Email

Email

- Electronic Mail
  - It's like mail, but electric
- Let's talk about how email travels through the Internet

SMTP

- SMTP (Simple Mail Transport Protocol)
    - Defines how email is *transported*
    - Delivers mail in an "envelope" and is not concerned with the content or format of the message it contains
- Transport is handled by multiple SMTP servers

SMTP

user@example.com
example.com SMTP Server
website.com SMTP Server
user@website.com

- user@example.com wants to send an email to user@website.com

SMTP

user@example.com
example.com
SMTP Server
website.com
SMTP Server
user@website.com

- user@example.com drafts a wonderful email using their Mail User Agent (MUA)
  - MUA is your email client (eg. Outlook, Gmail, a browser, or whatever software you use to access your email)
- They click send when they are done writing the email

SMTP

user@example.com
example.com SMTP Server
website.com SMTP Server
user@website.com

- The user's MUA sends the email to the SMTP server for example.com which plays 2 roles:
  - Mail Submission Agent (MSA) - Responsible for accepting sent messages
  - Mail Transfer Agent (MTA) - Responsible for sending the email over the Internet

SMTP

user@example.com
example.com SMTP Server
website.com SMTP Server
user@website.com

- The sending server's MTA performs a DNS lookup to find the IP address of the recipients SMTP using the domain of the email address
- Send the email to this SMTP server

SMTP

user@example.com
example.com SMTP Server
website.com SMTP Server
user@website.com

- The recipient server plays multiple roles:
  - Mail Exchanger (MX) - The entry point for a delivered message
    - This is the server listed in the DNS record
  - Mail Delivery Agent (MDA) - Responsible for receiving and storing email messages

SMTP

user@example.com
example.com
SMTP Server
website.com
SMTP Server
user@website.com

- The recipient then uses their email client (MUA) to access their email from the MDA
  - Using email protocols IMAP and/or POP
- Reads the email in their MUA

SMTP

user@example.com
example.com SMTP Server
website.com SMTP Server
user@website.com

- Security Concern
- How do the SMTP servers verify that an email is legitimate?

SMTP

user@example.com
example.com
SMTP Server
website.com
SMTP Server
user@website.com

- Security Concern
- The sending server can use a login system authenticate the sending user

SMTP

user@example.com
example.com SMTP Server
website.com SMTP Server
user@website.com

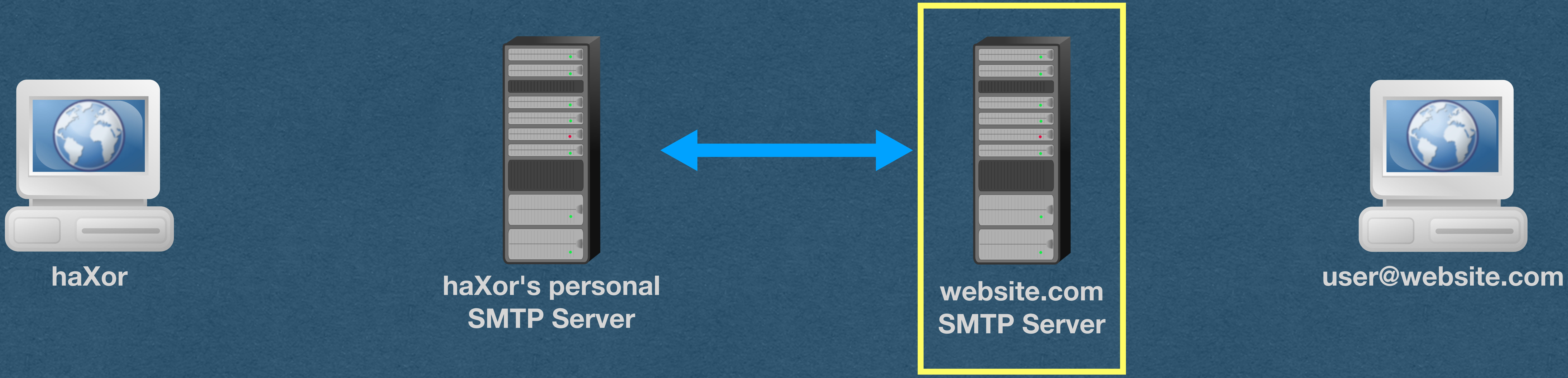
- Security Concern
- The receiving server has a more difficult task
- How does it know the server sending the email has the authority to send email on behalf of this user?

