Graham Dungan
February 10, 2025
UIN: 332001764

CSCE 413: Software Security
Self-Study: ShellShock

What is the Vulnerability?
The ShellShock vulnerability is an exploit found within the GNU Bash shell disclosed on the 24th of September, 2014. The ShellShock exploit entails crafting a malicious code snippet that begins with an empty function definition `() { :;};` and finding a way to pass the string as an environment variable. Due to the empty function definition at the beginning of the string, the bash shell will interpret any remaining parts of the string as executable code. The vulnerability had possibly existed years before it had been discovered, and within days of it being disclosed many attacks were using ShellShock, including DDoS attacks and compromising servers.

What was the Root Cause?
As per the design of the Unix operating system, it is common practice to call upon subroutines to handle computation or tasks. This methodology is efficient as software can depend upon common, well-written, and fast subroutines to handle methods that would be otherwise repetitive and arduous to implement.

Because of this standard, however, some subroutines depend upon environment variables as a means of receiving input. This is where the root cause of ShellShock is found- due to a vulnerability in parsing strings in environment variables, any environment variables found containing `() { :;};` (an empty function header) would treat all remaining text as executable code. Due to this vulnerability, ShellShock adjacent exploits would craft malicious strings and input them into programs that depend upon the use of environment variables.

Notable exploitation vectors included CGI-based web servers, OpenSSH, and some DHCP clients. CGI-based web servers included arguments as environment variables, OpenSSH stores information within the environment variable `SSH_ORIGINAL_COMMAND`, and DHCP clients could include additional malicious strings in requesting IP addresses.

What is the Extent of its Impact?
Since the use of ShellShock results in arbitrary code being executed within a shell, reverse shells, data exfiltration, privilege escalation, lateral movement, and botnet infections are all possible.

ShellShock was quickly patched, however, a lack of frequently updating systems rendered a few companies and services vulnerable. Some of the most prevalent attacks resulting from ShellShock were numerous DDoS attacks with Incapsula, CloudFlare, and Akamai reporting millions of attacks worldwide.

How to Patch it?
This code was ultimately patched by changing the format of creating exported functions. Instead of executing all parts of the string as code, only parts of the string enclosed within brackets `{...}` are included. Following, bash has provided new means of sanitizing environment variables during startup (checking for extraneous/malicious data).

How Could it Have Been Prevented?
This is an instance where the proposed vulnerability is the product of a very widely used system having found an oversight that had existed for years. It is a wicked problem in the sense that such vulnerabilities may not be immediately evident or testable before production. In this instance, a vulnerability like ShellShock could have been mitigated with thorough testing and security-oriented design. Sanitization and a 0-trust policy of user input while designing such systems, even if the user of the shell is seemingly trustworthy, should have led the design process.

References

1. Shellshock (software bug) - Wikipedia
2. CVE-2014-6271 - NIST
3. What is Shellshock vulnerability? - Beagle Security
4. The Shellshock Bug In About Four Minutes - Tom Scott, YouTube
5. Shellshock Vulnerability and Attack - Hackrypt, YouTube