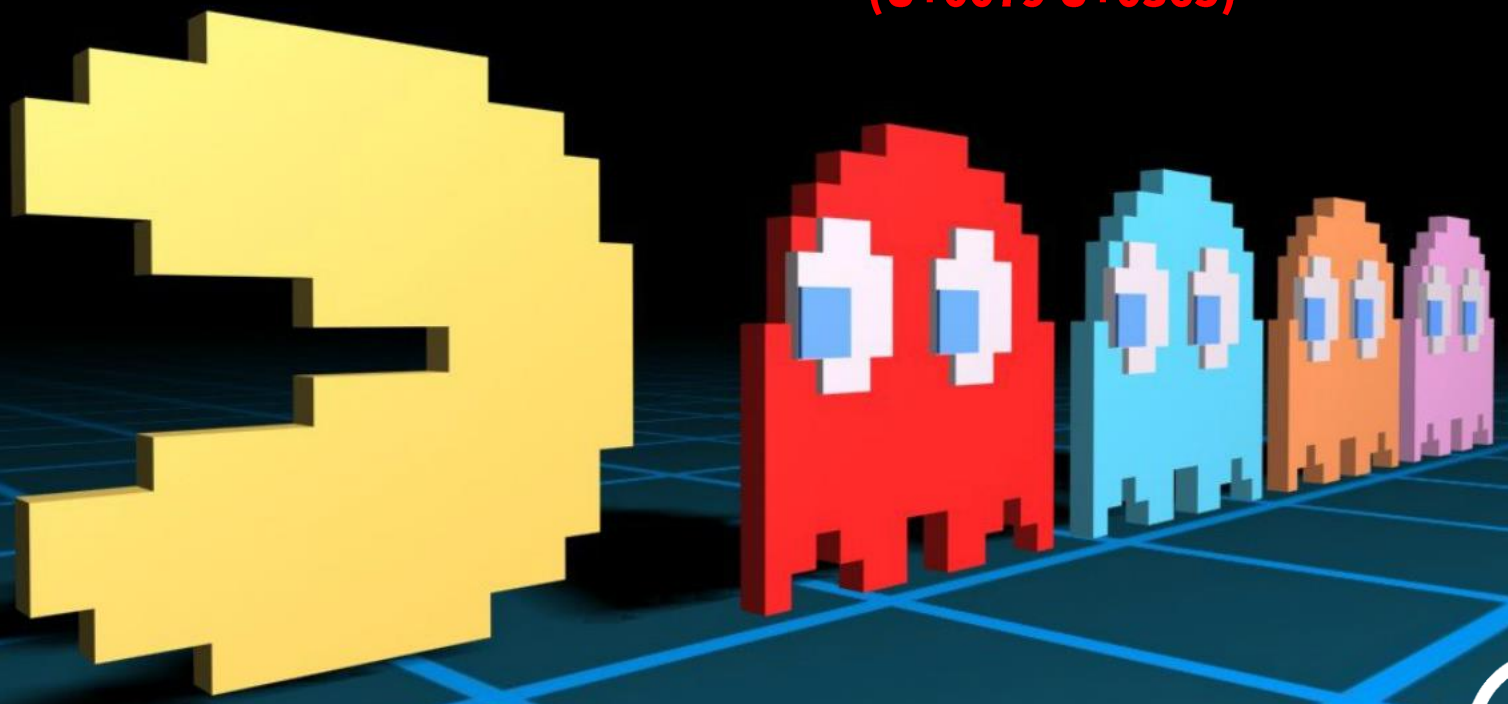
Unicode Vulnerabilities That Could B i̇te You

(U+0079 U+0365)



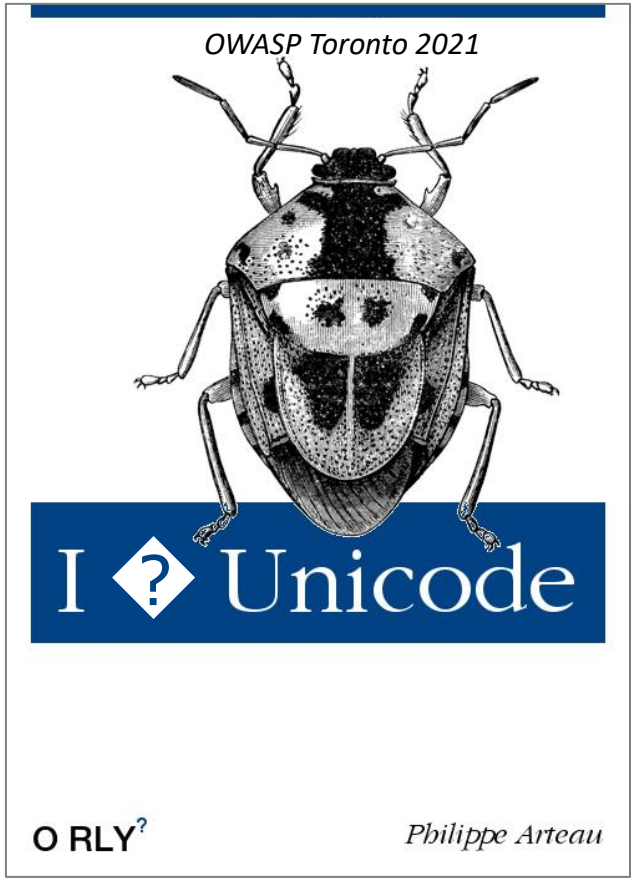
Presented by Philippe Arteau from GoSecure

OWASP® Toronto

Agenda

- Small history of encoding

- Normalization
- Case modification
- Bypassing filters
- Homograph attacks
- Data integrity



GoSecure

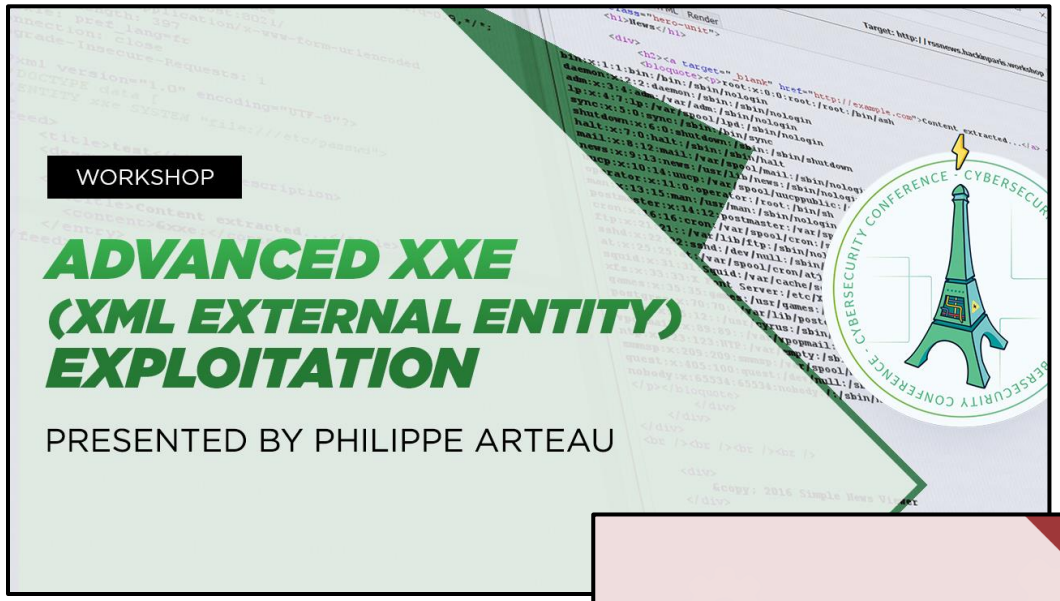
Bio

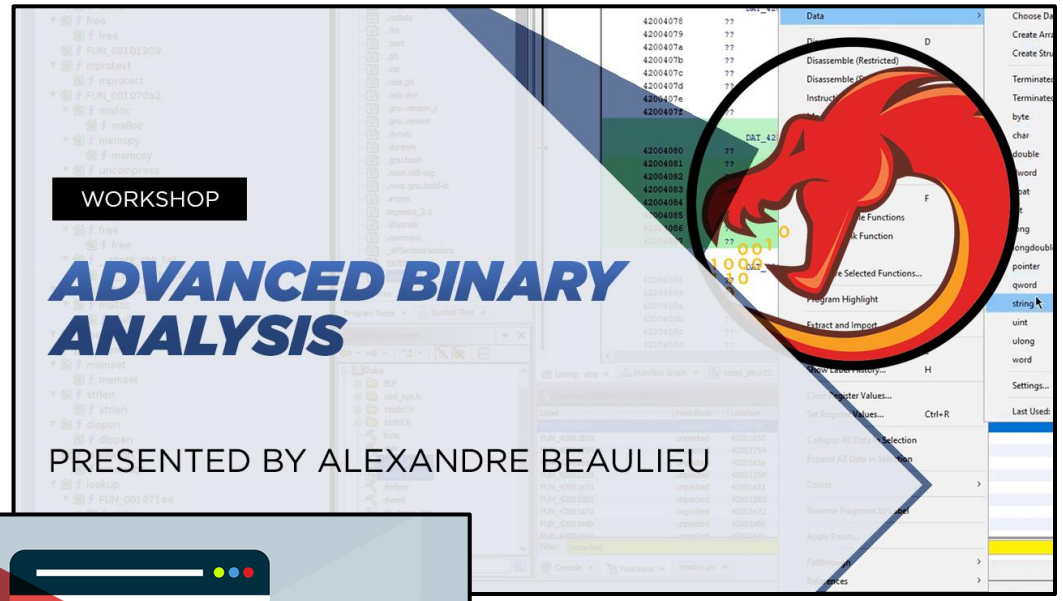
- Philippe Arteau
- Security Researcher at 

