Investigate Data Leakage Case

Keywords: Windows Security Event Logs, XML, Python

Topics

- Introduction to Python for Digital Forensics
- Introduction to XML
- Retrieve information from `bookstore.xml` file
- Investigate Windows Security Event Logs
  - binary logs are converted to a `.xml` file

Introduction to Python for Digital Forensics

What is Python?

- Python is a versatile, high-level programming language.
  - Known for its simplicity, readability, and extensive ecosystem.
- Key Features:
  - Readability: Clear, easily understood syntax.
  - Interpreted: No compilation required; code runs directly.
  - Cross-Platform: Works on Windows, macOS, Linux, and more.
  - Comprehensive Library: Rich standard library for various tasks.
  - Open Source: Freely available with an active community.

Use Cases

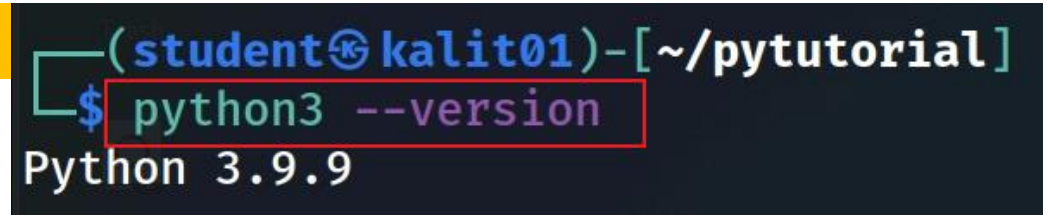
- Web Development
- Data Science
- Scientific Computing
- Automation
- Artificial Intelligence
- Digital Forensics

Learning Python for digital forensics

- Scripting for specific forensic tasks
  - parse log files, analyze registry entries, or extract metadata
- Automation: automate repetitive tasks
- Forensic Tool Development
  - perform custom analysis and investigations.
- Data Manipulation
- Data Visualization
- Community and Resources
  - many Python libraries for digital forensics

