Università degli Studi di Cagliari

Ma le CTF servono davvero a qualcosa?

Lorenzo {Siriu, Pisu}

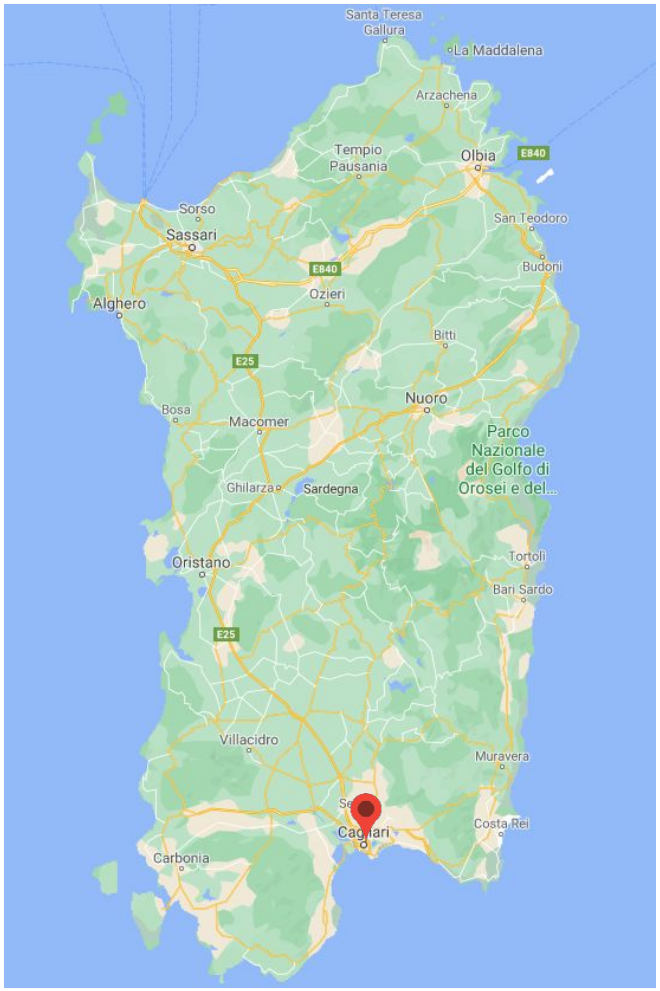
srdnlen@gmail.com

https://srdnlen.it

CyberChallenge.IT Workshop 2023 - 01/07/2023

Srdnlen

Srdnlen si pronuncia “Sardinia Len”