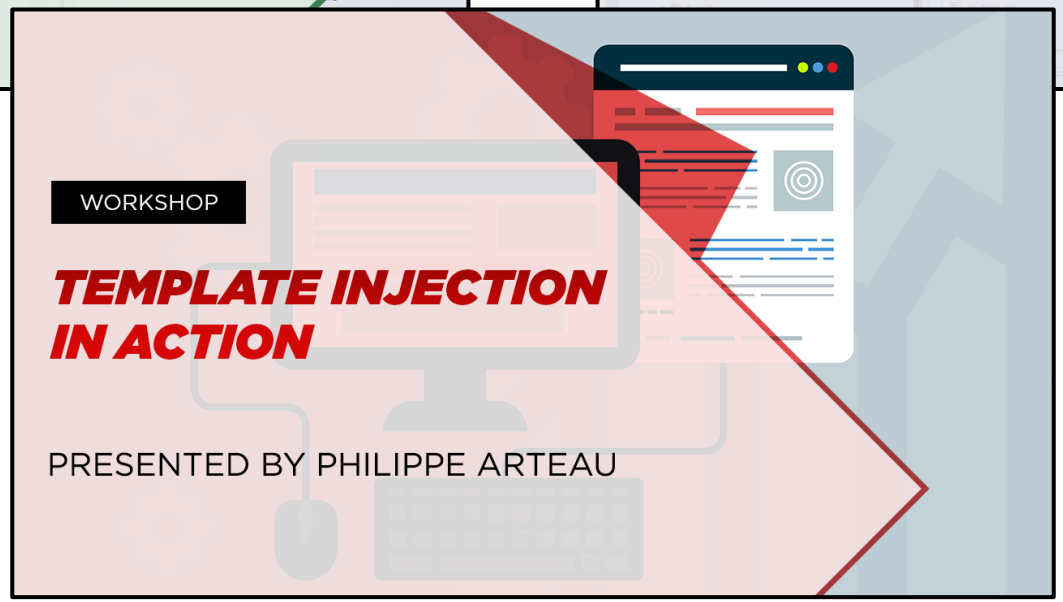
- Open-source developer
  - Find Security Bugs (SpotBugs - Static Analysis for Java)
  - Security Code Scan (Roslyn – Static Analysis for .NET)
  - Burp and ZAP Plugins (Retire.js, CSP Auditor)

- Volunteer for the 

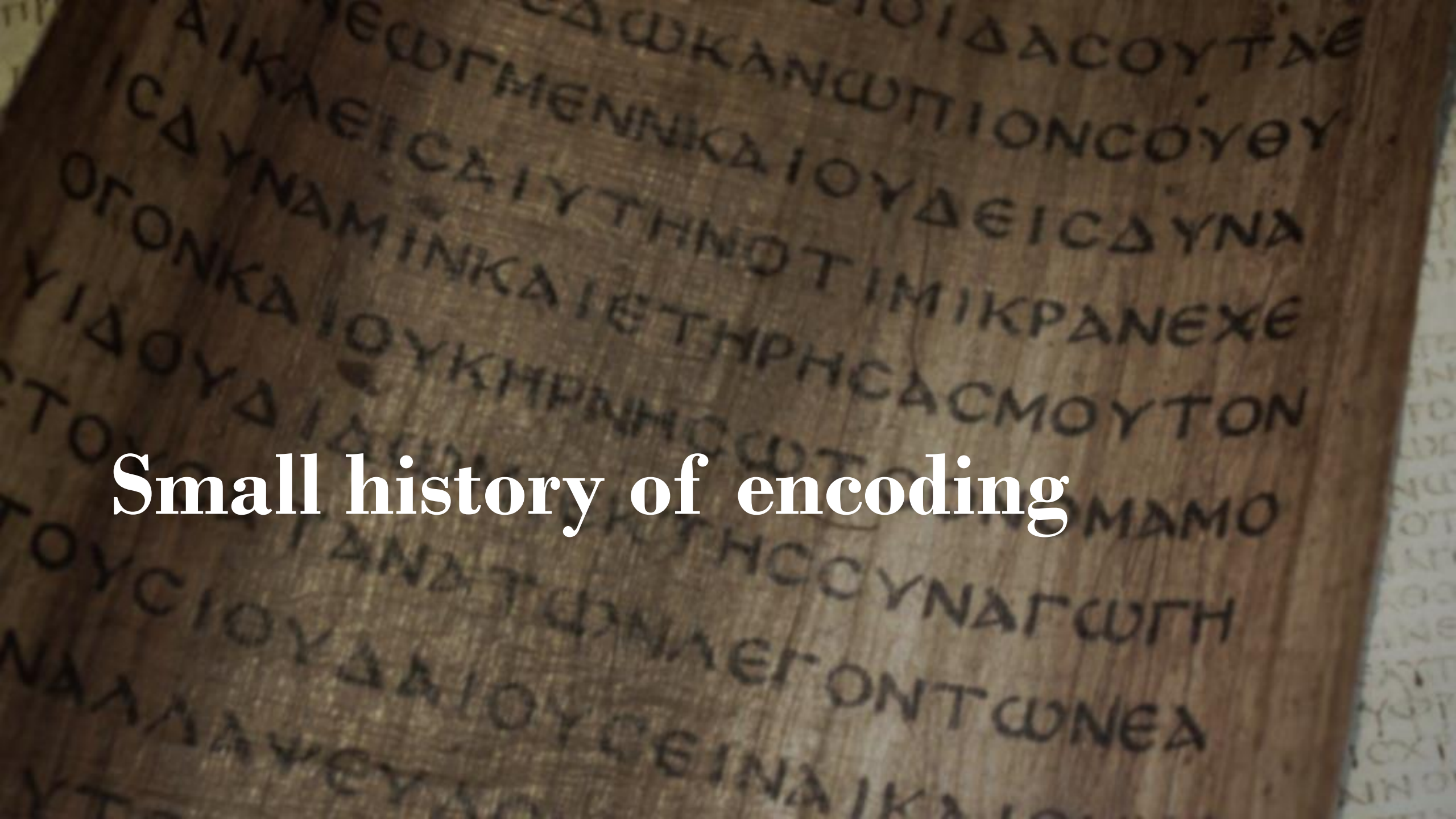
conference and former trainer


WORKSHOP
ADVANCED XXE
(XML EXTERNAL ENTITY)
EXPLOITATION
PRESENTED BY PHILIPPE ARTEAU


WORKSHOP
ADVANCED BINARY
ANALYSIS
PRESENTED BY ALEXANDRE BEAULIEU


WORKSHOP
TEMPLATE INJECTION
IN ACTION
PRESENTED BY PHILIPPE ARTEAU

GoSecure



Small history of encoding

ASCII

Created in the 1960s

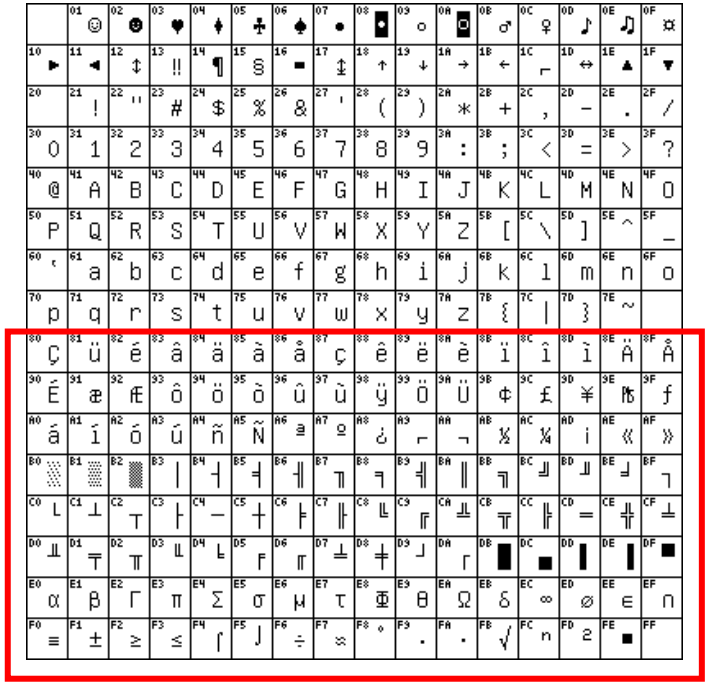
- Control Characters (0-31)
  - Null (0), Bell (7), Backspace (8), EOF (26)
- Standard Character Set (32-127)
  - A…Z, a-z,
- Extended character Set (128-255)
  - Accented characters, symbols, box characters

