Security Evaluation on Amazon Web Services' REST API Authentication Protocol Signature Version 4

Khanh Hoang Huynh
Jason Kerssens

Supervisors:
Alex Stavroulakis (KPMG)
Aristide Bouix (KPMG)

UNIVERSITY OF AMSTERDAM

Introduction

- AWS
  - As of 2018, AWS has a dominant share of 47.8% in the cloud service market[1] (IaaS, PaaS)
- Signature Version 4
  - Protocol used for authentication of HTTP API requests
  - Ensures data integrity, verification of the requesting user, and protection against reuse of signed requests
- Other protocols (different functionalities) do not provide end-to-end integrity
  - OAuth 1.0/2.0, SSL/TLS and HTTP Authentication

[1] https://www.ciodive.com/news/iaas-Azure-AWS-Google-Cloud-Alibaba/559716/

Research question

Does the Signature Version 4 protocol, used when sending a request to AWS REST API endpoints, provide data integrity, verification of the requesting user, and protection against reuse of signed requests?

- How does Signatures Version 4 Protocol ensure data integrity, verification of the requesting user, and protection against reuse of signed requests?

- What kind of attacks are able to undermine data integrity, verification of the requesting user, or protection against reuse of signed requests?

Signature Version 4

- Signing key is derived from the secret access key
- HMAC-SHA1 or HMAC-SHA256
- The signature created is a string in hexadecimal and has a length of 64

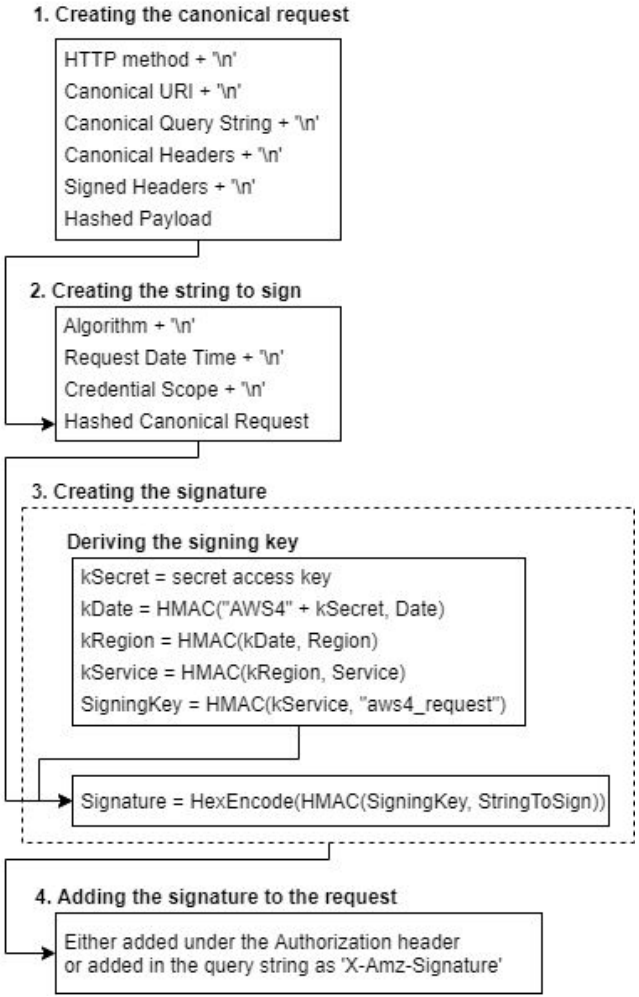

Figure 1: Signature Version 4 signing procedure

Experiments

- Signature Version 4 makes use of HMAC-SHA
    - Attacks on HMAC-SHA are not feasible
- Replay Attack, Modifying Request, HTTP smuggling, and Timing Attacks
- As we look at Signature Version 4, we ignore SSL/TLS
- Attacks were first performed on our local server
- Attacks were then performed on AWS IAM and S3 services

Used Technologies

- Python

- Escher

- Burp

Python Logo source: https://www.python.org/
Emarsys Logo (Escher Creator) source: https://www.emarsys.com/
Burp Suite logo source: https://portswigger.net/

Replay Attack

- Protection against reuse of signed requests
- Intercept request and resend
- For how long?
- What kind of requests?

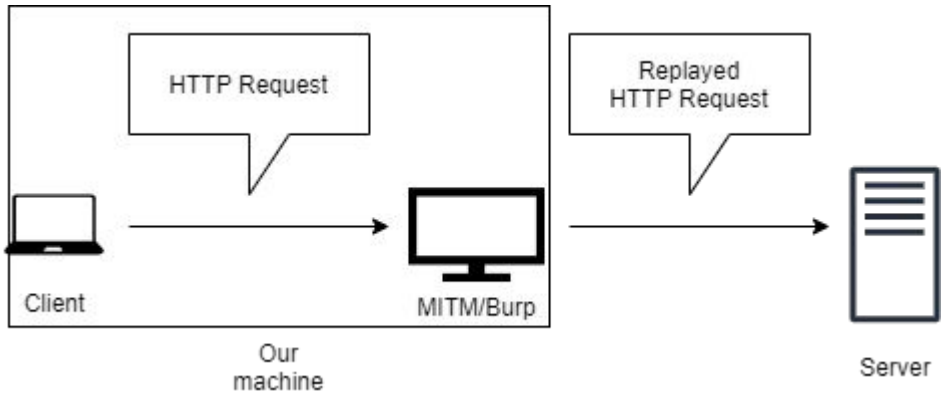

Figure 2: The setup of our replay attack

Modifying the request

- Ensurance of data integrity
- Intercept request, modify it, and send it to intended destination
- What parts of the request can be modified?