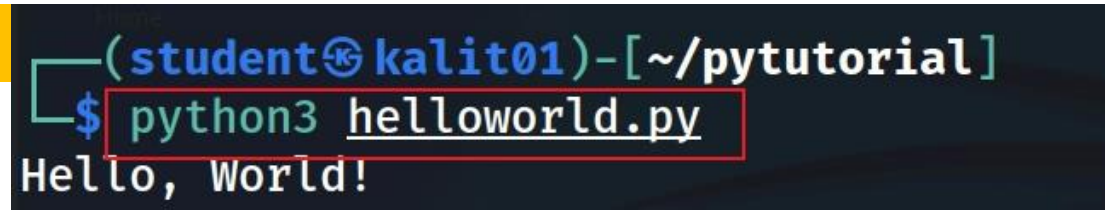
Python `helloworld.py`

Check Python version

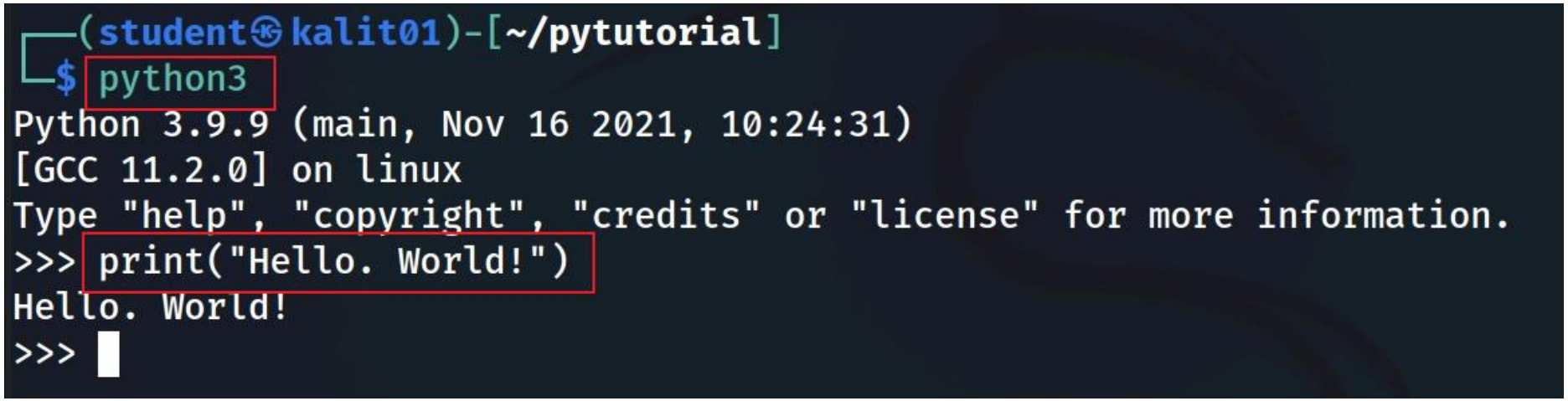


`helloworld.py`

Run python file



Python interactive model



Python Variables

https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/py_tutorial.ipynb

Open `ipynb` using https://colab.research.google.com/

```
1  # Declare a variable and initialize it
2  a = 10
3  print(a)
4  # re-declaring the variable works
5  a = 'ubalt'
6  print(a)
```

✓ 0.0s

```
10
ubalt
```

Python Conditional Statements

```
[3]
```

```python
x,y =10,1

if(x < y):
    st= "x is less than y"
else:
    st= "x is greater than y"
print (st)
print ("x is {} and y is {}".format(x,y))
```

✓ 0.0s

```
x is greater than y
x is 10 and y is 1
```

For loop over a list

```
1  #use a for loop over a collection
2  months = ["Jan","Feb","Mar","April","May"]
3  # This is a set months = {'January', 'February', 'March', 'April'}
4  for m in months:
5      print(m)
```

[8] ✓ 0.0s

```
Jan
Feb
Mar
April
May
```

Add months to a set

```
# create a month set and
# use a for loop over the  collection
month_set = set()
element1 = "Jan"
element2 = "Feb"

month_set.add(element1)
month_set.add(element2)

for m in month_set:
    print(m)
```

[17] ✓ 0.0s

```
Jan
Feb
```

Lists are ordered collections of items, while sets are unordered and have no duplicated items

Function

```
1  #define a function
2  def square(x):
3      return x*x
4
5  input = 2
6  output = square(input)
7  print("The square of {} is {}.".format(input, output))
```

✓ 0.0s

```
The square of 2 is 4.
```

Reuse functions defined in a module

Define a module `pymymodule.py` contains functions, e.g., `square()`

pymymodule.py > ...

```python
# define a function
def square(x):
    return x * x

```

reusing the function by importing the module

```python
# Import pymymodule as mm (an abbreviation)
import pymymodule as mm

# Use the square function
result = mm.square(5)
print(result)  # This will print: 25

```

[14] ✓ 0.0s

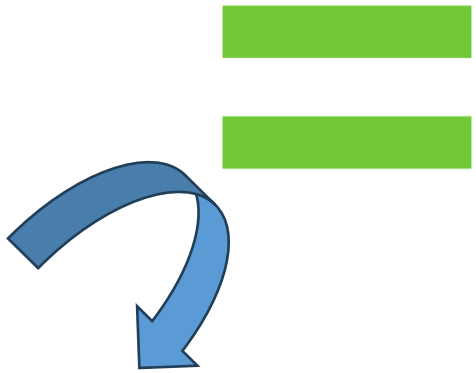
```
25
```

Save and read file

```
1  f = open("py_tutorial.txt", "w")
2  f.write("The content will be overwritten each time!")
3  f.close()
4
5  #open and read the file after the overwriting:
6  f = open("py_tutorial.txt", "r")
7  print(f.read())
```

[22] ✓ 0.0s

```
The content will be overwritten each time!
```



```
1  # Save the updated file to a new file
2  with open("py_tutorial.txt", "w") as f:
3      f.write("The content will be overwritten each time!")
4
5  #open and read the file after the overwriting:
6  with open("py_tutorial.txt", "r") as f:
7      print(f.read())
```

