementation of the practices above should help mitigate command injection attacks.

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

Protect your enterprise while reducing cost and complexity

From infrastructure, data and application protection to cloud and managed security services, IBM Security Services has the expertise to help safeguard your company's critical assets. We protect some of the most sophisticated networks in the world and employ some of the best minds in the business.

IBM offers services to help you optimize your security program, stop advanced threats, protect data and safeguard cloud and mobile. Security Intelligence Operations and Consulting Services can assess your security posture and maturity against best practices in security. An Application Security Assessment can help improve data and network security by assessing your application vulnerability. With IBM Managed Security Services, you can take advantage of industry-leading tools, security intelligence and expertise that will help you improve your security posture—often at a fraction of the cost of in-house security resources.

About IBM Security

IBM Security offers one of the most advanced and integrated portfolios of enterprise security products and services. The portfolio, supported by world-renowned IBM X-Force research and development, provides security intelligence to help organizations holistically protect their people, infrastructures, data and applications, offering solutions for identity and access management, database security, application development, risk management, endpoint management, network security and more. IBM operates one of the world's broadest security research, development and delivery organizations, monitors billions of security events per day in more than 130 countries, and holds more than 3,000 security patents.

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

About the author

David McMillen, Senior Threat Researcher, IBM Managed Security Services. David brings more than 25 years of network security knowledge to IBM. David began his career at IBM over 15 years ago as a member of the core team that created the IBM Emergency Response Service, which eventually grew and evolved into IBM Internet Security Systems.

As an industry-recognized security expert and thought leader, David has a rich background in IT security. He thrives on identifying threats and developing methods of solving complex problems. His specialties are intrusion detection and prevention, ethical hacking, forensics, and analysis of malware and advanced threats. As a member of the IBM Managed Security Services Threat Research Team, David takes the intelligence he has gathered and quickly produces tangible remedies that can be implemented within a customer's network on IBM's own proprietary threat detection engines.

David became interested in security in the 1980s, when he owned and operated one of the first companies to offer penetration and vulnerability testing. As the Internet's footprint grew, it became clear to him that there was a new challenge on the horizon: protecting data. David next worked with IBM Business Partner WheelGroup (later acquired by Cisco), where he helped develop the NetRanger IDS intrusion detection system and NetSonar, a vulnerability scanner. David also assisted with the development of the very first IBM intrusion detection system, BillyGoat. David has subsequently developed several other security-based methods and systems that have been patented by IBM.

Contributors

Scott Craig – Threat Researcher, IBM Security

Michelle Alvarez – Threat Researcher, IBM Security

For more information

To learn more about the IBM Security portfolio, please contact your IBM representative or IBM Business Partner, or visit: ibm.com/security

For more information on security services, visit: ibm.com/security/services

Follow @IBMSecurity on Twitter or visit the IBM Security Intelligence blog

[1] *http://phpsecurity.readthedocs.io/en/latest/Injection-Attacks.html*

Contents

Executive overview

Trends: Shellshock, WordPress and Perlbot

Geography and industry trends

Anatomy of a command injection attack

Protect applications by restricting common operators

Common injection patterns and results

Recommendations

Protect your enterprise while reducing cost and complexity

About IBM Security

About the author

IBM Security

SEL03098-USEN-00