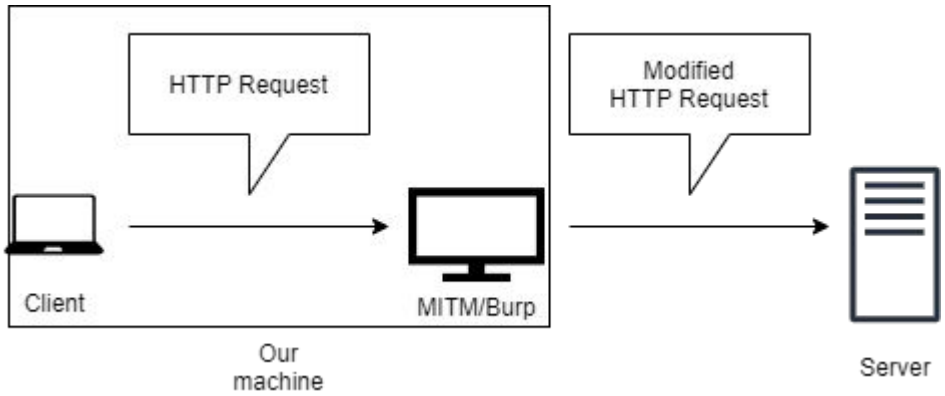

Figure 3: The setup of our modifying the request

HTTP Smuggling

- Verification of the requesting user
- Discrepancy in front-end server and back-end server

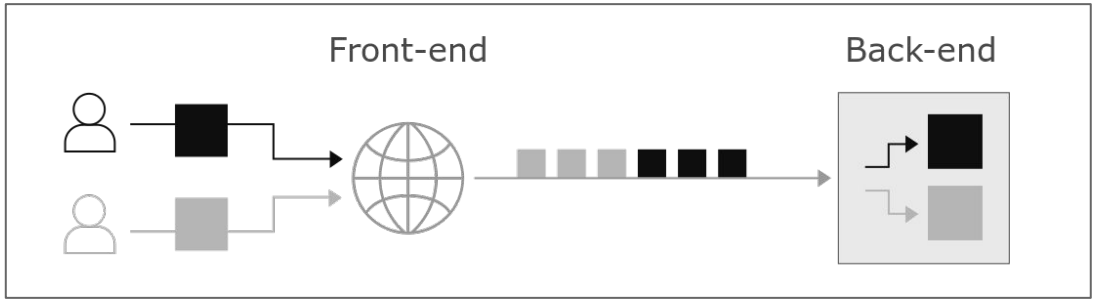

Figure 4: The request flow of modern website architecture. Source: https://portswigger.net/

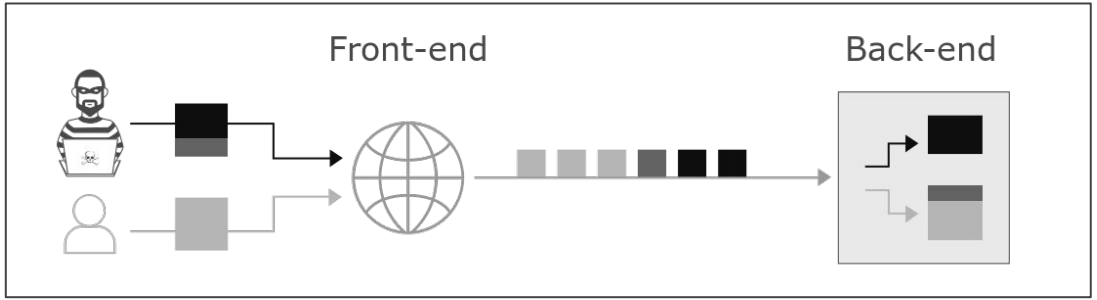

Figure 5: Example of a HTTP request smuggling attack. Source: https://portswigger.net/

