Dishing out DoS: How to Disable and Secure the Starlink User Terminal

Joshua Smailes†
University of Oxford
joshua.smailes@cs.ox.ac.uk

Edd Salkield†
University of Oxford
edd.salkield@cs.ox.ac.uk

Simon Birnbach
University of Oxford
simon.birnbach@cs.ox.ac.uk

Martin Strohmeier
armasuisse Science + Technology
martin.strohmeier@armasuisse.ch

Ivan Martinovic
University of Oxford
ivan.martinovic@cs.ox.ac.uk

† The authors contributed equally to this paper.

***Abstract*—Satellite user terminals are a promising target for adversaries seeking to target satellite communication networks. Despite this, many protections commonly found in terrestrial routers are not present in some user terminals.**

As a case study we audit the attack surface presented by the Starlink router's admin interface, using fuzzing to uncover a denial of service attack on the Starlink user terminal. We explore the attack's impact, particularly in the cases of drive-by attackers, and attackers that are able to maintain a continuous presence on the network. Finally, we discuss wider implications, looking at lessons learned in terrestrial router security, and how to properly implement them in this new context.

I. Motivation

It is well known that router administrator interfaces present an attack surface, allowing attackers to scan for vulnerabilities and reconfigure the router through malicious requests [3], [2]. Unlike in a terrestrial setting, a satellite internet router is often part of a physical system including a motorized dish which can be affected. However, these new routers are being implemented without the institutional memory of historic vulnerabilities and their mitigations, opening up new denial of service attacks through rotating the dish and overusing the motor.

We therefore assess the security of the Starlink user terminal, paying particular attention to the attack surface exposed by its web admin interface. We explore both how requests are made to this interface and the effects of sending undocumented commands, through the use of a fuzzer capable of iterating through the unauthenticated command space. Through this approach we find an exploit in the command decoding and execution logic which, when combined with commands affecting the state of the dish, result in denial of service persisting until the router is physically power-cycled. This can be widely exploited due to poor security practices such as a lack of password authentication on the admin interface, or default passwords on the WiFi network itself.

This vulnerability was reported to Starlink, and a fix has been promptly deployed.

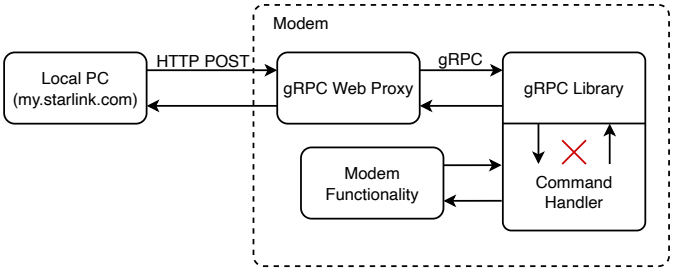


Fig. 1: Overview of the Starlink modem functionality. gRPC calls are encapsulated within HTTP POST requests by the web interface, which are decoded and processed. Malformed gRPC requests cause the command handler to crash, resulting in the modem no longer being able to respond to commands.

