



ESPECIALIDAD EN CALIDAD DE PROYECTOS DE SOFTWARE

PROYECTO
CIBERSEGURIDAD

Equipo 3

Sara Carolina Gómez Delgado
Luis Eduardo Robles Jimenez
María Cristina Velázquez García
María Fernanda Zavala López

MAYO 2024



Temas elegidos

01

Herramienta para decodificadores y reglas con base en logs (txt)

- Creación de decodificadores y reglas en archivos XML en Wazuh
- Creación de herramienta en Python para creación de decodificadores y reglas

02

API de Wazuh

Integración de herramientas de la API en un notebook de Python

¿Por qué elegimos estos temas?

Dado que el tema 1 era parte obligatoria del proyecto, decidimos que el tema que mejor se podía integrar con el resto del mismo era utilizar la API de Wazuh, primero para poder modificar los decodificadores y reglas creados y posteriormente para integrar más aplicaciones del mismo servicio, haciendo una mejor cohesión del proyecto



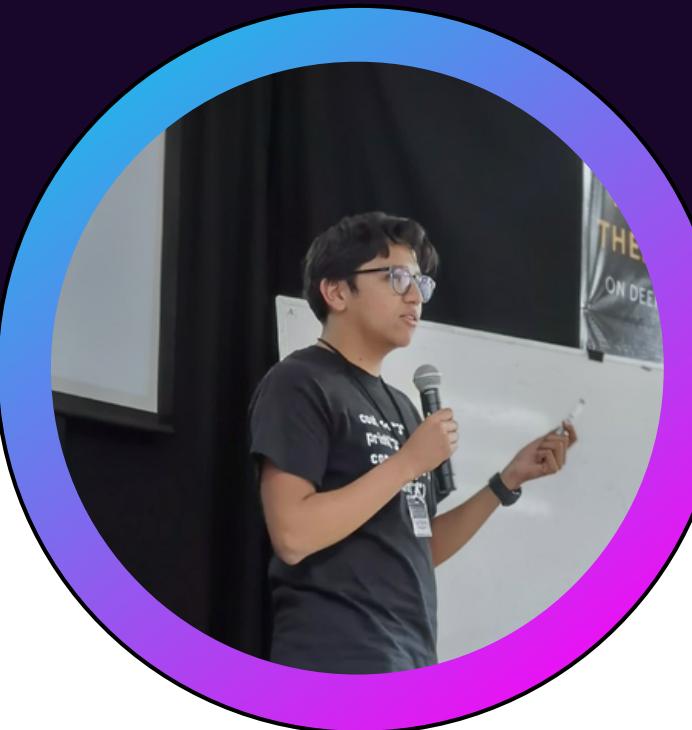
Distribución de tareas y responsabilidades

A pesar de que el equipo estuvo al pendiente de las dificultades de los otros miembros, principalmente las tareas se dividieron de manera general de la siguiente manera:



Fernanda Zavala

- Primer approach con python para la creación de decodificadores y reglas, conexión la API a través del CURL
- Creación de archivos de logs, decodificadores y reglas en XML



Luis Eduardo Robles

- Desarrollo de Notebook en Python para la creación de decodificadores y reglas



Cristina Velázquez

- Desarrollo de Notebook en Python para la conexión con la API e implementación de herramientas con la misma



Sara Gómez

- Desarrollo de la presentación
- Manejo y administración de archivos XML con python y librería especializada



TIPO DE LOGS SELECCIONADOS

Simulamos los Logs que se generan de una tienda en linea de artículos tecnológicos, en donde se incluyen actividades como:

- Login (exitoso o fallido)
- Agregar productos al carrito
- Compra de producto
- Cancelar una orden
- Actualizar el perfil del usuario



Ejemplo de Logs:

```
May 08 15:23:45 macbookdemaria techstore_inventory: type=Information username=maria event=login status=succeeded
```

```
May 08 16:45:12 laptopdejuan techstore_inventory: type=Information username=juan event=checkout total=50.00
```

```
May 08 17:30:05 pcdepedro techstore_inventory: type=Information username=pedro event=add_to_cart product=shirt price=25.00
```

```
May 08 18:15:30 tabletdeana techstore_inventory: type=Information username=ana event=cancel_order order_id=12345
```

```
May 08 19:00:18 phoneofsara techstore_inventory: type=Information username=sara event=update_profile
```

```
May 08 20:45:30 pcdepedro techstore_inventory: type=error username=pedro event=login status=failed
```