[24] ✓ 0.0s

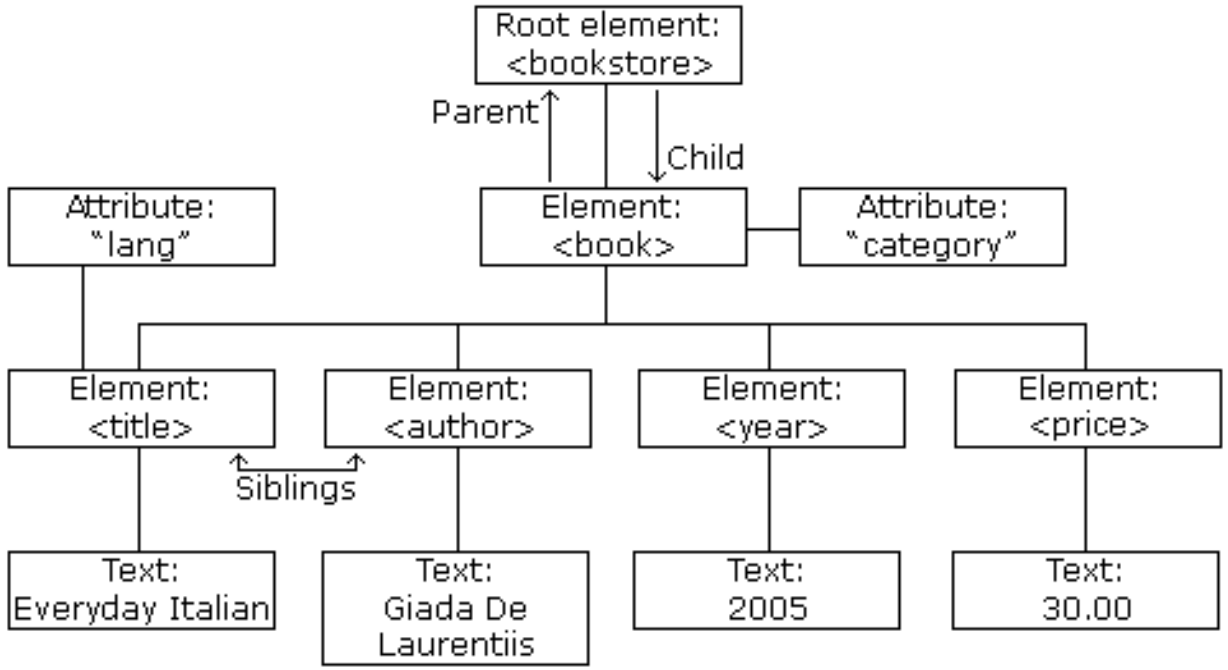
```
The content will be overwritten each time!
```

Introduction to XML

What is XML

- An XML file is an e**X**tensible **M**arkup **L**anguage file that structures data for storage and transport.
- XML files can be used for various purposes
  - Data transfer: XML files can store and retrieve data for different applications.
  - Formatting documents: XML files can define how web pages or other documents are displayed.
  - Web searching: XML files can help search engines index and rank web pages
  - Creating layouts: XML files can design the user interface of mobile applications.
  - Storing configuration data: XML files can store the settings and preferences of an application.

UTF-8 is capable of encoding all 1,112,064 characters using one to four one-byte (8-bit) code units.



bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```


https://www.w3schools.com/xml/xml_tree.asp

XML Elements vs. Attributes

- Both examples provide the same information about the language of the title.
- Avoid attributes, use elements instead

```
23  <book category="cooking">
24    <title lang="en">Everyday Italian</title>
25    <author>Giada De Laurentiis</author>
26    <year>2005</year>
27    <price>30.00</price>
28  </book>
29
30  <book >
31    <category>cooking</category>
32    <title >Everyday Italian</title>
33    <lang>en</lang>
34    <author>Giada De Laurentiis</author>
35    <year>2005</year>
36    <price>30.00</price>
37  </book>
```