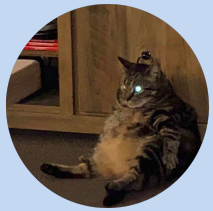
Timeline

Fondatore: Prof. Davide Maiorca
Ricercatore @ UniCa
Capitano e coach

2019








2020





2021



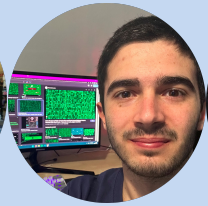




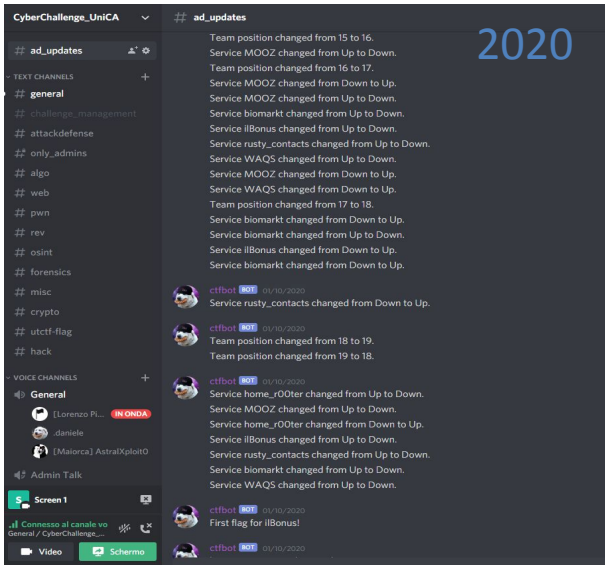
2022

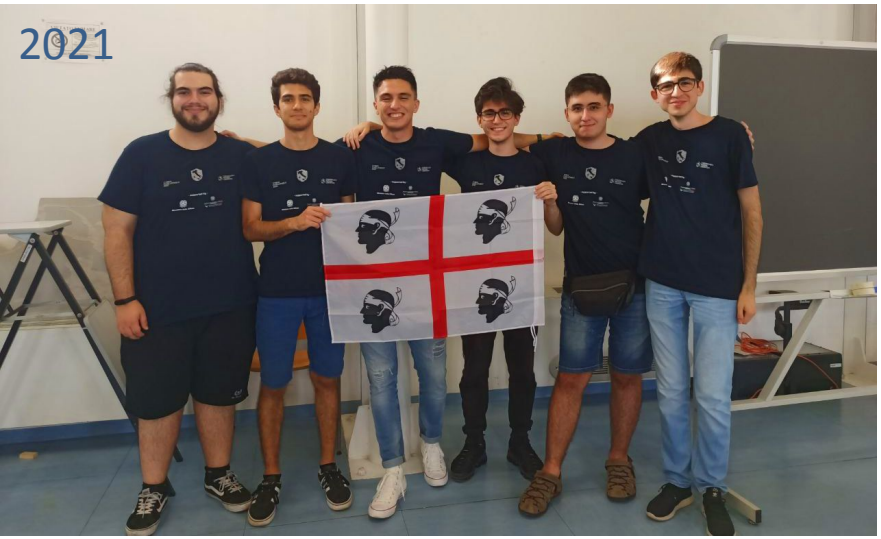





CyberChallenge Negli Anni





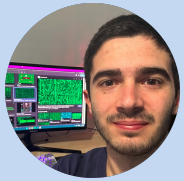
Skills

Web







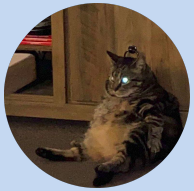

Misc/Foren/Network







Pwn/Rev

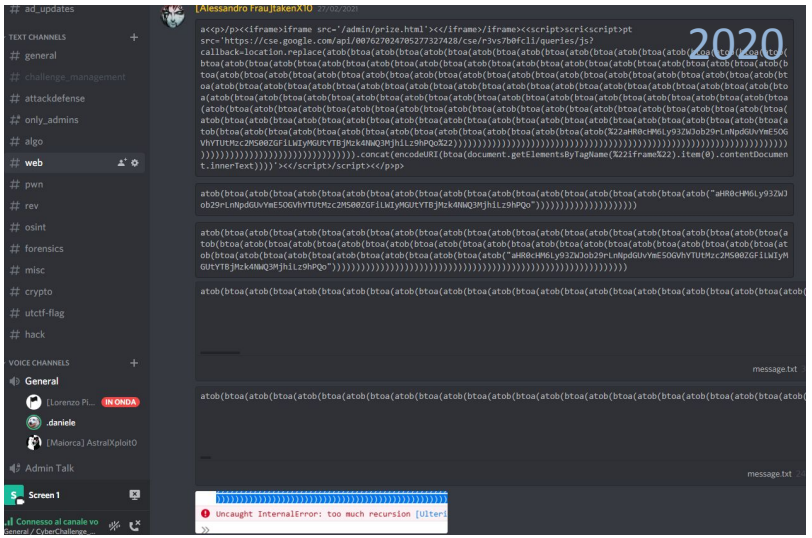





Crypto



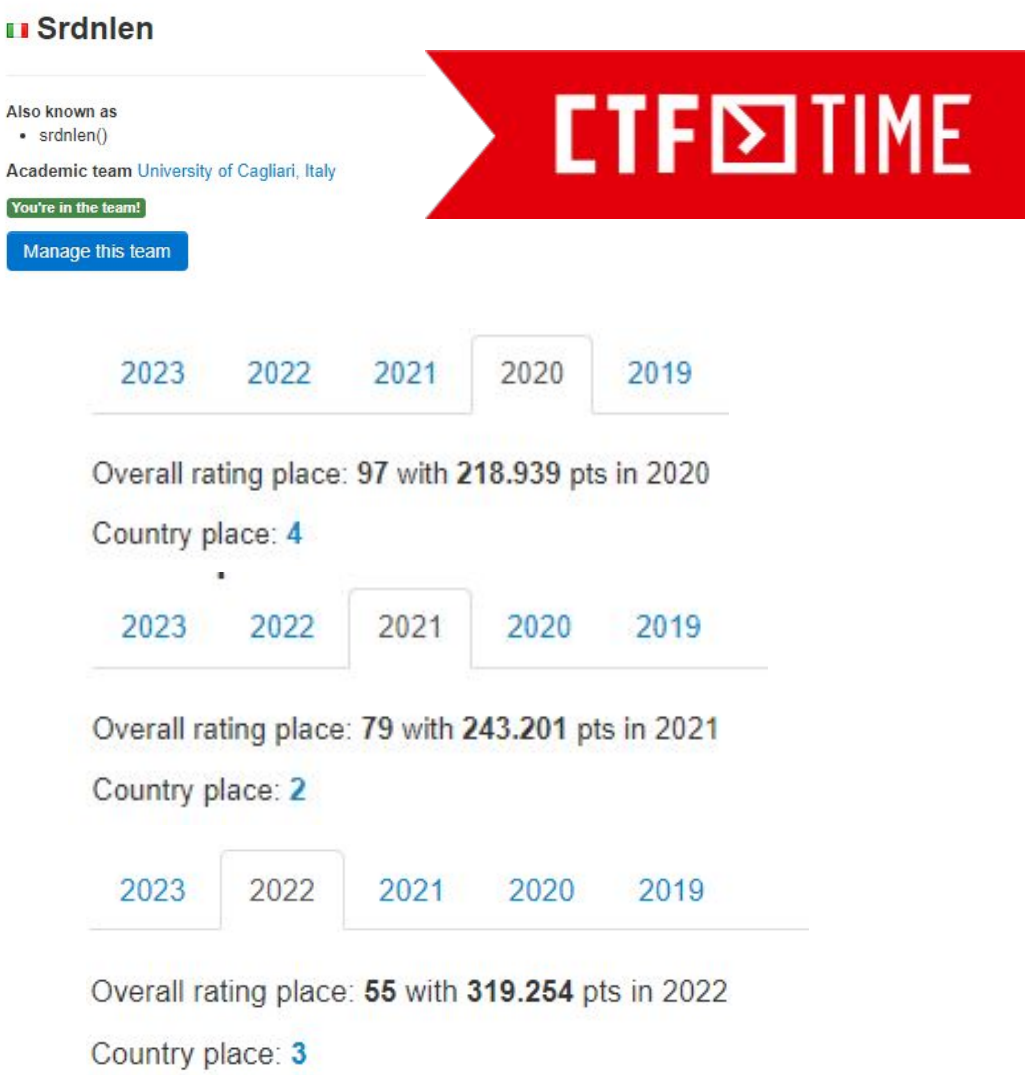
Srdnlen - Allenamenti (e non solo)



La Nostra Prima CTF

picoCTF

3 Anni Dopo


Srdnlen
Also known as
• srdnlen()
Academic team University of Cagliari, Italy
You're in the team!
Manage this team
CTF TIME
2023 2022 2021 2020 2019
Overall rating place: 97 with 218.939 pts in 2020
Country place: 4
2023 2022 2021 2020 2019
Overall rating place: 79 with 243.201 pts in 2021
Country place: 2
2023 2022 2021 2020 2019
Overall rating place: 55 with 319.254 pts in 2022
Country place: 3

Outline

Ora entriamo nella parte tecnica della presentazione

- **Crittografia**
  - Crittografia nelle CTF VS crittografia nel mondo reale
  - Presentazione della challenge “Custom Protocol” della Google CTF
- **Web security**
  - Web security nelle CTF VS web security nel mondo della ricerca