GoSecure

Code Pages

Attempt to standardize ascii implementation to support special characters

- IBM-PC / DOS Latin US : Code page 437
- Greek : Code page 737
- Cyrillic : Code page 855
- ANSI / Windows-1252 : Code page 1252
- …



More code pages : https://www.ascii-codes.com/

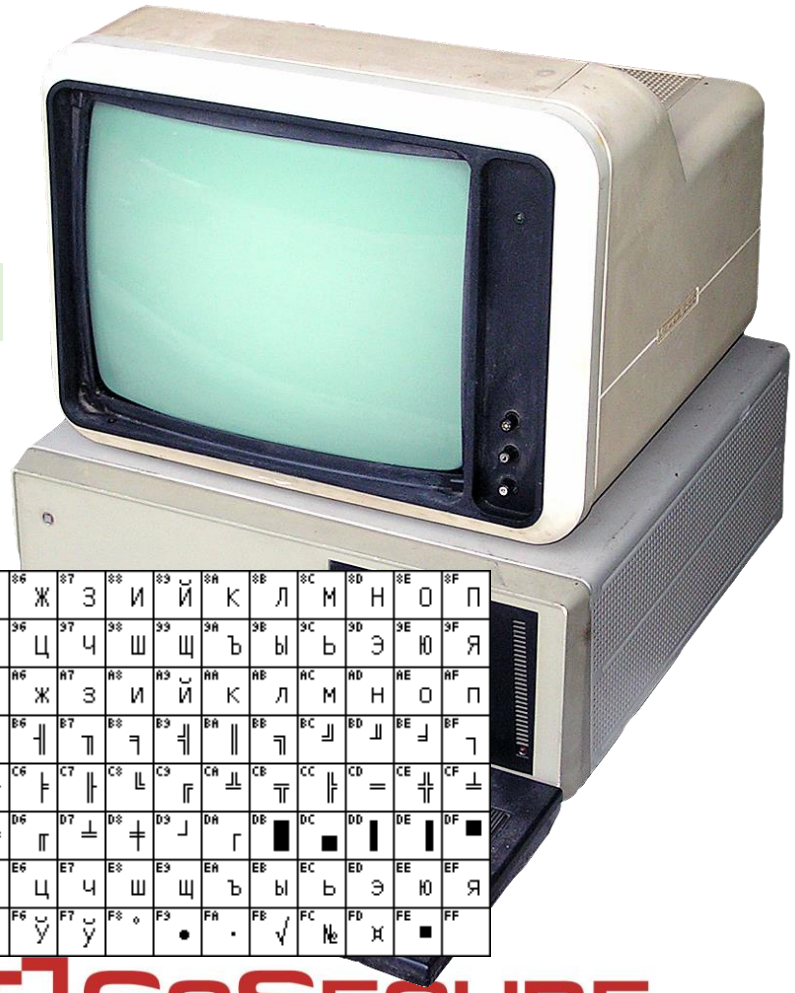
GoSecure

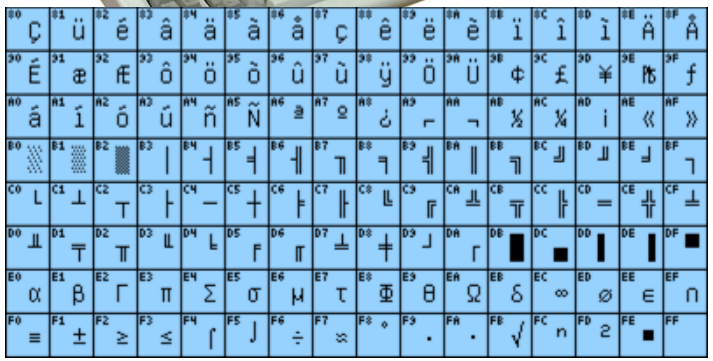
Unexpected Encoding

Déjà vu_

D\x82j\x85 vu

DBjE vu

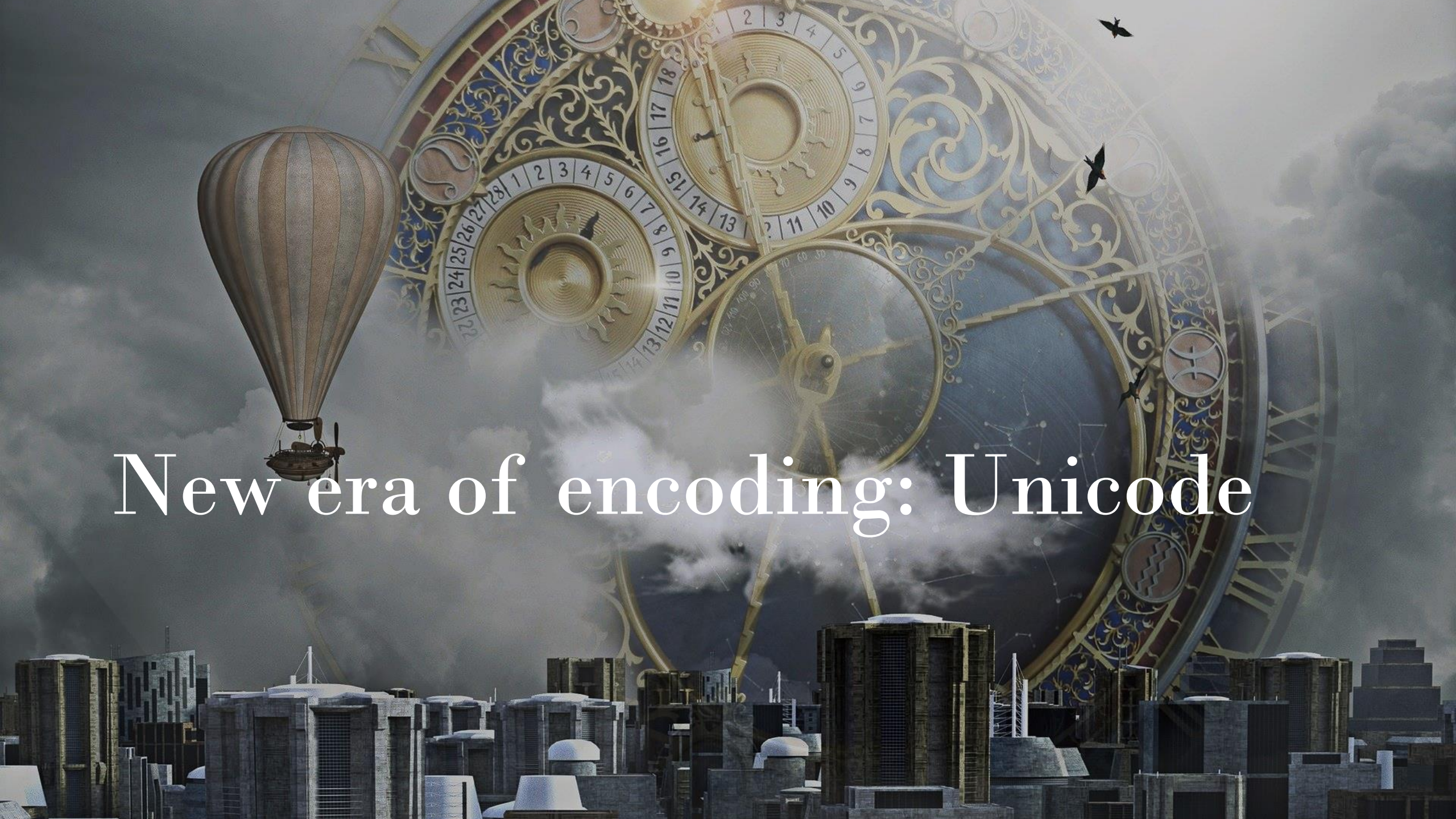




CP-437
DOS Latin US

CP-866
DOS Cyrillic Russian

GoSecure



New era of encoding: Unicode

Unicode Code Points

- Uniquely index “all” characters : Latin, Cyrillic, Asian, African script, Mathematical symbol, Measuring Unit, Egyptian Hieroglyphs and Emojis.