Retrieve information from `bookstore.xml`

List all tags of `bookstore.xml`

bookstore_list_tags.py > ...

```python
import xml.etree.ElementTree as ET

tree = ET.parse("bookstore.xml")
root = tree.getroot()

# Create an empty set to store unique tag names
tag_names = set()

# Iterate through the elements and collect unique tag names
for element in root.iter():
    tag_names.add(element.tag)

# Convert the set to a sorted list and print the tag names
tag_list = sorted(tag_names)
for tag in tag_list:
    print(tag)

```

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Download lab xml file bookstore.xml
You can find it in the Github repo!

Download python code
It is located in the Github repo!

Execution results: list all tags

```
┌──(student㉿kalit01)-[~/mylab]
└─$ python3 bookstore_list_tags.py
author
book
bookstore
price
title
year
```

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

List a **book** attribute: **category**

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

bookstore_list_book_category_attrib.py > ...

```python
import xml.etree.ElementTree as ET

tree = ET.parse("bookstore.xml")
root = tree.getroot()

# Iterate through the book elements and print their category attributes
for book in root.findall(".//book"):  # Find all 'book' elements at any depth
    # Get book category attribute
    cate = book.attrib.get("category")
    print("book category: {}".format(cate))
```

https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/bookstore_list_book_category_attrib.py

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Execution command

```
$ python3 bookstore_list_book_category_attrib.py
```

Execution results

```
book category: cooking
book category: children
book category: web
```

List a `book` tag: `title` and its `text`

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

bookstore_list_titles_v1.py > ...

```python
import xml.etree.ElementTree as ET

tree = ET.parse("bookstore.xml")
root = tree.getroot()

# Iterate through the book elements and print their titles
for book in root.findall(".//book"):  # Find all 'book' elements at any depth
    # Find the first 'title' elements at current depth
    title_element = book.find("title")
    if title_element is not None:
        print("Book Title: {}".format(title_element.text))
```

https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/bookstore_list_book_titles_v1.py

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Execution command

```
$ python3 bookstore_list_titles_v1.py
```

Execution results

```
Book Title: Everyday Italian
Book Title: Harry Potter
Book Title: Learning XML
```

List a book tag: `category` and its `text` (alternative)

bookstore_list_titles_v2.py > ...

```python
import xml.etree.ElementTree as ET

tree = ET.parse("bookstore.xml")
root = tree.getroot()

# Find and print all the "year" elements
for title_element in root.findall(".//title"):
    print(title_element.text)
```

Show the first **book** details

bookstore_show_first_book.py > ...

```python
import xml.etree.ElementTree as ET

tree = ET.parse("bookstore.xml")
root = tree.getroot()

# Access the first child element of the root directly using indexing
first_element = root[0]

# Print the tag name and text content of the first element
print("Tag Name:", first_element.tag)
for child in first_element:
    print(f"{child.tag}: {child.text}")
```

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/bookstore_show_first_book.py

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Execution command

```
$ python3 bookstore_show_first_book.py
```

Execution results

```
Tag Name: book
title: Everyday Italian
author: Giada De Laurentiis
year: 2005
price: 30.00
```

Update one `author`

"Giada Laurentiis"

bookstore_update_one_author.py > ...

```python
import xml.etree.ElementTree as ET

tree = ET.parse("bookstore.xml")
root = tree.getroot()

# Find the "author" element with the current name and update it
for author_element in root.findall(".//author"):
    if author_element.text == "Giada De Laurentiis":
        author_element.text = "Giada Laurentiis"

# Serialize the updated XML to a string
updated_xml_content = ET.tostring(root, encoding="utf-8")

# Print the updated XML content
print(updated_xml_content.decode("utf-8"))
```

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/bookstore_update_one_author.py