HTTP Smuggling

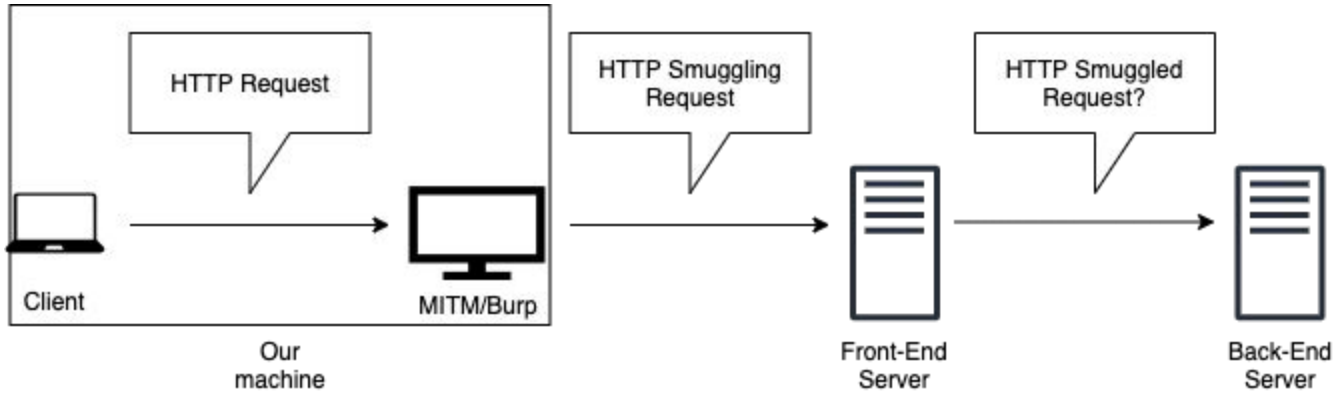

Figure 6: The setup for HTTP smuggling attacks

Probing Timing Attack

- Verification of the requesting user
- Side-channel attack on Signature of HTTP request
- Measure execution time of request and response
- Correlation between execution time and number of valid bits
- Implementation dependent

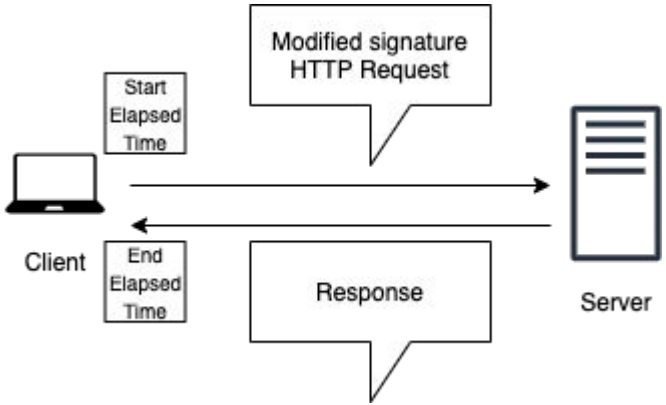

Figure 7: The setup of our Probing Timing attack

Example:

‘aaaa’ != ‘aaaa’
‘aaaa’ != ‘aabb’

Probing Timing Attack Experiment in detail

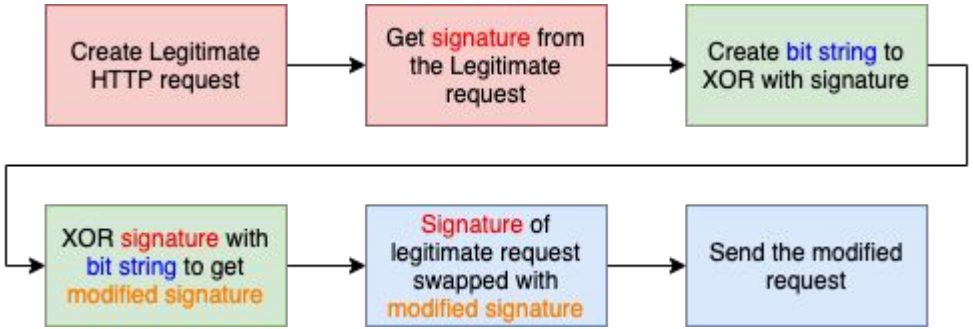

Figure 8: Flow of how we manipulated the signature

http://localhost:5000/validate_request?foo=bar&abc=cba&X-EMS-Algorithm=EMS-HMAC-SHA256&X-EMS-Credentials=EXAMPLEAPIKEY%2F20200202%2Fcredential%2Fscope&X-EMS-Date=20200202T143407Z&X-EMS-Expires=1209600&X-EMS-SignedHeaders=host&X-EMS-Signature=e5990c4ea65b5e382f4ce5356b03ca5bf5df4c07be06f84224b7ac4d96dfeda2

1110...0010 ⊕ 0000...0001 = 1110...0011

Figure 9: An example of changing one bit of the correct signature

http://localhost:5000/validate_request?foo=bar&abc=cba&X-EMS-Algorithm=EMS-HMAC-SHA256&X-EMS-Credentials=EXAMPLEAPIKEY%2F20200202%2Fcredential%2Fscope&X-EMS-Date=20200202T143407Z&X-EMS-Expires=1209600&X-EMS-SignedHeaders=host&X-EMS-Signature=e5990c4ea65b5e382f4ce5356b03ca5bf5df4c07be06f84224b7ac4d96dfeda3

Results (replay attack)

- Replaying of requests possible
- Default valid for 15 mins for IAM
- X-AMZ-Expires option for S3
- Prevented by SSL/TLS