  - Server-Side Template Injection
  - Cross-Site Scripting

Chi Sono

Università

- 3° anno informatica UniCA (ora)
- CyberChallenge.IT 2022
  - Primo classificato finale locale
  - Membro squadra UniCA alla finale nazionale
- Tutor di crypto CyberChallenge.IT 2023

Srdnlen

- Aree di competenza
  - crypto (principale)
  - rev/pwn (come supporto)

Crypto - CTF vs Mondo Reale

CTF:

- "Wow, interessante questo problema: ci posso creare una challenge"
- "Inseriamo una vulnerabilità in un protocollo crittografico"

Mondo reale:

- "Dobbiamo essere sicuri che il protocollo non sia vulnerabile"
- "Devi seguire queste indicazioni altrimenti il protocollo è vulnerabile"

Sempre mondo reale:

- "Se nessuno conosce la mia encryption sono al sicuro"
- "Guarda, ho appena creato questo *custom protocol*!"

CUSTOM PROTOCOL

La challenge è della Google CTF 2022 ed è un protocollo RSA accessibile da remoto.

- **Se ci connettiamo possiamo:**
  1. Ottenere la **chiave pubblica** dell'RSA: `N, e`
  2. Ottenere la `flag` **cifrata**
  3. Inviare un **messaggio cifrato** per sapere se **contiene la** `flag`
  4. Ottenere una **firma RSA della** `flag`
  5. Inviare una **firma RSA** per sapere se è **valida per la** `flag`

Siamo piuttosto limitati in quello che possiamo fare.