Execution command

```
$ python3 bookstore_update_one_author.py
```

Execution results

```
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

bookstore.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <bookstore>
3    <book category="cooking">
4      <title lang="en">Everyday Italian</title>
5      <author>Giada De Laurentiis</author>
6      <year>2005</year>
7      <price>30.00</price>
8    </book>
9    <book category="children">
10     <title lang="en">Harry Potter</title>
11     <author>J K. Rowling</author>
12     <year>2005</year>
13     <price>29.99</price>
14   </book>
15   <book category="web">
16     <title lang="en">Learning XML</title>
17     <author>Erik T. Ray</author>
18     <year>2003</year>
19     <price>39.95</price>
20   </book>
21 </bookstore>
```

Update `price` with `price +1`

bookstore_update_price_plus1.py > ...

```python
import xml.etree.ElementTree as ET

tree = ET.parse("bookstore.xml")
root = tree.getroot()

# Find and update all the "price" elements
for price_element in root.findall(".//price"):
    current_price = float(price_element.text)
    new_price = current_price + 1
    price_element.text = str(new_price)

# Serialize the updated XML to a string
updated_xml_content = ET.tostring(root, encoding="utf-8")

# Save the updated XML to a new file
with open("bookstore_updated_price.xml", "wb") as f:
    f.write(updated_xml_content)
```

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/bookstore_update_price_plus1.py

Execution command

```
$ python3 bookstore_update_price_plus1.py
```

Execution results

bookstore_updated_price.xml

```xml
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>31.0</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>30.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>40.95</price>
  </book>
</bookstore>
```

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Remove namespace in XML

- A namespace is a mechanism that allows you to uniquely identify elements and attributes within an XML document.
  - It's used to avoid naming conflicts when different XML vocabularies or documents are combined or used together.
- Namespaces are particularly useful when
  - you have XML documents that contain elements and attributes with the same names, but they are intended to represent different things.

bookstore_ns.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore xmlns="http://schemas.exampl.com">
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

- XML Namespaces provide a method to avoid element name conflicts.
  - defined by an **xmlns** attribute in the start tag of an element.
  - xmlns:*prefix*="*URI*".

Download `bookstore` with the namespace

```
┌──(student㉿kalit01)-[~/mylab]
└─$ wget -q https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/bookstore_ns.xml

┌──(student㉿kalit01)-[~/mylab]
└─$ ls -l bookstore_ns.xml
-rw-r--r-- 1 student student 587 Sep 15 21:13 bookstore_ns.xml
```

bookstore_remove_ns.py > ...

```python
import xml.etree.ElementTree as ET

tree = ET.parse("bookstore.xml")
root = tree.getroot()


# Define a function to recursively remove all namespace prefixes
def remove_namespace_prefix(element):
    print(element.tag)
    element.tag = element.tag.split("}", 1)[-1]  # Remove namespace prefix
    for child in element:
        remove_namespace_prefix(child)


# Remove namespace prefixes from the root element and its descendants
remove_namespace_prefix(root)

# Convert the modified XML tree to a string
modified_xml = ET.tostring(root, encoding="utf-8")

# Save the updated XML to a new file
with open("bookstore_removed_ns.xml", "wb") as f:
    f.write(modified_xml)
```

https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/bookstore_remove_ns.py