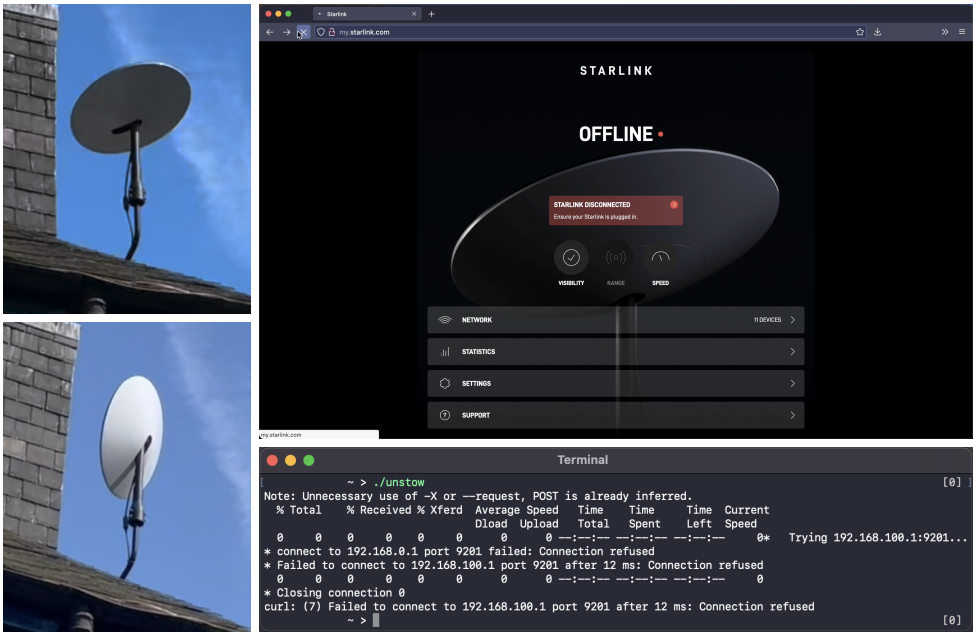
II. Attack

The Starlink user terminal is typically administered via the "http://my.starlink.com" web interface. This sends commands to the modem over the local network, using gRPC (Google Remote Procedure Calls) encapsulated within HTTP "POST" requests. As shown in Figure 1, these requests are decoded by a gRPC web proxy, and forwarded to a command handler.

Although some gRPC commands require password authentication, the vast majority do not, including commands affecting the physical state of the dish.

Since the gRPC payload is usually between 2 and 5 bytes, the command space can be fuzzed with random contents to discover corner cases in the command handler which lead to unexpected behavior. Through this we discovered the "kill" command `00 00 00 00 03 EA 3E 00`, which causes the command handler of the user terminal to crash.

Since the modem will no longer respond to commands, the terminal is frozen in whatever state it was in before the kill command was sent. An attacker can therefore lock the physical dish into a stowed state as seen in Figuree 2, persistently denying service even after the attacker is no longer present on the network. In this state, restoring internet access requires a physical power-cycle.



(a) The dish in "active" and "stowed" modes.

(b) A screenshot of the web control panel error screen following the attack, and the result of sending commands to an inoperative dish.

Fig. 2: The outcome of a successful attack on the Starlink dish, and the resulting web control panel and response to commands.

III. Impact and Mitigations

Since this attack can be deployed from any device connected to the local network, large networks containing many untrusted users are at the greatest risk. Examples may include maritime and aviation traffic, internet cafés, or large organizations. Since the Starlink routers do not password protect the network by default, this is a serious concern.

There is also potential for remote "drive-by" attacks, provided the attacker can in some way cause a device on the same network as the dish to send HTTP requests. This is because executing the attack only requires a few seconds of connection on the local network, and can cause outages on the order of minutes or hours. The Cross-Origin Resource Sharing (CORS) policies of modern browsers prevent javascript from making unauthorized requests to external domains or addresses, so javascript-based attacks are unlikely unless legacy browsers are used [4]. However, the lack of encryption on the connection to "http://my.starlink.com" allows local attackers to hijack the DNS requests or respond with a malicious website to make the request instead. Furthermore, the attacker could trick a user into executing a malicious executable or script, which could easily be used to make these requests.

We argue for the importance of securing these physical systems through satellite router configuration. Even simple mitigations, such as adopting password protection, using local TLS certificates, and the use of HTTP Strict Transport Security [1] would prevent many such attack vectors.

Furthermore, a dedicated administrative wireless network would prevent drive-by attacks, since no device will be connected both to the administrative interface and the public internet. This bears similarities to the "guest network" feature already provided, which provides an internet connection without access to the guest network.

IV. Conclusion

In this work we have explored the security challenges faced by the Starlink router in light of existing work on router security, and uncovered a denial of service vulnerability that abuses the physical properties of the attached dish. This has highlighted the challenges inherent in establishing a secure connection between the browser and router for administrative purposes, whilst maintaining user convenience, which affect secure satellite router design more generally.

Some technical improvements are required, but a significant factor in this is steering users into making well-informed choices to maximize security. These choices include changing administrative passwords, updating TLS certificates, and making use of guest networks to reduce the risk of drive-by attacks. Through good UX design, it is therefore possible to have a polished user experience without sacrificing security.

References

[1] J. Hodges, C. Jackson, and A. Barth, "HTTP Strict Transport Security (HSTS)," Internet Requests for Comments, RFC Editor, RFC 6797, 11 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6797

[2] P. Jeitner, H. Shulman, L. Teichmann, and M. Waidner, "XDRI Attacks – and – How to Enhance Resilience of Residential Routers," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4473–4490.

[3] M. Niemietz and J. Schwenk, "Owning your home network: Router security revisited," *arXiv preprint arXiv:1506.04112*, 2015.

[4] Web Hypertext Application Technology Working Group. (2023) Fetch Living Standard – CORS Protocol. [Online]. Available: https://fetch.spec.whatwg.org/#http-cors-protocol