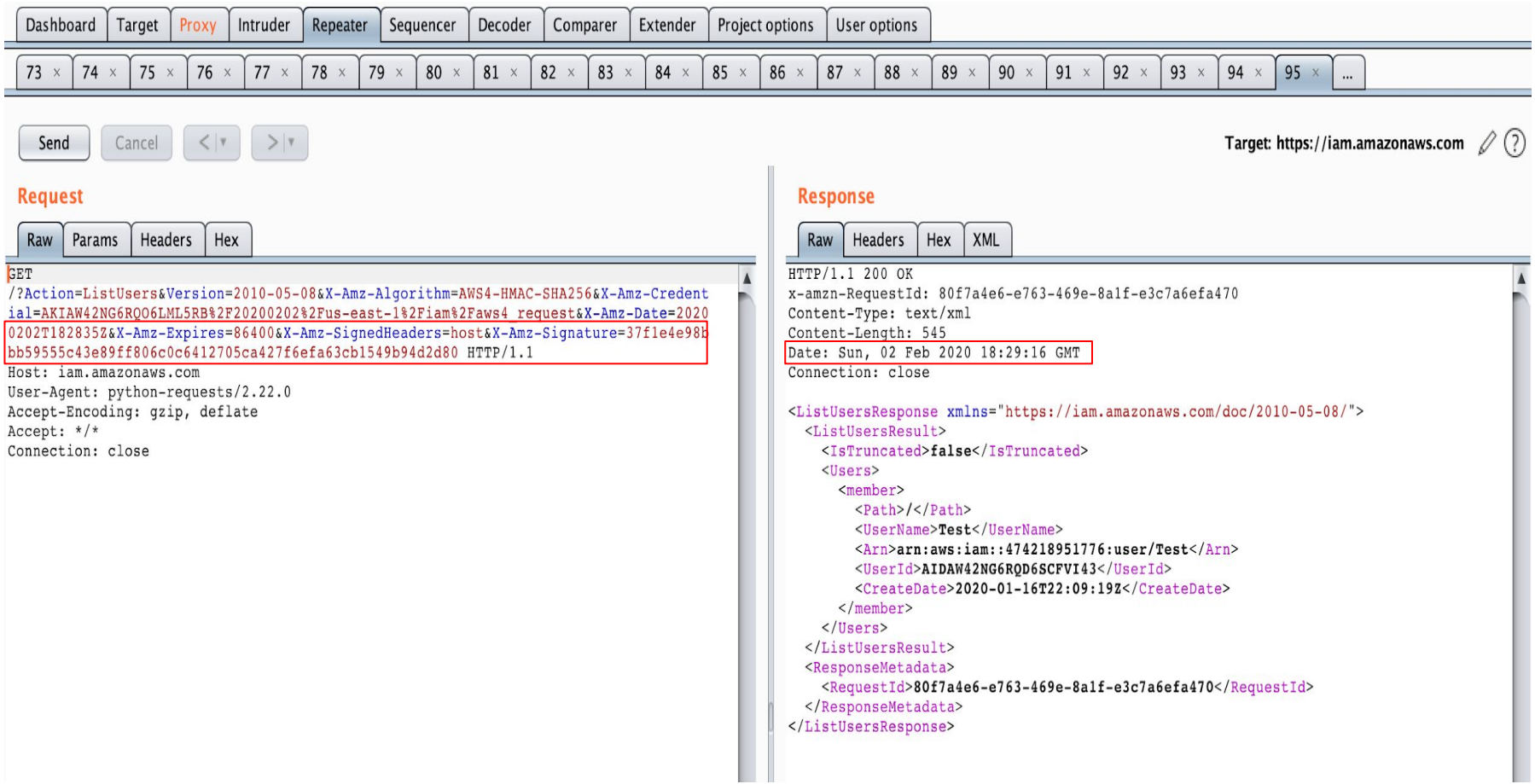

Figure 10: HTTP Request and response

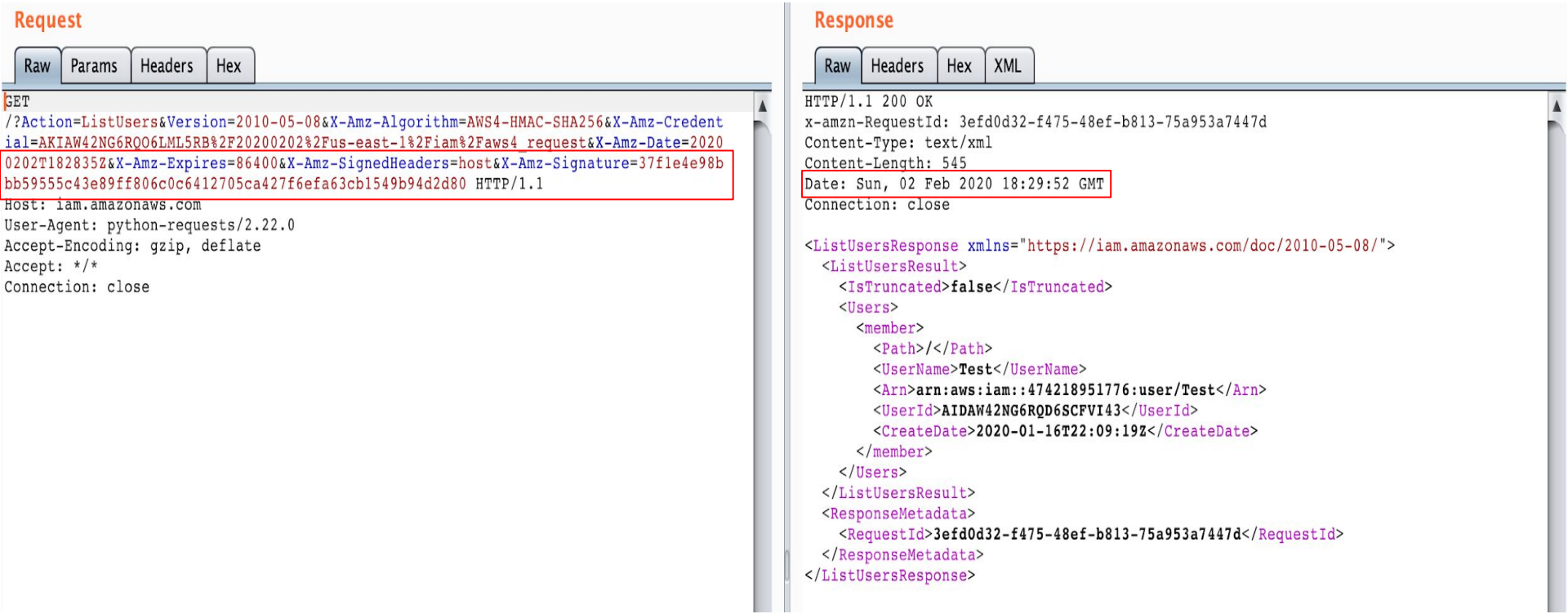

Figure 11: Replayed a HTTP Request and response

Results (modifying requests)

- Signed parts cannot be changed
- S3 unsigned payload option
- Prevented by SSL/TLS