Archivo “decodificadores”

The screenshot shows the Wazuh Management interface with the 'Decoders' tab selected. The title bar includes icons for navigation and a dropdown menu, followed by 'wazuh.' and the current section. The main content area displays the XML configuration for decoders named 'techstore_inventory' and 'log_type'. The code uses color-coded syntax highlighting for tags and attributes.

```
1 <decoder name="techstore_inventory">
2   <program_name>techstore_inventory</program_name>
3   </decoder>
4
5 <decoder name="log_type">
6   <parent>techstore_inventory</parent>
7   <regex>type=(\S+) username=(\S+) event=(\S+)</regex>
8   <order>type, username, event</order>
9   </decoder>
10 <decoder name="log_type">
11   <parent>techstore_inventory</parent>
12   <regex>type=(\S+) username=(\S+) event=(\S+) status=(\S+)</regex>
13   <order>type, username, event, status</order>
14   </decoder>
15 <decoder name="log_type">
16   <parent>techstore_inventory</parent>
17   <regex>type=(\S+) username=(\S+) event=(\S+) order_id=(\S+)</regex>
18   <order>type, username, event, order_id</order>
19   </decoder>
20 <decoder name="log_type">
21   <parent>techstore_inventory</parent>
22   <regex>type=(\S+) username=(\S+) event=(\S+) total=(\S+)</regex>
23   <order>type, username, event, total</order>
24   </decoder>
25 <decoder name="log_type">
26   <parent>techstore_inventory</parent>
27   <regex>type=(\S+) username=(\S+) event=(\S+) product=(\S+) price=(\S+)</regex>
28   <order>type, username, event, product, price</order>
29   </decoder>
30 <decoder name="log_type">
31   <parent>techstore_inventory</parent>
32   <regex>type=(\S+) username=(\S+) event=(\S+) product=(\S+) price=(\S+)</regex>
33   <order>type, username, event, product, price</order>
34   </decoder>
35
36
```



Archivo “reglas”

≡ |  wazuh. ▾ Management Rules

◀ fzavalal.xml

```
1 <group name="techstore_inventory">
2   <rule id="131000" level="3">
3     <decoded_as>techstore_inventory</decoded_as>
4     <field name="event">cancel_order</field>
5     <description>The event indicates that an order with ID ${order_id} was cancelled</description>
6   </rule>
7   <rule id="131001" level="7">
8     <decoded_as>techstore_inventory</decoded_as>
9     <field name="type">error</field>
10    <description>Seems like something went wrong</description>
11  </rule>
12  <rule id="131002" level="3">
13    <decoded_as>techstore_inventory</decoded_as>
14    <field name="event">checkout</field>
15    <description>User ${username} has made a purchase for a total of ${total}</description>
16  </rule>
17  <rule id="131003" level="3">
18    <decoded_as>techstore_inventory</decoded_as>
19    <field name="event">add_to_cart</field>
20    <description>User ${username} has added ${product} to the cart</description>
21  </rule>
22  <rule id="131004" level="3">
23    <decoded_as>techstore_inventory</decoded_as>
24    <field name="event">checkout</field>
25    <description>User ${username} has made a purchase for a total of ${total}</description>
26  </rule>
27  <rule id="131005" level="5">
28    <decoded_as>techstore_inventory</decoded_as>
29    <field name="event">login</field>
30    <description>The user ${username} has logged in</description>
31  </rule>
32</group>
33
34
35
```



Resultado en “Ruleset test”

The screenshot shows the Wazuh API Console interface with the 'Ruleset Test' tab selected. The main area displays the test results for a single event:

```
May 08 20:45:30 pcdepedro techstore_inventory: type=error username=pedro event=login status=failed
```

Below the results, the test summary is shown:

```
**Phase 1: Completed pre-decoding.  
full event: 'May 08 20:45:30 pcdepedro techstore_inventory: type=error username=pedro event=login status=failed'  
timestamp: 'May 08 20:45:30'  
hostname: 'pcdepedro'  
program_name: 'techstore_inventory'  
  
**Phase 2: Completed decoding.  
name: 'techstore_inventory'  
event: 'login'  
status: 'failed'  
type: 'error'  
username: 'pedro'  
  
**Phase 3: Completed filtering (rules).  
id: '131001'  
level: '7'  
description: 'Seems like something went wrong'  
groups: '[ "techstore_inventory" ]'  
firedtimes: '2'  
mail: 'false'  
**Alert to be generated.
```

At the bottom right of the main area, there are two buttons: 'Test' (blue) and 'Clear session' (red).