A	é	水	∑	
U+0041	U+00e9	U+6C34	U+2211	U+ 1f4a9

- Code Points **do not** define how it should be **encoded** (bytes representation)

To get detailed CP info:
https://www.fileformat.info/info/unicode/char/1f4a9/index.htm

GoSecure

UTF-8

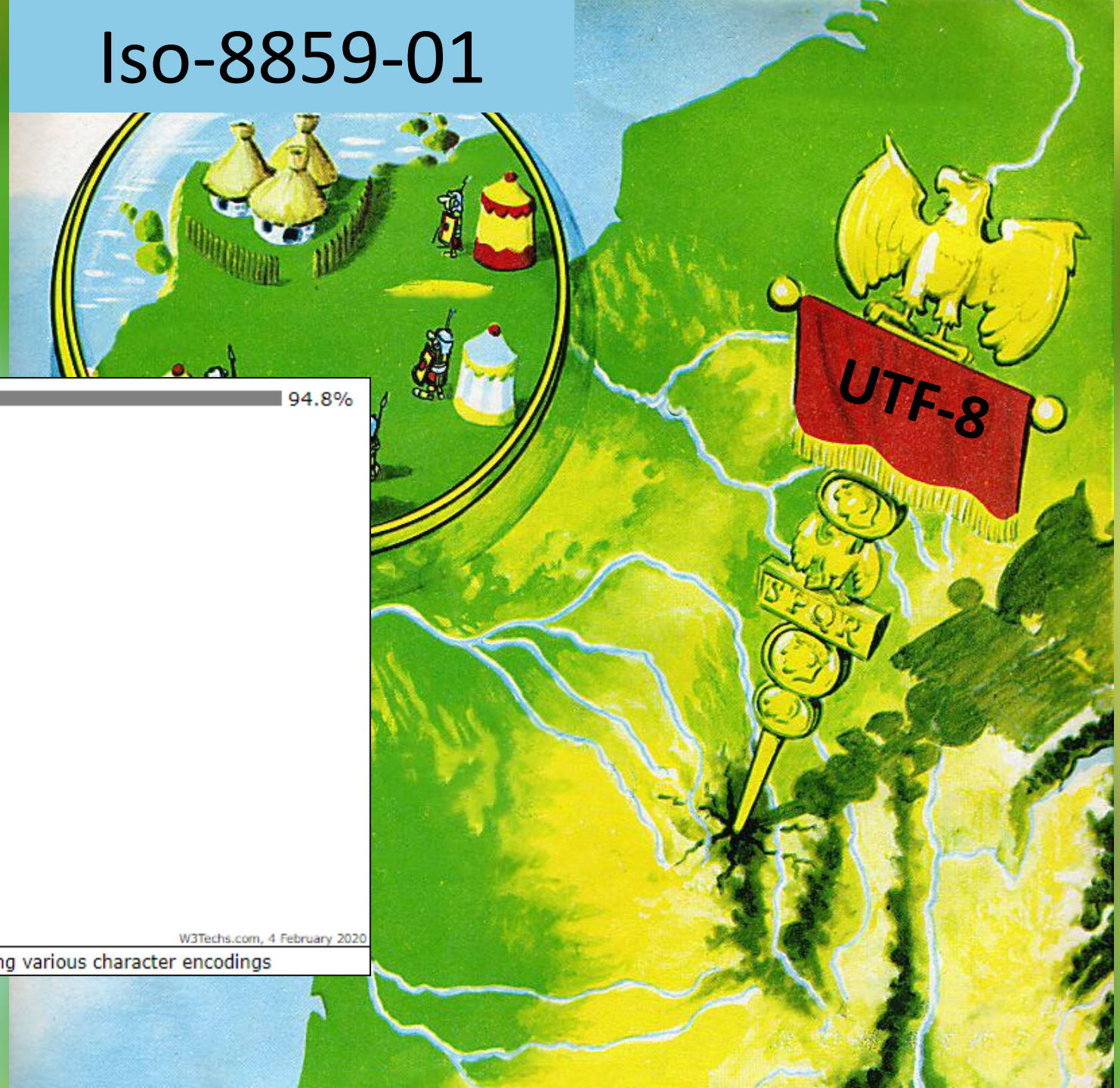
- Encoding format supporting Unicode Code Points
- The most popular encoding format
- First presented in 1993, Officialized in 2003 (RFC 3629)
- Variable width length encoding … 1 byte != 1 character

0___ ____					
110_ ____	10__ ____				
1110 ____	10__ ____	10__ ____	10__ ____		
[…]					
1111 110_	10__ ____	10__ ____	10__ ____	10__ ____	10__ ____

https://en.wikipedia.org/wiki/UTF-8#History

GoSecure

Iso-8859-01

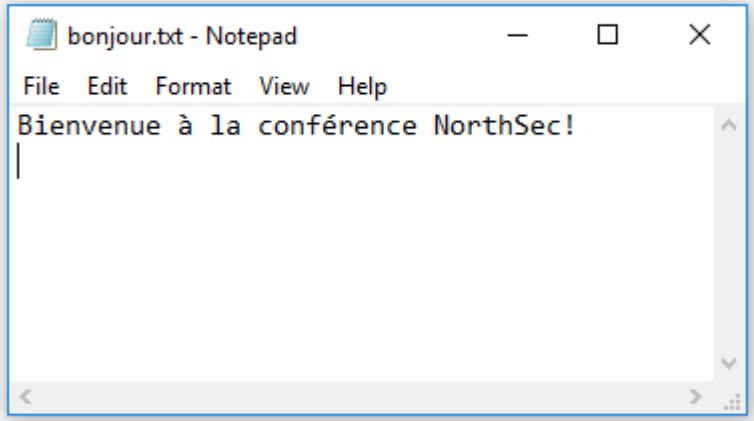


UTF-8	94.8%
ISO-8859-1	2.4%
Windows-1251	1.0%
Windows-1252	0.5%
Shift JIS	0.2%
GB2312	0.2%
EUC-KR	0.2%
EUC-JP	0.1%
ISO-8859-9	0.1%
GBK	0.1%
ISO-8859-2	0.1%
Windows-1250	0.1%
Big5	0.1%
Windows-1254	0.1%

W3Techs.com, 4 February 2020

Percentages of websites using various character encodings

Unexpected Encoding



ISO-8859-1

… \xe0 la conf\xe9rence

```
philippe@GS00RD01:~/encoding$ cat bonjour.txt
Bienvenue à la conférence NorthSec!
philippe@GS00RD01:~/encoding$
```

UTF-8

GoSecure

Normalization issues

NFC/NFKC Normalisation

Ç	⟷	C+◌̧
q+◌̇+◌̣	⟷	q+◌̣+◌̇
가	⟷	ᄀ + ᅡ
Ω	⟷	Ω

Objective :

- Compare two Unicode strings
- Use to build username, name, ids, description in URL or path.
- *Converting any Unicode string to [Basic Latin](#) (ASCII) only string.*

https://unicode.org/reports/tr15/#Norm_Forms

GoSecure

NFC vs NFKC

Canonical Equivalence (NFC)

Figure 1. Examples of Canonical Equivalence

Subtype	Examples
Combining sequence	Ç ⟷ C+◌̧
Ordering of combining marks	q+◌̇+◌̣ ⟷ q+◌̣+◌̇
Hangul & conjoining jamo	가 ⟷ ᄀ + ᅡ
Singleton equivalence	Ω ⟷ Ω

Compatibility Equivalence (NFKC)

Figure 2. Examples of Compatibility Equivalence

Subtype	Examples
Font variants	ℌ → H
	ℍ → H
Linebreaking differences	[NBSP] → [SPACE]
Superscripts/subscripts	i⁹ → i9
	i₉ → i9
Squared characters	㌀ → アパート
Fractions	¼ → 1/4
Other	dž → dž

GoSecure

Normalisation

When building URL or path, normalization can lead to security issues.