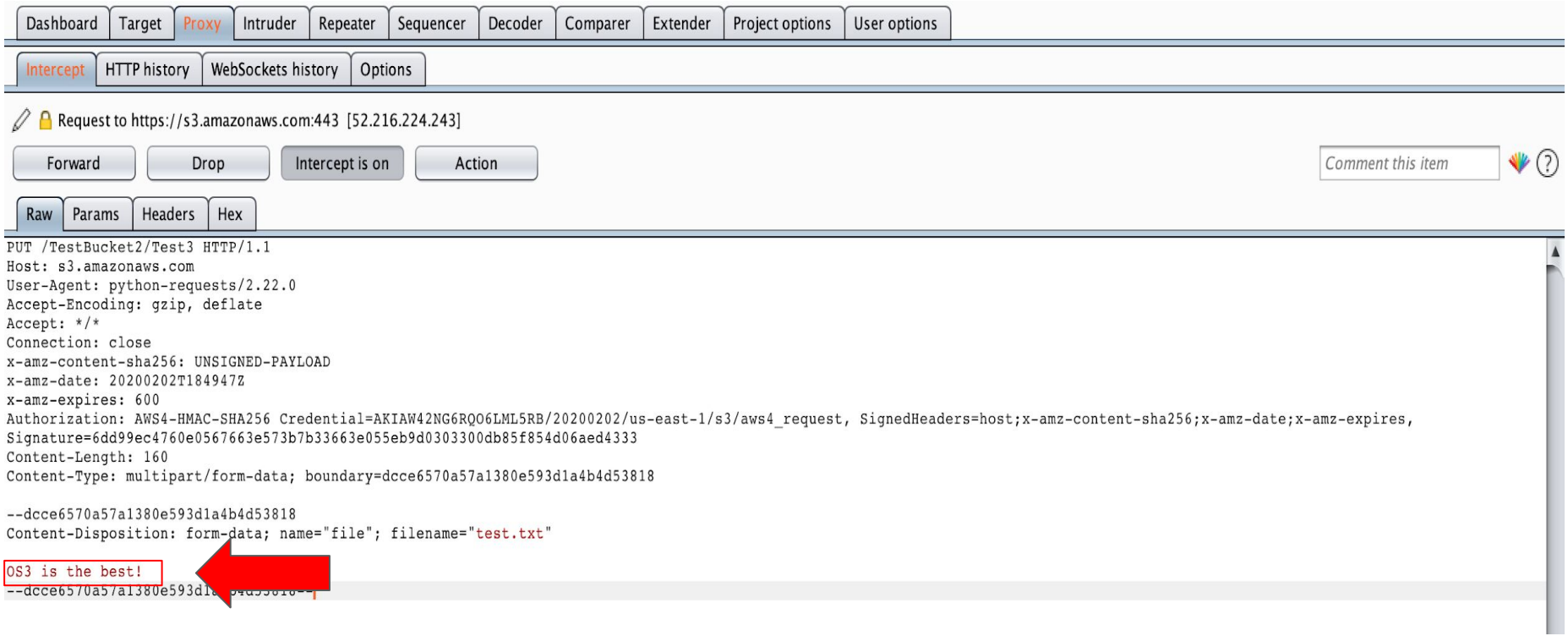

Figure 12: HTTP request payload to be modified

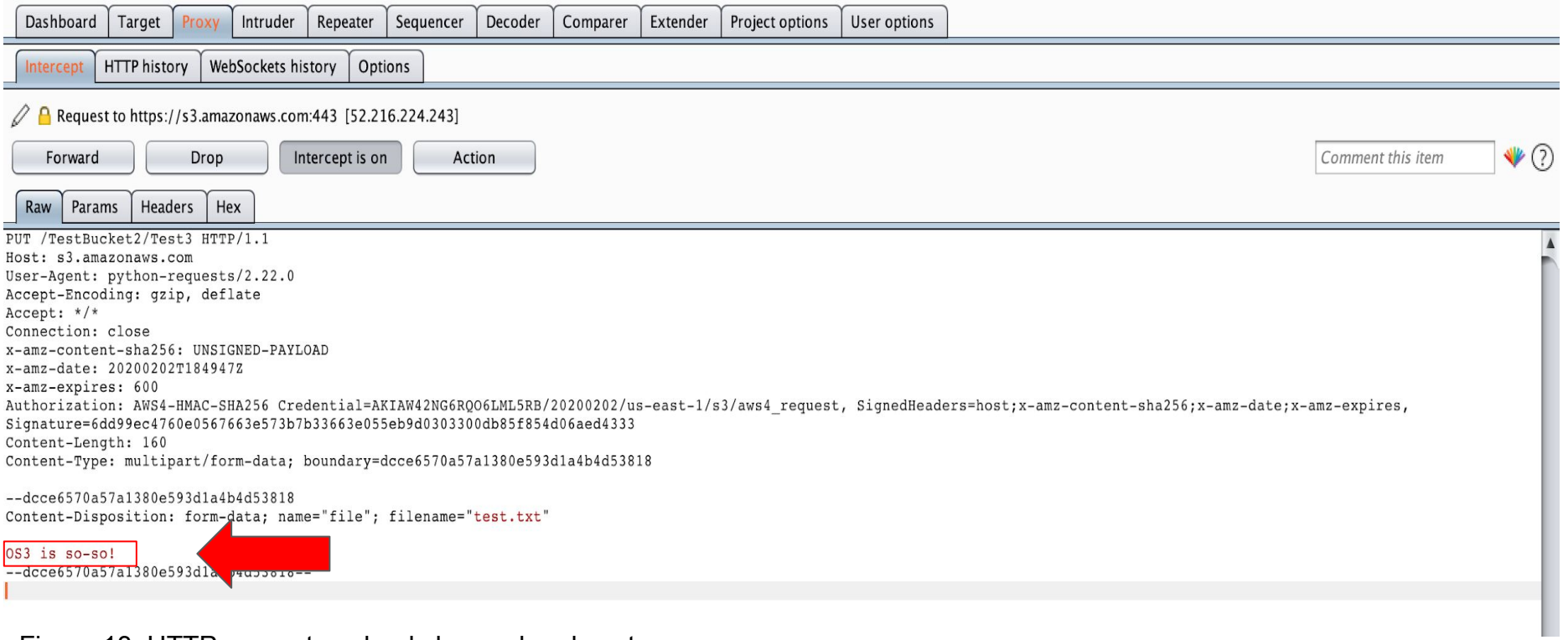

Figure 13: HTTP request payload changed and sent

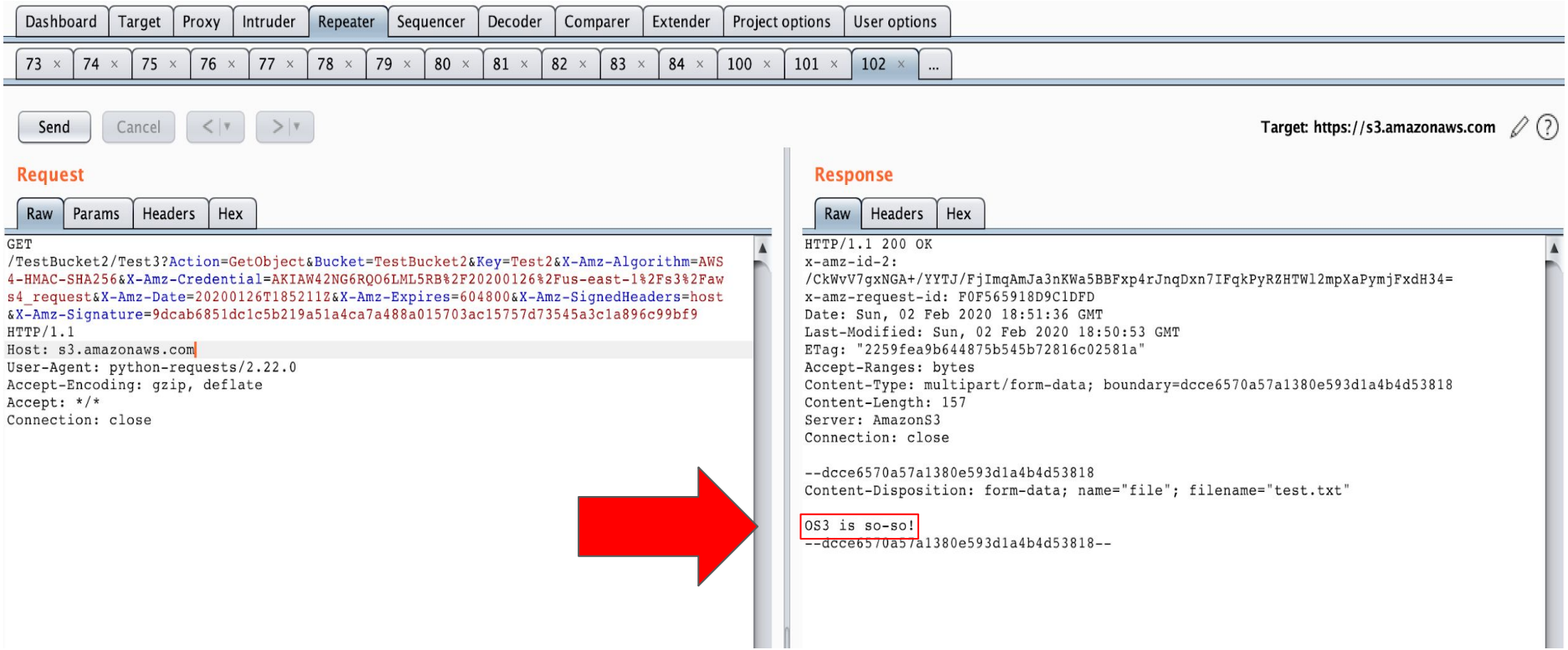