Execution command

```
$ python3 bookstore_remove_ns.py
```

Execution results

bookstore_removed_ns.xml

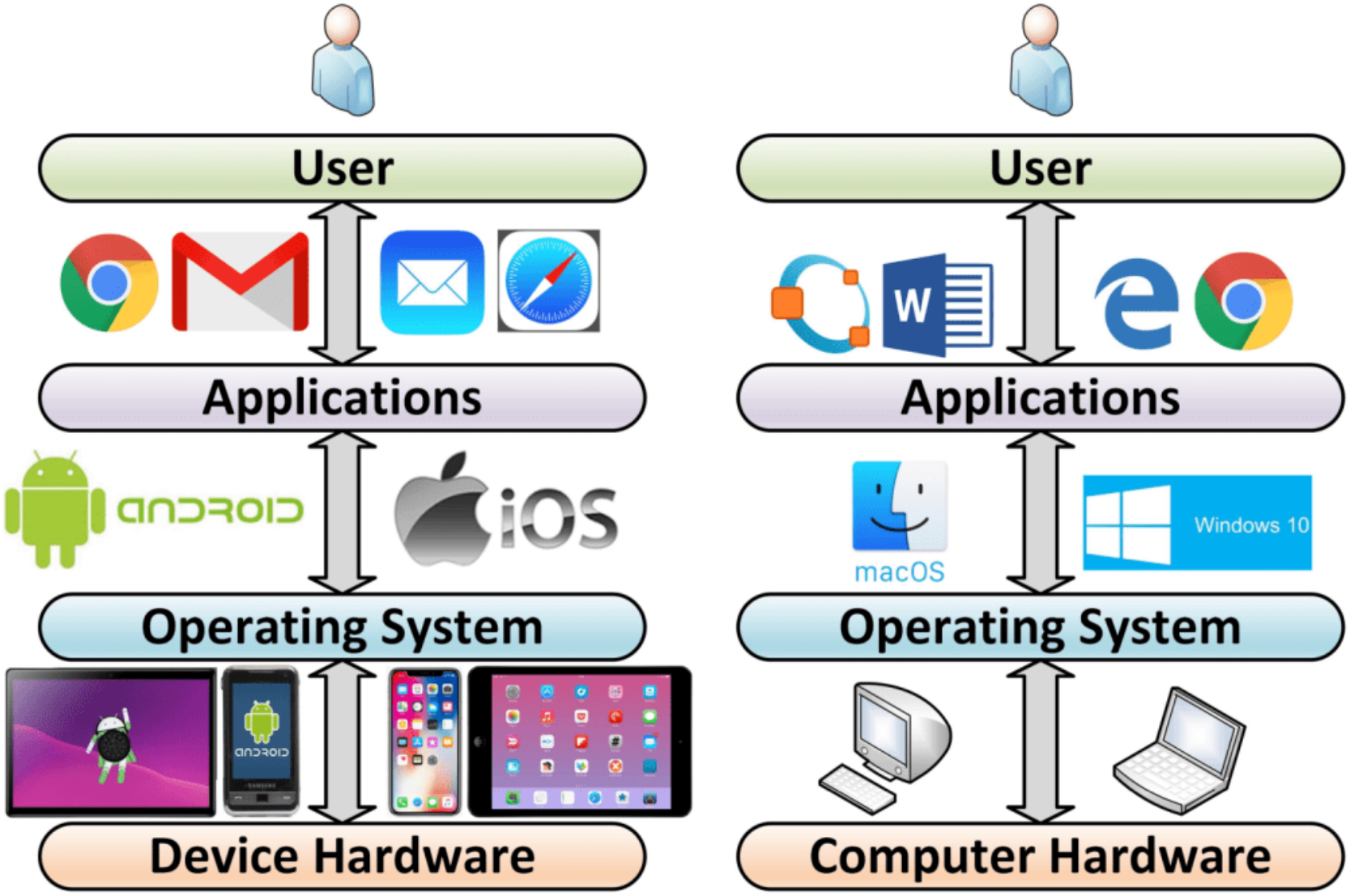
```xml
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

bookstore.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Investigate Windows Security Event Logs

Layers of a computer system and evidence generation

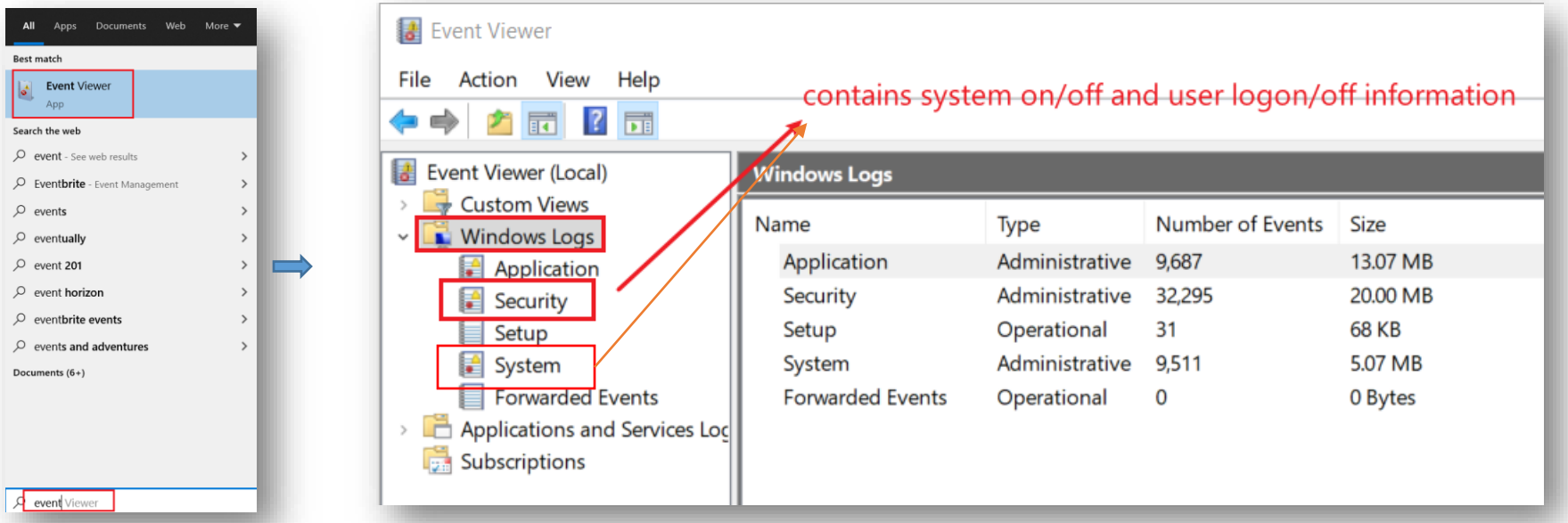


- Evidence is generated from each layer
- User activities= Applications+ OS+ Disk/Memory

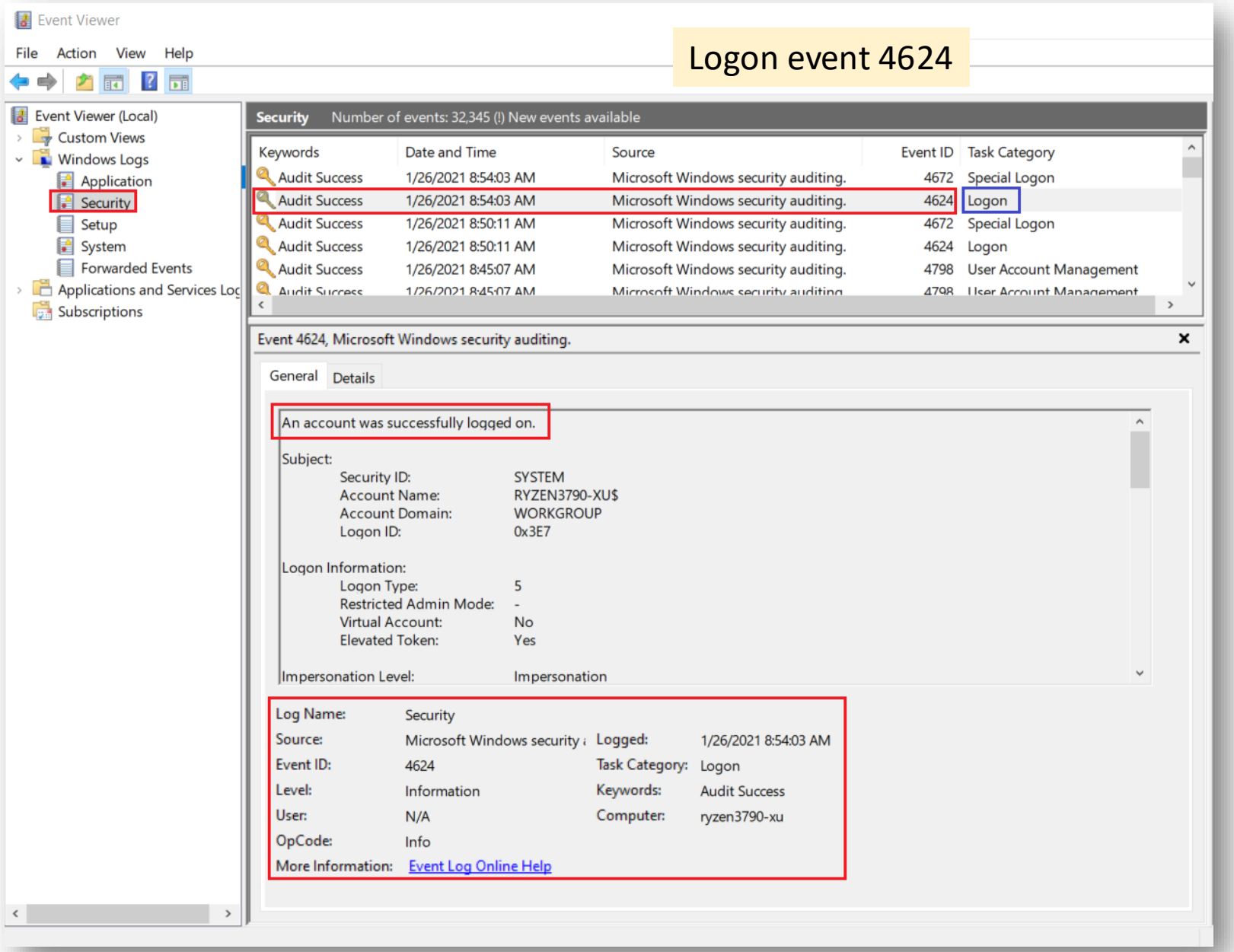
https://windsongtraining.ca/the-technology-layer-cake-users-apps-os-and-hardware/

12. List all traces about the system on/off and the user logon/logoff (It should be considered only during a time range between 09:00 and 18:00 in the timezone (Eastern Time) from Question 4.

- The Windows event log is a detailed record of system, security and application notifications stored by the Windows
- It's a useful tool for
  - troubleshooting all kinds of different Windows problems
  - forensics analysis



Logon event 4624



How to search events?

https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx

Get a copy of security event log

Please follow Environment Setup Lab and copy `Security.evtx to` the current folder

Verify the current folder contains `Security.evtx`