℅ (U+8453)	㌅ (U+8454)
﹫ (U+FE6B)	／ (U+FF0F)
？ (U+FE56)	‥ (U+2025)
ａ (U+FF41)	ⓒ (U+9426)

GoSecure

Normalization in action (1/5)

https://www.evil.c^{a}⁄c.microsoft.com

https://www.evil.ca/c.microsoft.com

https://gosecure.github.io/unicode-pentester-cheatsheet/

GoSecure

Normalization in action (2/5)

https://hola@www.microsoft.com

https://hola@www.microsoft.com

https://gosecure.github.io/unicode-pentester-cheatsheet/

GoSecure

Normalization in action (3/5)

[…]/documents/1﹖view=admin&?view=user

[…]/documents/1?view=admin&?view=user

https://gosecure.github.io/unicode-pentester-cheatsheet/

GoSecure

Normalization in action (4/5)

[…]/user/admin／/info

[…]/user/admin//info

https://gosecure.github.io/unicode-pentester-cheatsheet/

GoSecure

Normalization in action (5/5)

[...]/documents/download/：／：／admin／secret

[...]/documents/download/../../admin/secret

https://gosecure.github.io/unicode-pentester-cheatsheet/

GoSecure

```
253     /**
254      * Returns true if name matches against template.<p>
255      *
256      * The matching is performed as per RFC 2818 rules for TLS and
257      * RFC 2830 rules for LDAP.<p>
258      *
259      * The <code>name</code> parameter should represent a DNS name.
260      * The <code>template</code> parameter
261      * may contain the wildcard character *
262      */
263     private boolean isMatched(String name, String template,
264                               boolean chainsToPublicCA) {
265
266         // Normalize to Unicode, because PSL is in Unicode.
267         name = IDN.toUnicode(IDN.toASCII(name));
268         template = IDN.toUnicode(IDN.toASCII(template));
269
270         if (hasIllegalWildcard(name, template, chainsToPublicCA)) {
271             return false;
272         }
273
274         // check the validity of the domain name template.
275         try {
276             // Replacing wildcard character '*' with 'x' so as to check
277             // the domain name template validity.
278             //
279             // Using the checking implemented in SNIHostName
280             new SNIHostName(template.replace('*', 'x'));
281         } catch (IllegalArgumentException iae) {
282             // It would be nice to add debug log if not matching.
283             return false;
284         }
285
286         if (checkType == TYPE_TLS) {
287             return matchAllWildcards(name, template);
288         } else if (checkType == TYPE_LDAP) {
289             return matchLeftmostWildcard(name, template);
290         } else {
291             return false;
292         }
293     }
```

Hostname Verification Issue In OracleJDK and OpenJDK

(CVE-2020-14577)

GoSecure

Normalization Recommendations

- If you need to do normalization, normalized prior to security validation
- Review your library (HTTP and Network)
- Prefer whitelist over blacklist
- Strict validation on user input stored or pass to other API

GoSecure

Case modification

Case modification

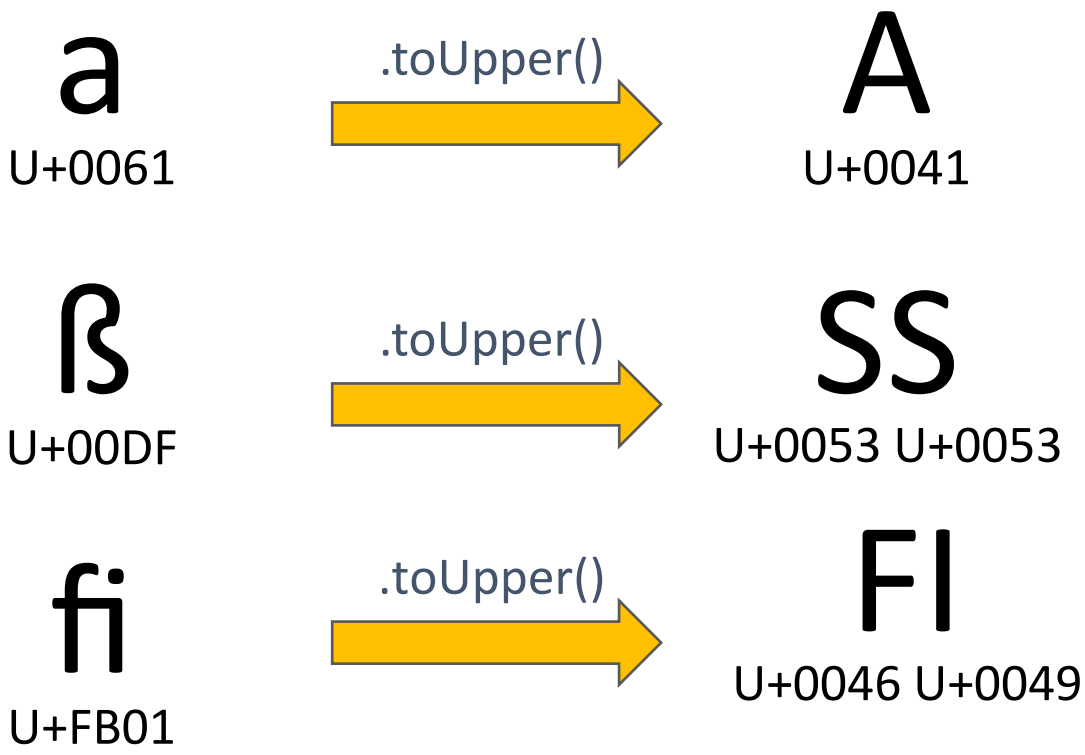
What happens when « to upper case » and « to lower case » are called?

- Unicode define the behavior when characters are converted to a lower case or uppercase variant.
- Not all character have a variant.
- Many characters share the same variant [1]

[1] Reference: Why there are no uppercase character for X? (collision)
http://unicode.org/faq/casemap_charprop.html#10

GoSecure

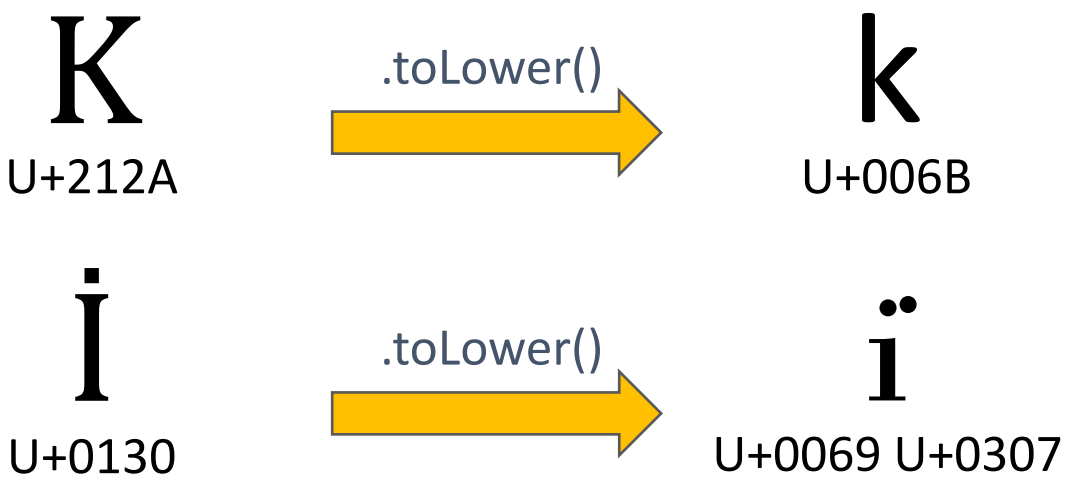
Upper case modification examples



https://gosecure.github.io/unicode-pentester-cheatsheet/

GoSecure

Lower case modification examples



https://gosecure.github.io/unicode-pentester-cheatsheet/

GoSecure

Potential issues

- Character collision
  - When doing comparison with Hostname, Username, Roles, etc.
- Bypass WAF, whitelist or blacklist

```
if current_user.upper() == "ADMIN":
"adm\u0131n".upper() == "ADMIN"
```