ESPECIALIDAD EN CALIDAD DE PROYECTOS DE SOFTWARE

Herramienta para la creación de decodificadores y reglas



Ciberseguridad > Code > Wazuh > logs.log

```
1 May 08 15:23:45 macbookdemaria techstore_inventory: type=information username=maria event=login
2 May 08 17:30:05 pcdepedro techstore_inventory: type=information username=pedro event=add_to_cart product=shirt price=25.00
3 May 10 12:42:23 pclalito techstore_inventory: type=information username=lalito event=login
4 May 12 12:52:12 pcsarita techstore_inventory: type=information
```

wazuh. Management Rules

< fzavalal.xml

```
1 <group name="techstore_inventory">
2   <rule id="131000" level="3">
3     <decoded_as>techstore_inventory</decoded_as>
4     <field name="event">cancel_order</field>
5     <description>The event indicates that an order with ID ${order_id} was cancelled</description>
6   </rule>
7   <rule id="131001" level="7">
8     <decoded_as>techstore_inventory</decoded_as>
9     <field name="type">error</field>
10    <description>Seems like something went wrong</description>
11  </rule>
12  <rule id="131002" level="3">
13    <decoded_as>techstore_inventory</decoded_as>
14    <field name="event">checkout</field>
15    <description>User ${username} has made a purchase for a total of ${total}</description>
16  </rule>
17  <rule id="131003" level="3">
18    <decoded_as>techstore_inventory</decoded_as>
19    <field name="event">add_to_cart</field>
20    <description>User ${username} has added ${product} to the cart</description>
21  </rule>
22  <rule id="131004" level="3">
23    <decoded_as>techstore_inventory</decoded_as>
24    <field name="event">checkout</field>
25    <description>User ${username} has made a purchase for a total of ${total}</description>
26  </rule>
27  <rule id="131005" level="5">
28    <decoded_as>techstore_inventory</decoded_as>
29    <field name="event">login</field>
30    <description>The user ${username} has logged in</description>
31  </rule>
32 </group>
```

wazuh. Management Decoders

< fzavalal.xml

```
1 <decoder name="techstore_inventory">
2   <program_name>techstore_inventory</program_name>
3   </decoder>
4 
5 <decoder name="log_type">
6   <parent>techstore_inventory</parent>
7   <regex>type=(\S+) username=(\S+) event=(\S+)</regex>
8   <order>type, username, event</order>
9   </decoder>
10 <decoder name="log_type">
11   <parent>techstore_inventory</parent>
12   <regex>type=(\S+) username=(\S+) event=(\S+) status=(\S+)</regex>
13   <order>type, username, event, status</order>
14   </decoder>
15 <decoder name="log_type">
16   <parent>techstore_inventory</parent>
17   <regex>type=(\S+) username=(\S+) event=(\S+) order_id=(\S+)</regex>
18   <order>type, username, event, order_id</order>
19   </decoder>
20 <decoder name="log_type">
21   <parent>techstore_inventory</parent>
22   <regex>type=(\S+) username=(\S+) event=(\S+) total=(\S+)</regex>
23   <order>type, username, event, total</order>
24   </decoder>
25 <decoder name="log_type">
26   <parent>techstore_inventory</parent>
27   <regex>type=(\S+) username=(\S+) event=(\S+) product=(\S+) price=(\S+)</regex>
28   <order>type, username, event, product, price</order>
29   </decoder>
30 <decoder name="log_type">
31   <parent>techstore_inventory</parent>
32   <regex>type=(\S+) username=(\S+) event=(\S+) product=(\S+) price=(\S+)</regex>
33   <order>type, username, event, product, price</order>
34   </decoder>
35 
```



Nuestra herramienta para generar los archivos se divide en 2 partes.