Figure 14: Successfully modified HTTP request and uploaded to AWS

Results (HTTP Smuggling)

- Not successful, as AWS responds with HTTP Status Code 500

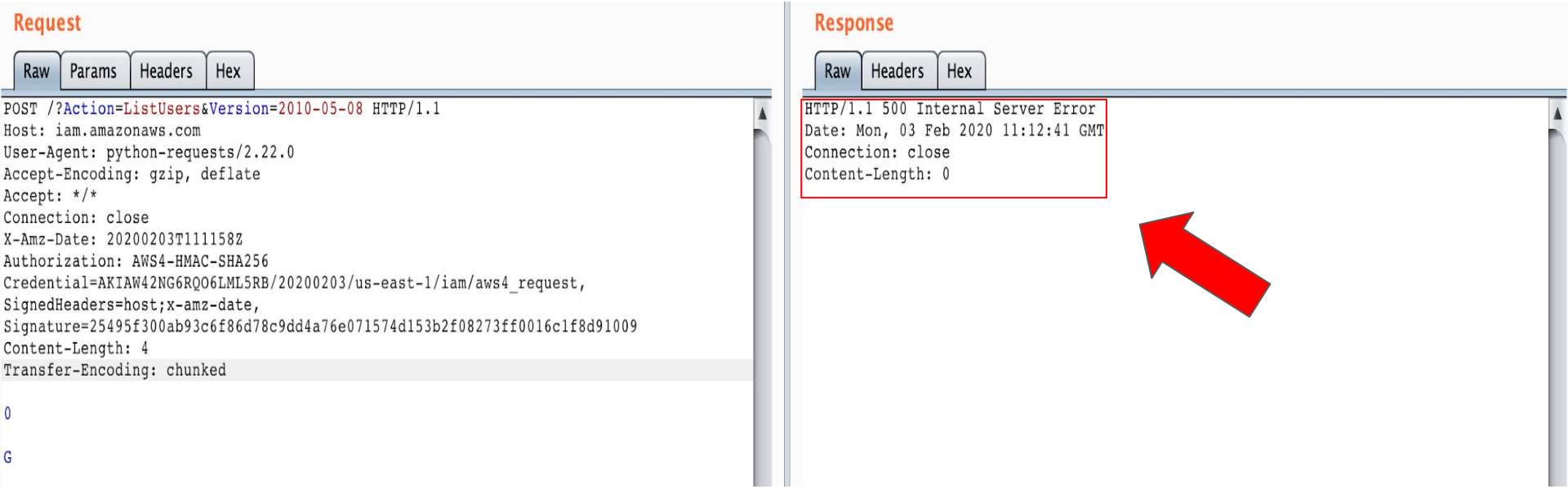

Figure 15: Result of executing the detecting if http smuggling is possible

Results (Timing attack)

- Escher
- Correlation found? (between execution time and number of correct bits in signature)

