**PART 1**

1. Scanning

Never gonna give you up

1. On the protocol of a service

3

**PART 2**

1. Byteless

I get a byte with a little help from my friends

**PART 4**

1. Careless copying

Bits that don’t fit get discarded, segmentation fault, some local var get overwritten, process execution flow change

1. Point taken

DoS, execute malicious code, internal values leaked, Nothing

1. Binary exploit

All except reinstall themselves from scratch

1. Too little too late

Only planning a software with security in mind right from the start is not a very plausible act. It is widely known that modern software are increasingly complex and layered, it is challenging to always think about security from the start. The cybersecurity field is also very vast and ever evolving. New vulnerabilities emerge and it is almost impossible to keep up. Human factors is also a huge downside, as designing processes involves humans which are more than capable to overlook security aspects. We need to always assume that every program - no matter how intricately designed they are - always have bugs to catch. Fuzzing is needed to complement these flaws. It catches unforeseen vulnerabilities that by some chance managed to slip out [2]. It also helps adapt to new vulnerabilities and identify them [3]. It also can help out with the human factor by being unbiased and automated, mitigating oversights [1]. That said, integrating security into design processes is indeed important but by no means should we only rely on it. We need fuzzing to back up those security plans.

References

[1] <https://brightsec.com/blog/fuzzing/>

2] <https://www.code-intelligence.com/blog/what-is-fuzzing-infographic>

[3] https://owasp.org/www-community/Fuzzing

**PART 5**

1. Devops

The DevOps movement is a framework that promotes faster and better application development, where developers may access production systems and be responsible for the maintenance. This deviates from the traditional practice where there usually is a different team from productions and developers. DevOps is generally an innovative solution to exercise a good agility and good delivery to users. Given the benefits, it is indeed a bit too good to be true, since it poses new security challenges. "Traditionally", as aforementioned, developers are unable to access production servers to reduce risks such as misuse or unintentional errors. With DevOps, however, developers gain access which opens a window for an attack if not managed properly [1] - which is natural since the attack surface increases. One of the ways to "mess it up" is the fact that codes and configurations are constantly changed and shifts, which could potentially lead to disruptions or even sensitive data exposures [2] - if not tested well. While fully trusting developers' skills in functional coding, security knowledge is indeed a whole new field and entirely different. Oversights can always happen, especially because DevOps is in a way a "shortcut" which opens an even bigger window for vulnerabilities [3]. Traditionally, there would be multi-layered reviews and approvals before changes are deployed, but DevOps apparently "do not have time for security" [1]. This means they prioritize speed over security and tend to skip security testings [2]. There are many reasons why secure DevOps is failing, such as: security testing is usually not automated nor integrated, false alarms, and resistance from the developers themselves [3]. At this point, it cannot be called oversights anymore. That said, there are also many reasons why DevOps is popular, like its rapid deployment and quick-fixes for bugs and possibly updates [4]. Continuous integration and continuous delivery (CI/CD) which are regular monitoring and feedback loops allows teams to detect issues early on. Business wise, it is tremendously beneficial with the huge amount of automated processes. It is also a good movement when considering product quality, since developers can better understand the market and gather direct feedback to meet user expectations [4]. After all, seeing real-life scenarios does help the developers shape the applications to a better one. All in all, DevOps is actually amazing when we consider the market and productions, since it speeds up traditional ways of working. Identifying and addressing problems are easier and more versatile, but as it opens a path to security problems, should one risk it? I think that one should implement a secure controls and monitoring to be able to both harness the benefits of DevOps and be secure.

References:

[1] https://www.cyberark.com/what-is/devops-security/#:~:text=One%20of%20the%20biggest%20security,highly%20susceptible%20to%20cyber%20attacks.

[2] <https://www.hackerone.com/knowledge-center/devops-security-challenges-and-6-critical-best-practices>

[3] <https://securityintelligence.com/why-isnt-secure-devops-being-practiced/>

[4] https://www.netapp.com/devops-solutions/what-is-devops/