1. Transformación de y hacia archivos XML
2. Análisis de decodificadores y reglas existentes

```

def setFromXml(xml_string):
    tree = ET.ElementTree(ET.fromstring('<foo>' + xml_string + '</foo>'))
    root = tree.getroot()
    elements = []
    for e in root.iter():
        element = None
        if e.tag == 'regex':
            element = e.text
            elements.append(element)

        if e.tag == 'rule':
            element = {
                'level': e.attrib['level']
            }
            elements.append(element)
        if e.tag == 'field':
            elements[-1]['rule'] = f'{e.attrib["name"]}|{e.text}'
        if e.tag == 'description':
            elements[-1]['description'] = e.text
    return elements

print(f'Added {new_decoders} new decoders')
if new_decoders:
    decoders_file = '<decoder name="techstore_inventory">\n\t<program_name>techstore_'
    for d in e_decoders:
        order = "".join(d.split("=(\s+)"))
        decoders_file += f'\n<decoder name = "log_type">'
        decoders_file += f'\n\t<parent>techstore_inventory</parent>'
        decoders_file += f'\n\t<regex>{d}</regex>'
        decoders_file += f'\n\t<order>{order}</order>'
        decoders_file += f'\n</decoder>'
    print(decoders_file)

print(f'Added {new_rules} new rules')
if new_rules:
    rules_file = '<group name="techstore_inventory">'
    id = 131000
    for r in e_rules:
        k, v = r['rule'].split('|')
        rules_file += f'\n\t<rule id = {id} level = {"level"}>'
        rules_file += f'\n\t\t<decoded_as>techstore_inventory</decoded_as>'
        rules_file += f'\n\t\t<field name="{k}">{v}</field>'
        rules_file += f'\n\t\t<description>{r["description"]}</description>'
        rules_file += f'\n\t</rule>'
        id += 1
    rules_file += '\n</group>'
    print(rules_file)

```

```
new_decoders, new_rules = 0, 0
for i, log in enumerate(logs):
    print(f'Log number {i}:', log)
    extra_info = log.split(':')[ -1]
    extra_fields = extra_info[1: ].split(' ')
    # Build decoder and add it if needed
    decoder = ""
    for field in extra_fields:
        k = field.split(' =')[0]
        decoder += f'{k}=(\\S+ ) '
    decoder = decoder[:-1]
    print(f'\tDecoder is: {decoder}')
    if not decoder in e_decoders:
        e_decoders.append(decoder)
        new_decoders += 1

    # Build rule and add it if needed
    blacklist = ['username', 'product', 'price', 'total']
    for field in extra_fields:
        k, v = field.split(' =')
        if k in blacklist: continue
        rule = f'{k}|{v}'
        rules_list = [e['rule'] for e in e_rules]
        if not rule in rules_list:
            description = input(f'\tWrite a description for rule {rule} or n to skip: ')
            if description == 'n': continue
            e_rules.append({
                'level': '5',
                'rule': rule,
                'description': description,
            })
            new_rules += 1
```



ESPECIALIDAD EN CALIDAD DE PROYECTOS DE SOFTWARE

Conexión con la API y uso de herramientas





Retos del proyecto

01

Entendimiento de los requerimientos del proyecto

De acuerdo a la descripción proporcionada para alcanzar el objetivo final y hacer el seguimiento adecuado de las actividades

02

Uso de la sintaxis adecuada en archivos XML de Wazuh

De acuerdo al tipo de Logs que teníamos, además del uso de la documentación para la creación de decodificadores y reglas

03

Conexión con la API de Wazuh

Crear una conexión adecuada con la API para que fuera posible realizar diversas tareas en el servidor de Wazuh.

04

Interpretación y creación de archivos

Tomar archivos de XML y su modificación dinámica para posteriormente volver a ser XML implicó manejo amplio de strings y varias revisiones en congruencia de datos.



Principales logros y aprendizajes

Sintaxis y estructura de Wazuh

Aprender a hacer uso de la sintaxis adecuada y poder realizar distintos decodificadores y reglas que se adaptaran a los diferentes tipos de Logs que podían ser proporcionados

Manejo de solicitudes y respuestas de la API

Manejar bien las solicitudes a la API y procesar bien las respuestas para obtener la información deseada.

Administración de decodificadores y reglas

El trabajo dinámico con dichas herramientas para el procesamiento de logs hizo una diferencia para la experiencia de uso de la aplicación. Dicho paso ahorra mucho tiempo de desarrollo manual a los usuarios.

GRACIAS

¿Alguna duda?