OSU – DADA
Network Security
Homework

Color the following paragraph green for parts you think are still valid, and red for parts you think are no longer valid from our perspective, 35 years later. Add bullet points to justify your opinion.

Robustness Principle: 1980-1989 from RFC-1122 Jonathan Postel, 1989

Once there was a great man, named Postel. See RFC 2468.

1.2.2 Robustness Principle

At every layer of the protocols, there is a general rule whose application can lead to enormous benefits in robustness and interoperability [ref to rfc760, 1980]:

> "Be liberal in what you accept, and conservative in what you send"

Software should be written to deal with every conceivable error, no matter how unlikely; sooner or later a packet will come in with that particular combination of errors and attributes, and unless the software is prepared, chaos can ensue. In general, it is best to assume that the network is filled with malevolent entities that will send in packets designed to have the worst possible effect.

- This point is still very valid in modern software, although often times security can be extracted away from the program developer themselves (and handled by a devops team). We are constantly reminded of the need for secure software with big tech company breaches.

This assumption will lead to suitable protective design, although the most serious problems in the Internet have been caused by unenvisaged mechanisms triggered by low-probability events; mere human malice would never have taken so devious a course!

- With more and more people on the internet, there's more likelihood for malice to play a part in serious problems generated on your internal network. And the opposite is likely true nowadays.

Adaptability to change must be designed into all levels of Internet host software. As a simple example, consider a protocol specification that contains an enumeration of values for a particular header field—e.g., a type field, a port number, or an error code; this enumeration must be assumed to be incomplete. Thus, if a protocol specification defines four possible error codes, the software must not break when a fifth code shows up.

- adaptability and modularity of code/protocols is one of the core principles to being a great software engineer even to this day. If a system you've built can handle serious changes and still work fine, you've built a good system.

An undefined code might be logged (see below), but it must not cause a failure.

The second part of the principle is almost as important: software on other hosts may contain deficiencies that make it unwise to exploit legal but obscure protocol features. It is unwise to stray far from the obvious and simple, lest untoward effects result elsewhere. A corollary of this is “watch out for misbehaving hosts”; host software should be prepared, not just to survive other misbehaving hosts, but also to cooperate to limit the amount of disruption such hosts can cause to the shared communication facility.

- Speaking to the point above, system durability is still vital in today's world especially in microservice architectures.