Proviamo ad analizzare il protocollo ed il codice del server…

CUSTOM PROTOCOL

Ogni messaggio parsato dal protocollo ha questi campi:

- `version` (~ 200 bits)
- `hmac key` (128 bits)
- `oid` (~ 1000 bits)
- `nonce` (160 bits)
- `algorithm name` (~ 1000 bits)
- `digest message` (160 bits)
- `message` (solo per la cifratura)
- `padding` (< 1200 bits)

dove i campi noti sono in verde e quelli sconosciuti in rosso

I protocolli RSA non dovrebbero far trapelare così tante informazioni del *plaintext*

One Small Oversight To Exploit Them All

Abbiamo 3 campi sconosciuti quando firmiamo un messaggio:

- **hmac key**
  - generata randomicamente in maniera sicura
- **nonce**
  - dipende dalla hmac key e il **tempo** (**in secondi**)
- **digest message**
  - dipende dalla hmac key e dal messaggio

Oh, questo non va bene…

```
def sign(self, msg: bytes, hmac_key: bytes) -> int:
  sp = pow(self.parse(msg, hmac_key, DataType.RSA_SIG), self.dp, self.p)
  sq = pow(self.parse(msg, hmac_key, DataType.RSA_SIG), self.dq, self.q)
  h = self.qInv*(sp - sq) % self.p
  s = sq + h*self.q
  return s
```

One Small Oversight To Exploit Them All

Dato che la generazione della nonce dipende dal tempo, potremmo avere una firma non valida quando viene verificata: questo perché i due messaggi firmati modulo *p* and *q* avrebbero nonce differenti.

Quando generiamo una firma non valida *s*, sappiamo che le seguenti equazioni valgono:

$$s^e - m_1 \equiv 0 \pmod{p}, \quad s^e - m_1 \not\equiv 0 \pmod{q}$$
$$s^e - m_2 \equiv 0 \pmod{q}, \quad s^e - m_2 \not\equiv 0 \pmod{p}$$

Quindi, se potessimo recuperare quei piccoli campi sconosciuti, allora potremmo fattorizzare il modulo dell'RSA!

Ma come potremmo recuperarli?

Momento Meme

Meme found on https://mystiz.hk/posts/2022/2022-09-05-balsn/

Soluzione

Possiamo recuperare i campi sconosciuti con Coppersmith

- **Trova radici piccole di polinomi (β=0.5)**
  - Non tutte le implementazioni sono adatte al nostro problema
  - Potremmo avere bisogno di implementare una particolare versione

Soluzione di y011d4

- **Combinando le precedenti equazioni otteniamo questa:**

$$(s^e - m_1)(s^e - m_2) \equiv 0 \pmod{N}$$

  - Con **LLL** possiamo recuperare alcune informazioni che ci permettono di **semplificare il problema** e risolverlo più facilmente

Mi raccomando ricordate sempre:

`CTF{They_say_never_implement_your_own_crypto...ok_ok_lesson_learned}`

Il Mio Percorso

Prima di CyberChallenge (2018)

