Dishy Paper

Anonymous Authors

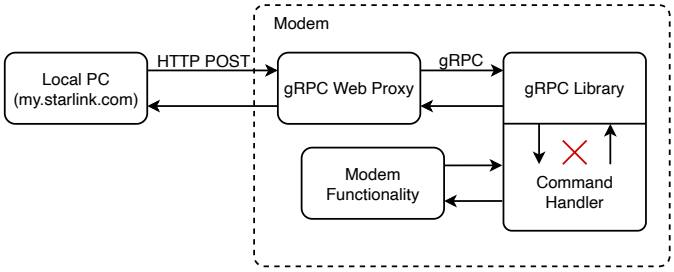

Fig. 1: Overview of the Starlink modem functionality. gRPC calls are encapsulated in HTTP POST requests by the web interface, which are decoded and processed. Malformed gRPC requests cause the command handler to crash, resulting in the modem no longer being able to respond to commands.

***Abstract*—haha yes**

I. Motivation

Test cite [2].

II. Attack

In this section we explore the underlying architecture of the Starlink modem, and how this opens the system up to denial-of-service attacks.

The user terminal is typically administered via the "my.starlink.com" web interface. This interface makes calls to the modem over the local network, using the gRPC (Google Remote Procedure Calls) framework for sending remote commands [1]. These in turn are encapsulated within HTTP "POST" requests, allowing the browser to **TODO ???** Although typically sent using the web interface, these gRPC calls can be made on their own from any device or application on the local network **TODO**. **TODO local gRPC clients, HTTP requests.** They are also unauthenticated, requiring no credentials for commands.

This interface exposes the **(TODO not quite)** full range of administrative commands, extending from telemetry queries through to commands on the dish itself. As a result, an adversary on the local network can trivially cause rudimentary denial of service – for example, by sending the "stow" command to rotate the dish away from the sky, leaving it unable to connect to satellites overhead. By repeatedly sending these commands, service is persistently denied.

When encapsulated within HTTP requests, gRPC commands are very small – the payload is usually between 2 and 5 bytes. This gives a sufficiently small command space for effective fuzzing, since we can send commands of the correct format with random contents to see if any are valid. Through this approach we can find not only valid commands, but also invalid commands that expose corner cases in the command handler, causing unexpected behavior. One such command is the byte sequence `\x00\x00\x00\x00\x03\xea>\x00`, which causes the command handler to crash – this is shown in Figure 1. This stops the modem from responding to commands, but does not stop the terminal from functioning, effectively freezing its settings and state until the terminal is rebooted. A physical power-cycle is required in order to restore functionality.

Appendix **TODO** contains a shell script to send the malformed command to a user terminal on the local network. **TODO maybe picture of error message and/or stowed dish?**

III. Impact

This attack can have a signficant impact – since the state of the dish is frozen, an adversary can achieve persistent denial of service by first sending a command to stow the dish. This will interrupt service until the dish can be physically power-cycled, which is not always trivial. As long as the adversary remains on the local network, this attack can be repeated to cause continuous loss of service for users on the network. Therefore, attackers that can maintain presence on the network will have the greatest impact.

TODO drive-by attacks, potential for remote attacks on non-CORS browsers or through rogue executables

Since the attack can be deployed from any device connected to the local network, large networks containing many untrusted users are at the greatest risk. Such networks also suffer greater impact, as more devices are affected by network disruptions. The impact is magnified when Starlink is the only source of internet access for that customer. Examples may include maritime and aviation traffic, internet cafés, or large organisations.

Restoring service requires physical access to the terminal, so disruption will be increased where access is difficult or restricted. Examples may include secured rooftop installations.

A. Responsible Disclosure

This vulnerability has been reported to Starlink through their provided channels. It has been triaged and reproduced by their security team, and the root cause was determined to be a bug

Network and Distributed System Security (NDSS) Symposium 2023
28 February - 4 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.23xxx
www.ndss-symposium.org

in the gRPC server's handling of edge cases. A fix for this problem will have been fully deployed by the time of this paper's publication.

TODO something about how they suggest the "Guest Network" mode to isolate devices from being able to access the gRPC server.

TODO distinguish from discussion of design issues

We would like to thank the Starlink responsible disclosure team for promptly confirming the issue and deploying a fix, and working with us to **TODO**.

IV. Discussion

Yes

V. Conclusion

Imagine there's a really cool interesting conclusion here

References

[1] gRPC Authors. (2022) About grpc. [Online]. Available: https://grpc.io/about/

[2] J. Rocca. (2021, Mar.) Understanding Variational Autoencoders (VAEs). [Online]. Available: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73