SMTP

user@example.com
example.com SMTP Server
website.com SMTP Server
user@website.com

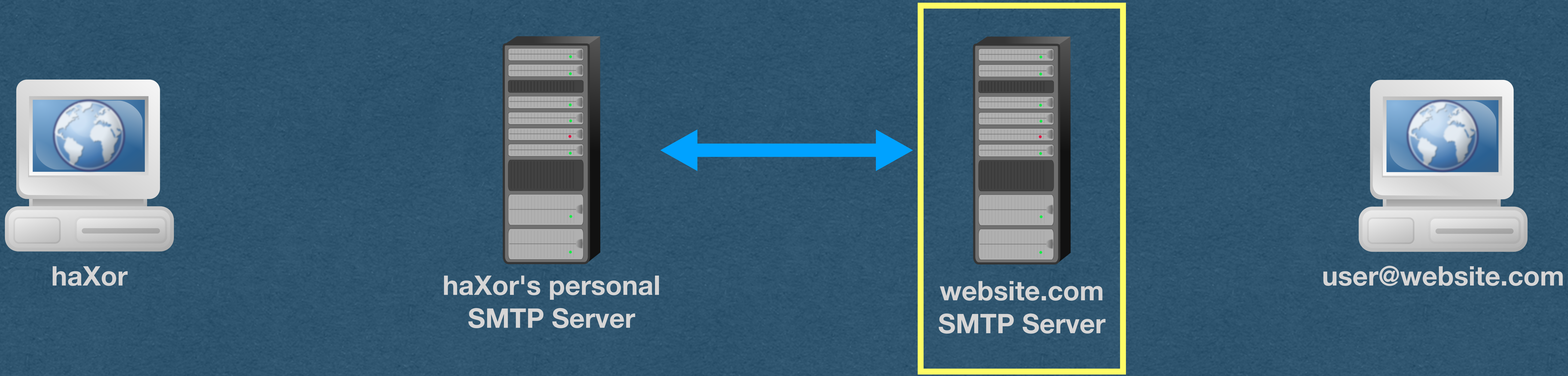
- Security Concern
- SMPT was defined before security on the Internet was a concern
  - The protocol does not help us here..
- Each SMTP server has to come up with its own way of verifying emails

SMTP



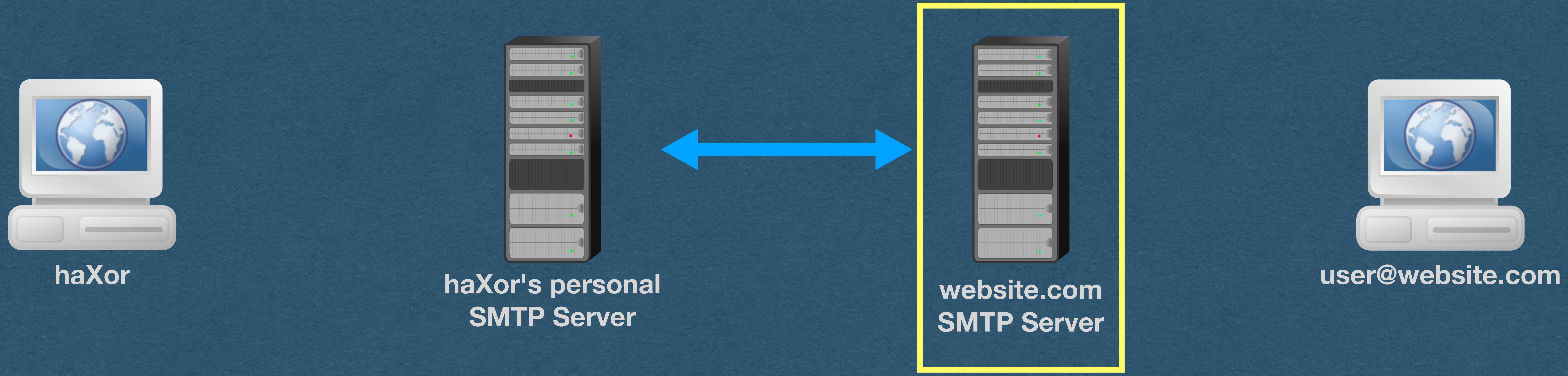
- Security Concern
- Example: An attacker runs their own SMTP server and sends email with your email address in the "from" field
  - Nothing stops them from doing this
  - It's up to the receiving server to detect this fraud

SMTP



- Security Concern
- Many servers verify that the sending SMTP server comes from a trusted IP address for that domain
- Can also monitor for suspicious behavior and ban IP addresses
- Unverified emails should not be trusted and may be blocked, marked as spam, or send with security warnings

SMTP



- Security Concern
- Due to the lack of verification in the protocol, it is difficult to run your own SMTP server

Email and the Project

- Intended to use the Gmail API
  - Allowed to use any other solution that works securely (You cannot have security/spam warning when we receive an email from your server)
- Gmail API uses OAuth 2.0
- It's ok to generate your credentials manually
  - Only need to obtain an access token for your own account
- Credentials must be kept secure (Not stored in a public repo)
  - Common to use placeholders, then you replace only in production
  - Can use environment variables that are manually set in dev and prod

Email Verification

- Email verification
  - Generate a random high-entropy token and send it in a url
  - If you get a request for that path, you've verified the email since no one can guess that much entropy and you only sent the link to that email
  - Since the token is in the url, it is sent on a GET request
    - User only needs to click the link to verify (No JS/AJAX/forms/etc. required)
    - Easy to send a link in an email

Email Tracking

- Send a unique url in an email
  - This is also used for tracking
  - Send an ad in an email and give everyone a unique link. Now you know what email addresses belong to people who click ads (Valuable information)
    - These email lists can be sold
      - Click one ad -> get more spam