- 1° anno di Università: voglio programmare i videogiochi

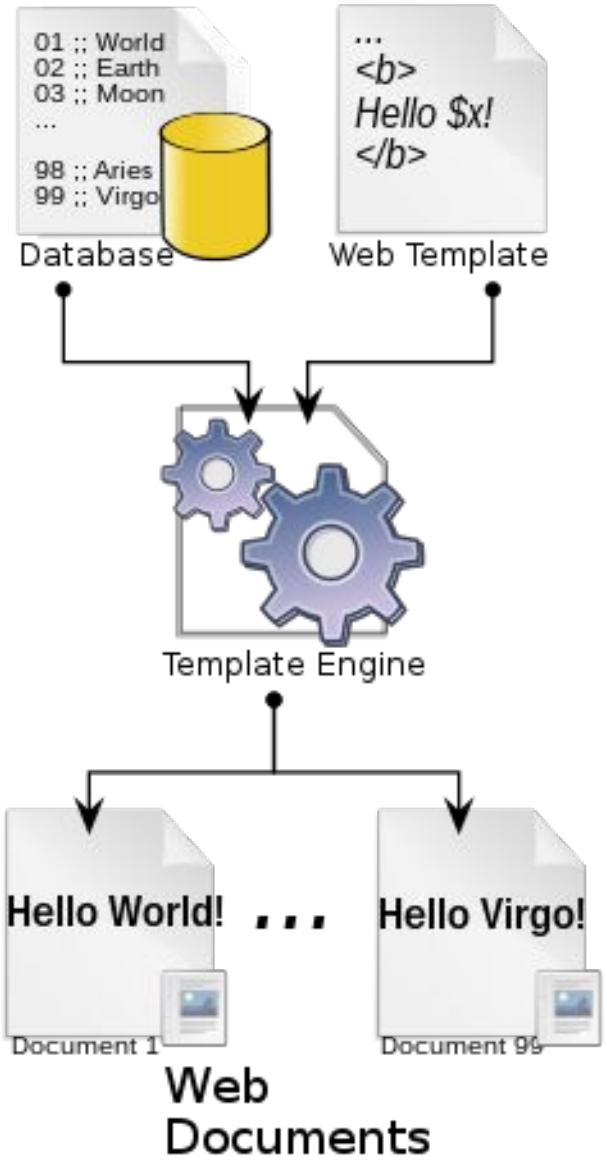
CyberChallenge.IT 2019

- 2° anno di Università

Dopo CyberChallenge

- Laurea magistrale: Computer Engineering, Cybersecurity and AI
- Tesi su “Server-Side Template Injection”
- Dottorato di ricerca - web security (ora)

Web - SSTI


01 :: World
02 :: Earth
03 :: Moon
...
98 :: Aries
99 :: Virgo
Database
...

Hello $x!

Web Template
Template Engine
Hello World!
...
Hello Virgo!
Document 1
Document 99
Web
Documents

Web - SSTI

CODICE SICURO – L'input dell'utente viene inserito nella pagina in maniera sicura

```
user_input = request.form['username']
template = "<html><h1>Welcome, {{ username }} !</h1></html>"
return render_template_string(template, username=user_input)
```

CODICE VULNERABILE – L'input dell'utente viene messo dentro il template

```
user_input = request.form['username']
template = "<html><h1>Welcome, %s !</h1></html>" % user_input
return render_template_string(template)
```

Web - SSTI

CODICE VULNERABILE – L'input dell'utente viene messo dentro il template

```
user_input = request.form['username']
template = "<html><h1>Welcome, %s !</h1></html>" % user_input
return render_template_string(template)
```

```
{{config.__class__.__init__.__globals__['os'].popen('ls').read()}}
```

A global object

Python introspective attributes

os module

arbitrary command execution

output collection

Web - SSTI

- **Lo stato dell'arte su SSTI è molto limitato**
  - Esistono tantissimi template engine
  - La maggior parte di essi consentono di ottenere RCE
  - Se non fosse stato per tutte le challenge di SSTI non avrei mai scoperto queste cose

Python	Jinja2
	Cheetah
	Django
	Genshi and Kid
	Mako
	web2py
	Tornado
	Chameleon
	Pyratemp
PHP	Twig
	Smarty
	Laravel (Blade)

Java	Jinjava
	Pebble
	Thymeleaf
	FreeMarker
	Apache Velocity
Ruby	ERB
	Slim
Golang	Default engine
Perl	Mojolicious
.NET	Razor
	ASP

JavaScript	Pug and Jade
	JsRender
	Handlebars
	Nunjucks
	Vue
	Dust
	doT
	EJS
	Marko
	SquirrellyJS
	Template7

Web - XSS

Un'altra vulnerabilità molto presente nelle CTF è Cross-Site Scripting

- **Questo codice è vulnerabile?**
- **Che payload si può usare per scrivere un exploit?**

```
1 $src = strtoupper(htmlspecialchars($_GET['src']));
2 $image = "<img src='".$src."' alt='Error' errorwidth=30% height=30%>";
3 echo $image;
```

Web - XSS

Le CTF spesso richiedono di sapere dettagli molto specifici su certe vulnerabilità

- **I browser interpretano HTML in maniera case insensitive**
- **JavaScript invece è case sensitive ma esiste JSfuck**

```
1 $src = strtoupper(htmlspecialchars($_GET['src']));
2 $image = "<img src='".$src."' alt='Error' errorwidth=30% height=30%>";
3 echo $image;
```

Payload: nonexistent' onerror='(![]+[])...'

Pala, Pisu, Sanna, Maiorca, Giacinto - A Targeted Assessment of Cross-Site Scripting Detection Tools - ITASEC 23

Conclusione

- **Le CTF non solo servono a qualcosa ma possono essere fondamentali per acquisire competenze spendibili nella propria carriera**
- **Grazie a CyberChallenge si possono acquisire le conoscenze adatte ad entrare nel mondo delle CTF e non solo**
  - Trovare altre persone con cui fare team
  - Acquisire competenze tecniche forti
  - Lavorare in ambito cybersecurity
  - Trovare argomenti su cui fare ricerca

Grazie!

Potete seguirci su Instagram e Twitter
@srdnlen

lorenzo.pisu@unica.it
lrnzsir@gmail.com