GoSecure

Certificate validation weakness

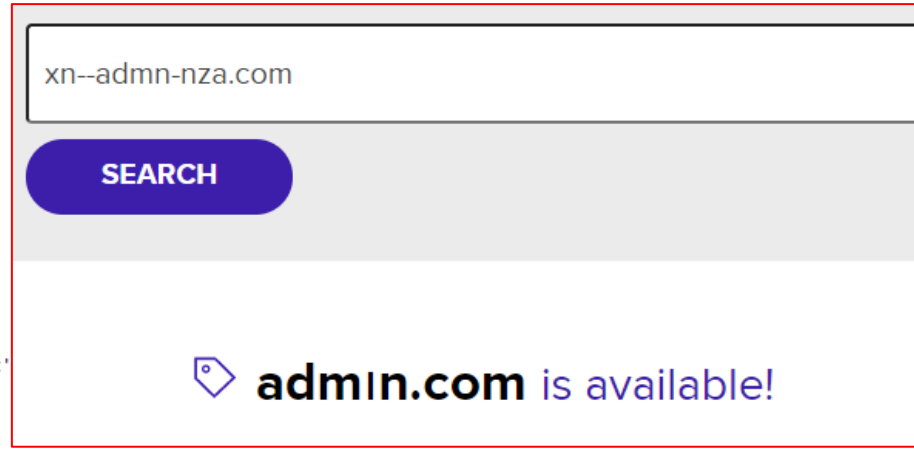
```
private static boolean matchAllWildcards(String name,
     String template) {
    name = name.toLowerCase();
    template = template.toLowerCase();
    StringTokenizer nameSt = new StringTokenizer(name, ".");
    StringTokenizer templateSt = new StringTokenizer(template, ".");

    if (nameSt.countTokens() != templateSt.countTokens()) {
        return false;
    }

    while (nameSt.hasMoreTokens()) {
        if (!matchWildCards(nameSt.nextToken(),
                    templateSt.nextToken())) {
            return false;
        }
    }
    return true;
}
```

GoSecure

Forgot password in Django (CVE-2019-19844)

Vulnerable implementation

```
278     def save(self, domain_override=None,
279              subject_template_name='registration/password_reset_subject.txt'
280              email_template_name='registration/password_reset_email.html',
281              use_https=False, token_generator=default_token_generator,
282              from_email=None, request=None, html_email_template_name=None,
283              extra_email_context=None):
284         """
285         Generate a one-use only link for resetting password and send it to the
286         user.
287         """
288         email = self.cleaned_data["email"]
289         for user in self.get_users(email):
290             if not domain_override:
291                 current_site = get_current_site(request)
292                 site_name = current_site.name
293                 domain = current_site.domain
294             else:
295                 site_name = domain = domain_override
296             context = {
297                 'email': email,
298                 'domain': domain,
299                 'site_name': site_name,
300                 'uid': urlsafe_base64_encode(force_bytes(user.pk)),
301                 'user': user,
302                 'token': token_generator.make_token(user),
303                 'protocol': 'https' if use_https else 'http',
304                 **(extra_email_context or {}),
305             }
306             self.send_mail(
307                 subject_template_name, email_template_name, context, from_email,
308                 email, html_email_template_name=html_email_template_name,
309             )
310
```



super@admin.com

super@adm\u0130n.com

super@xn--admn-nza.com

GoSecure

Forgot password in Django (CVE-2019-19844)

Correct implementation

```
292     def save(self, domain_override=None,
293              subject_template_name='registration/password_reset_subject.txt',
294              email_template_name='registration/password_reset_email.html',
295              use_https=False, token_generator=default_token_generator,
296              from_email=None, request=None, html_email_template_name=None,
297              extra_email_context=None):
298         """
299         Generate a one-use only link for resetting password and send it to the
300         user.
301         """
302         email = self.cleaned_data["email"]
303         email_field_name = UserModel.get_email_field_name()
304         for user in self.get_users(email):
305             if not domain_override:
306                 current_site = get_current_site(request)
307                 site_name = current_site.name
308                 domain = current_site.domain
309             else:
310                 site_name = domain = domain_override
311             user_email = getattr(user, email_field_name)
312             context = {
313                 'email': user_email,
314                 'domain': domain,
315                 'site_name': site_name,
316                 'uid': urlsafe_base64_encode(force_bytes(user.pk)),
317                 'user': user,
318                 'token': token_generator.make_token(user),
319                 'protocol': 'https' if use_https else 'http',
320                 **(extra_email_context or {}),
321             }
322             self.send_mail(
323                 subject_template_name, email_template_name, context, from_email,
324                 user_email, html_email_template_name=html_email_template_name,
325             )
```

GoSecure

Demonstration

GoSecure

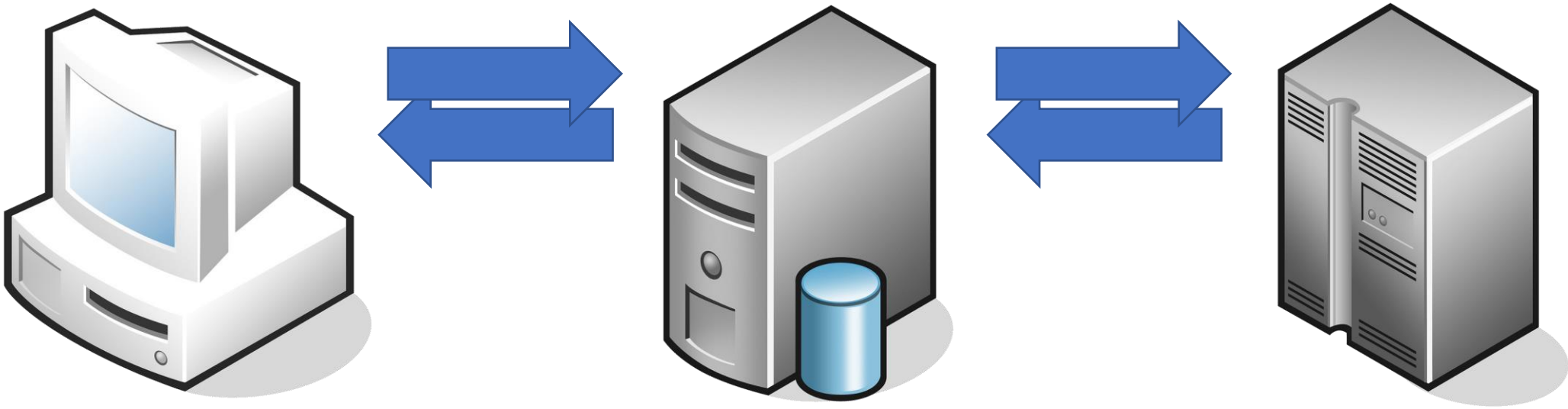
Mitigations for case modification

- Use ASCII only lower/upper functions
- Use invariant variation if available in your language.
  - In CSharp: “user input”.ToLowerInvariant()

GoSecure

When encoding allows security bypasses

Reason why security filtering can fail



Web Application Firewall

Web Application

GoSecure

Alternative to UTF-8

Unicode is not the only way to encode characters

Here are standard encoding commonly supported

- UTF-7
- UTF-16LE
- UTF-16BE
- UTF-32…

GoSecure

Malicious XML document in UTF-16

In UTF-16, every character is stored on two bytes

<book: 3c 62 6f 6f 6b => **00**3c **00**62 **00**6f **00**6f **00**6b

Byte Order Mark:

- First two bytes are used to define the endianness

Encoding	Byte Order Mark (bytes in hex)
UTF-8	EF BB BF
UTF-16BE	FE FF
UTF-16LE	FF FE

These bytes are more likely to be honored when a file is read than an HTTP request.

GoSecure

Malicious XML document in UTF-16

How the bytes are stored

```
> cat book_utf16be.xml
 < b o o k   x m l : i d = " s i m p l e _ b o o k "   x m l n s = " h t t p : / / d o c b o o k . o r g / n s / d o c b o o k
"   v e r s i o n = " 5 . 0 " >
   < t i t l e > V e r y   s i m p l e   b o o k < / t i t l e >
   < c h a p t e r   x m l : i d = " c h a p t e r _ 1 " >
      < t i t l e > C h a p t e r   1 < / t i t l e >
      < p a r a > H e l l o   w o r l d ! < / p a r a >
      < p a r a >
        I   h o p e   t h a t   y o u r   d a y   i s   p r o c e e d i n g   < e m p h a s i s > s p l e n d i d l y < / e
m p h a s i s > !
      < / p a r a >
   < / c h a p t e r >
   < c h a p t e r   x m l : i d = " c h a p t e r _ 2 " >
      < t i t l e > C h a p t e r   2 < / t i t l e >
      < p a r a > H e l l o   a g a i n ,   w o r l d ! < / p a r a >
   < / c h a p t e r >
< / b o o k >
```