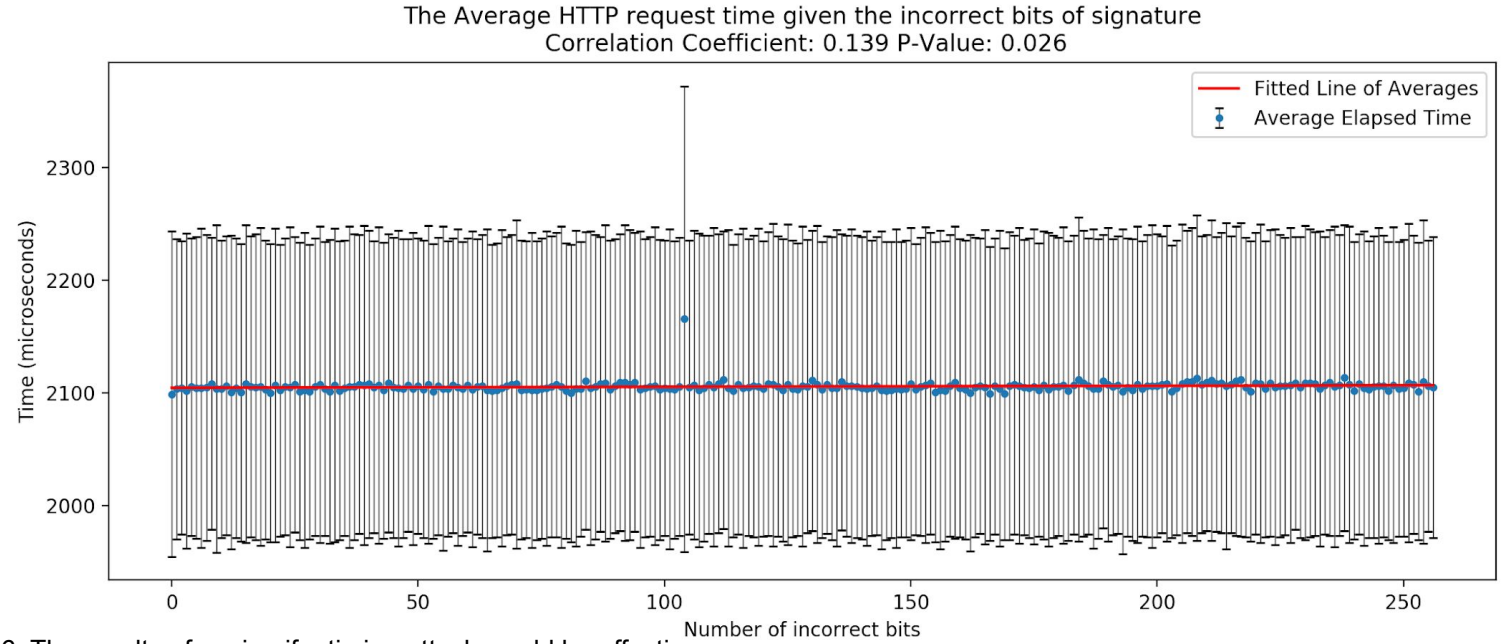

Figure 16: The results of seeing if a timing attack would be effective

Results (Timing attack)

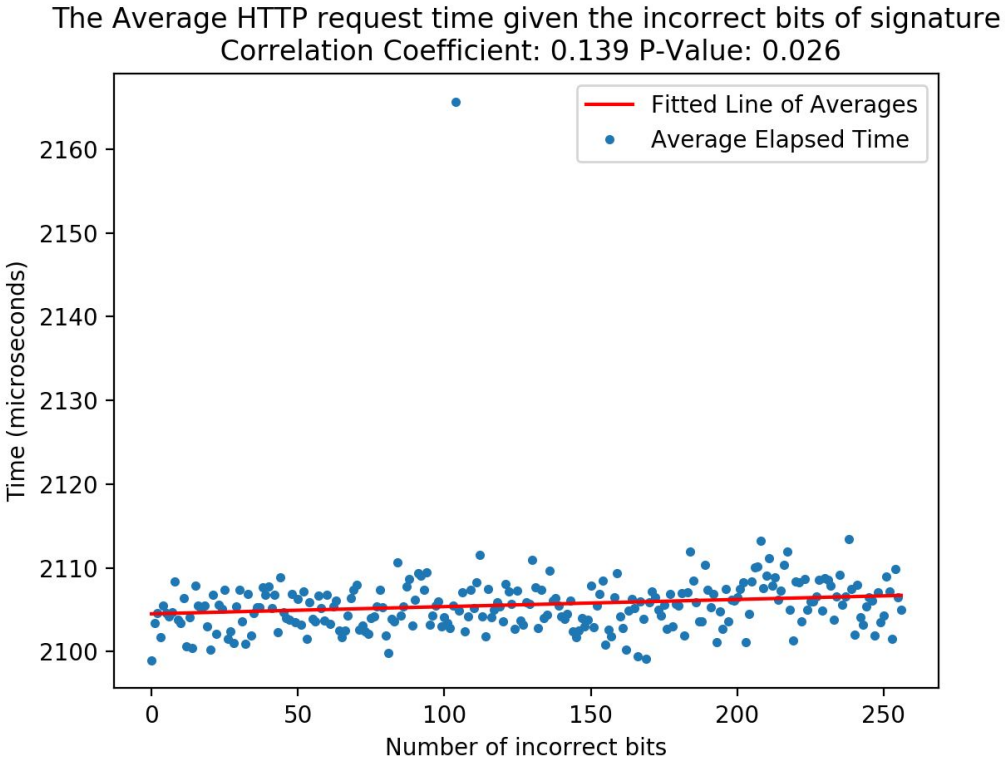

Figure 17: Figure 15, but without the standard deviation plotted

Conclusion

How does Signature Version 4 ensure protection?

- Data integrity: Signature
- User verification: API KEY ID, and Secret Access Key
- Reuse of signed portions: Expiration of request

What kind of attacks are possible?

- Replay attack: reuse of signed portions is possible for a limited time
- Modifying requests: signed portions of requests cannot be modified, unsigned portions can be modified
- HTTP Smuggling: not successful
- Timing attack: correlation found locally

Conclusion

Does the Signature Version 4 protocol, used when sending a request to AWS REST API endpoints, provide data integrity, verification of the requesting user, and protection against reuse of signed requests?

- Provides data integrity of signed portions
- Verifies that signed parts were indeed signed by user
- Does not fully provide protection of reuse of signed portions

Future work

- Other services
- Timing attack on AWS servers
- Inspect the SSL/TLS from AWS API endpoint

Conclusion

Does the Signature Version 4 protocol, used when sending a request to AWS REST API endpoints, provide data integrity, verification of the requesting user, and protection against reuse of signed requests?

- Provides data integrity of signed portions
- Verifies that signed parts were indeed signed by user
- Does not fully provide protection of reuse of signed portions