How your application will see it

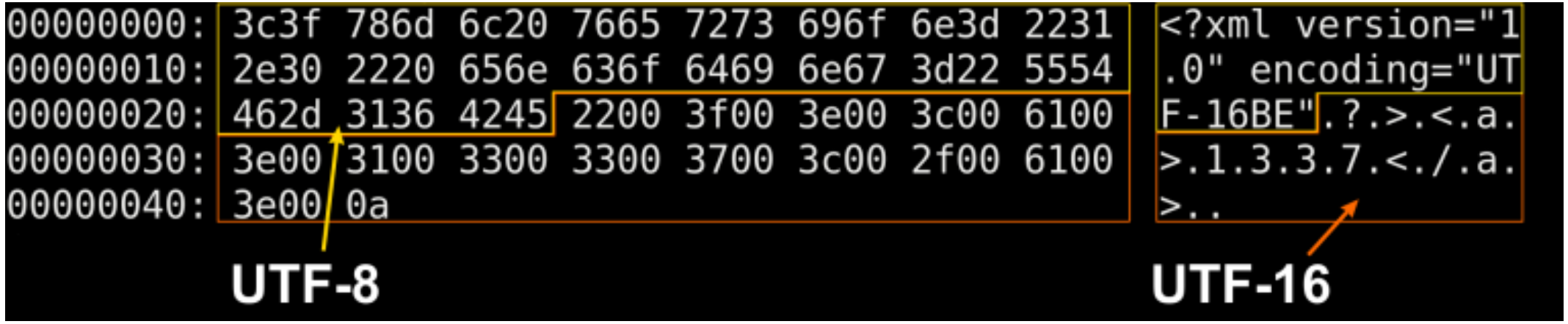
book_utf16be.xml

```
1  <book xml:id="simple_book" xmlns="http://docbook.org/ns/docbook" version="5.0">
2    <title>Very simple book</title>
3    <chapter xml:id="chapter_1">
4      <title>Chapter 1</title>
5      <para>Hello world!</para>
6      <para>
7        I hope that your day is proceeding <emphasis>splendidly</emphasis>!
8      </para>
9    </chapter>
10   <chapter xml:id="chapter_2">
11     <title>Chapter 2</title>
12     <para>Hello again, world!</para>
13   </chapter>
14 </book>
```

GoSecure

Malicious XML document with mixed encoding

- In protocols where encoding is defined, Byte Order Mark is not enough.
- XML support change encoding in the XML declaration section.



https://mohemiv.com/all/evil-xml/

GoSecure

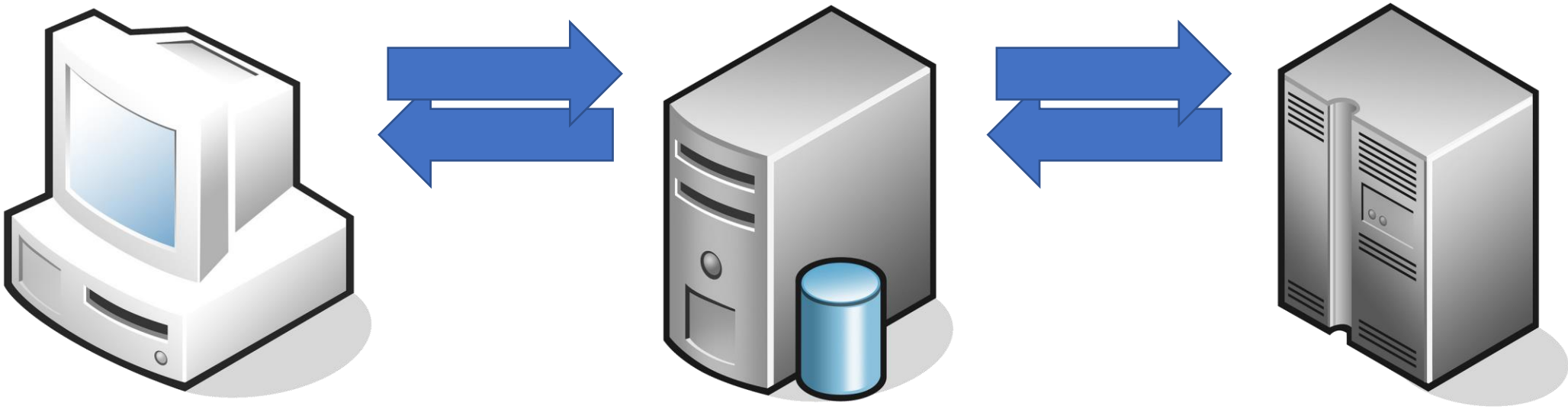
Other avenues

There are plenty obfuscation techniques that are do not require fancy Unicode encoding (out-of-scope for this presentation)

- XML entities (inside XML) /etc/passwd
- Double encoding %253C -> %3C -> <
- Non-printable characters java**\t**script:
- Partial encoding /etc/passwd
- Parser specific behavior

GoSecure

Storing user input inside a database



GoSecure

Persistent XSS inside database

Query

INSERT INTO ContentTable
VALUES (...,'＜img src=...')

＜img src=...

U+FF1C

Data once stored

<img src=...

U+003C

Example tested on SQL Server with default collation **SQL_Latin1_General_CP1_CI_AS**

GoSecure

Homograph attacks

With punycode domains

Punycode domains

Punycode is a representation of Unicode characters using ASCII characters.

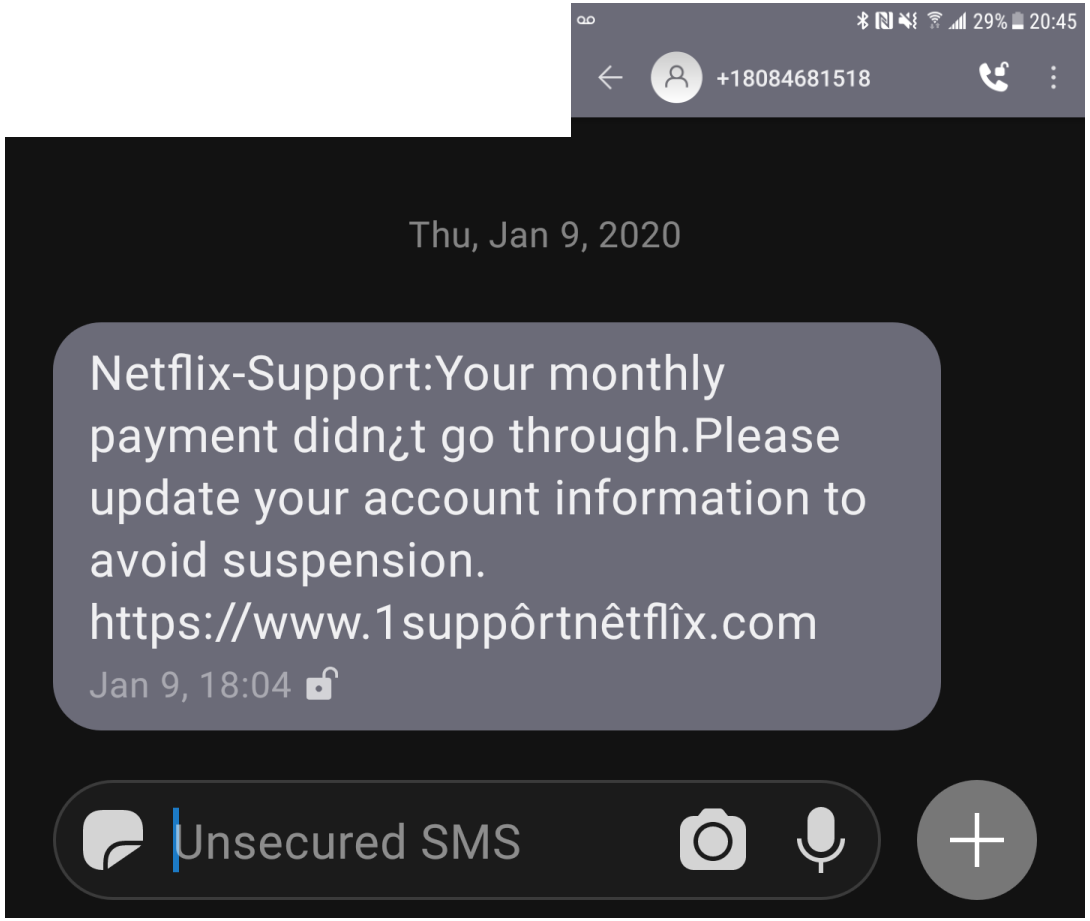
It use the bootstring encoding system.

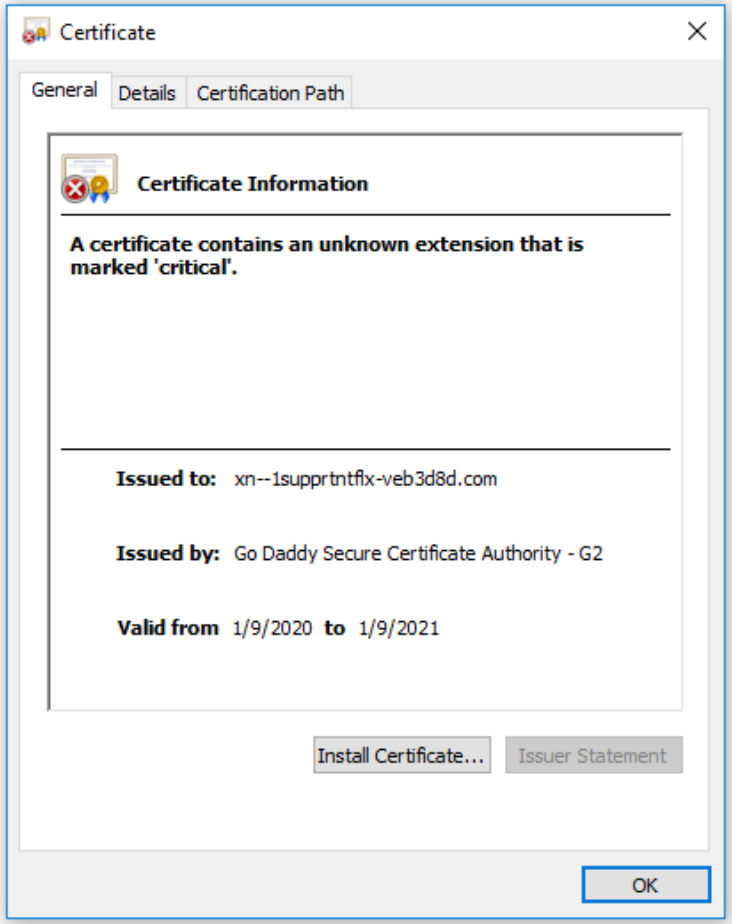
Example:

- nörthsec.com
- xn--nrthsec-90a.com

GoSecure

Homograph SMS Phishing

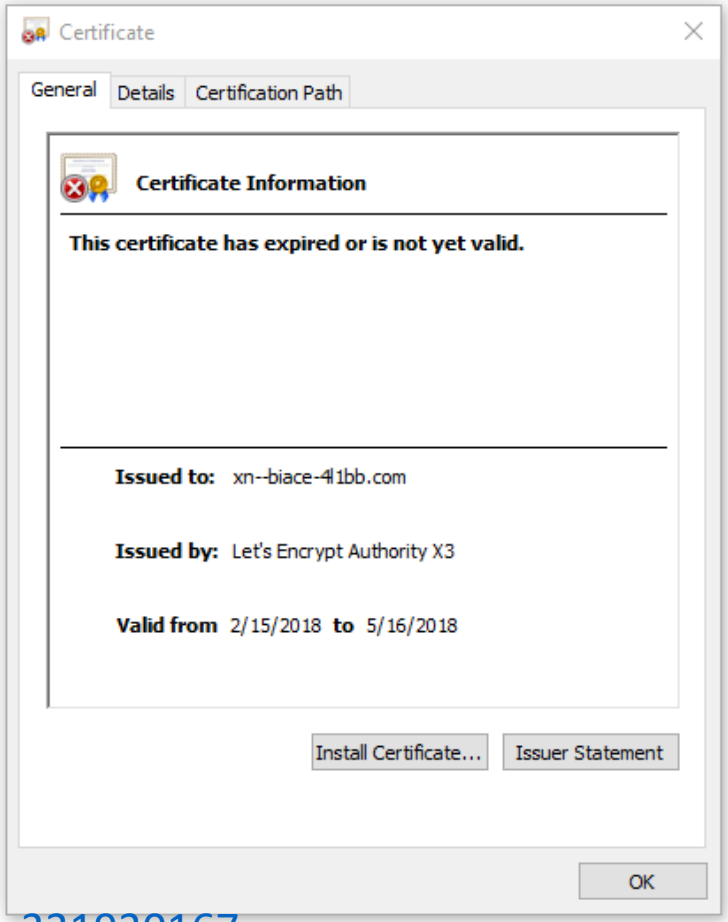




Source : https://crt.sh/?id=2310598224

GoSecure

Homograph phishing attack



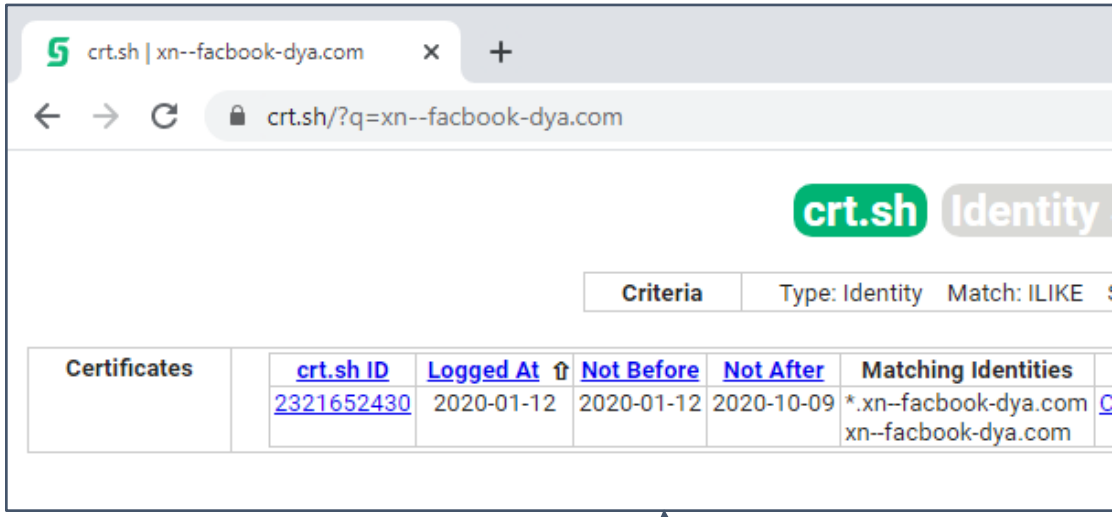
Source : https://bitcointalk.org/index.php?topic=5184169.0 & https://crt.sh/?id=331930167

GoSecure

Mitigations

Defenses

- Browser defense (mostly Chrome)
- Sender Policy Framework (SPF)
- Awareness training
- Find phishing domains on the TLS transparency log



Additional Reference:
https://us.norton.com/internetsecurity-how-to-cyber-security-best-practices-for-employees.html

GoSecure

Data integrity

Losing data because of encoding..

Data integrity

- Improper conversion can lead to partial data loss

- If the decoder does not recognize the encoding…
  - it may raise an exception
  - or it may replace the characters silently.

'conf**\xe9**rence OWASP' ➡ 'conf**\ufffd**rence OWASP'

GoSecure

Do NOT use

- Java : new String(binary_file_content, Encoding.UTF8)
- Python : binary_file_content.decode("utf-8", errors="ignore")

You SHOULD use

- Java:
  - CharsetDecoder decoder = Charsets.UTF_8.newDecoder().onMalformedInput(**CodingErrorAction.REPORT**);
    CharBuffer decoded = decoder.decode(ByteBuffer.wrap(input));
- Python
  - binary_file_content.decode("utf-8“)

GoSecure

Conclusion

Recommendation

- Be cautious when using
  - Case changing function (.upper(), .lower())
  - Normalization function (UTF-8 to ASCII)
  - Internalization function (Punycode to UTF-8)
- Restrict to ASCII (CP < 128) (username, service name, id)
- Declare explicitly the encoding to avoid implicit default that vary

Questions

Contact information

- parteau@gosecure.ca
- @GoSecure_Inc
- @h3xStream

Slides

https://gosecure.github.io/presentations/

References

References for normalization

- Black Hat presentation of HostSplit: https://i.blackhat.com/USA-19/Thursday/us-19-Birch-HostSplit-Exploitable-Antipatterns-In-Unicode-Normalization.pdf

- GitHub vulnerabilities with lowercase issue: https://eng.getwisdom.io/hacking-github-with-unicode-dotless-i/

GoSecure

References for Punycode

- Good presentation on Punycode visual attack: http://www.irongeek.com/i.php?page=security/out-of-character-use-of-punycode-and-homoglyph-attacks-to-obfuscate-urls-for-phishing
- Unicode characters utility demonstrated : https://gosecure.github.io/unicode-pentester-cheatsheet/

GoSecure