```
┌──(student㉿kalit01)-[~/mylab]
└─$ ls -l Security.evtx
-rwxr-xr-x 1 student student 1118208 Sep 13 19:11 Security.evtx
```

Convert `Security.evtx` to xml format

```
┌──(student㉿kalit01)-[~/mylab]
└─$ evtx_dump.py Security.evtx > SecurityEvt.xml
```

Verify .xml file

```
┌──(student㉿kalit01)-[~/mylab]
└─$ ls -l SecurityEvt.xml
-rw-r--r-- 1 student student 1555142 Sep 14 15:31 SecurityEvt.xml
```

Download the cleaned security event log

- We cleaned the `SecurityEvt.xml`
  - removed all namespaces as the attributes and tags as there are no name conflicts
  - reformatted in a pretty format (indented)
- The cleaned log file is `secuirtyEvt_formatted.xml`
  - Two Python codes for removing namespace and reformatting can be found in the repository
  - `securityevt_ns_remove.py` and `securityevt_format.py`
- You can directory download the cleaned security log

wget –q https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/securityEvt_formatted.xml

Download cleaned security event log

```
┌──(student㉿kalit01)-[~/mylab]
└─$ wget -q wget -q https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/mai
n/NIST_Data_Leakage_Case/py_version/pycode/security_evt_xml/securityEvt_formatted.xml

┌──(student㉿kalit01)-[~/mylab]
└─$ ls securityEvt_formatted.xml -l                                              4 ✕
-rw-r--r-- 1 student student 1617236 Sep 15 21:35 securityEvt_formatted.xml
```

View the cleaned security event log

securityEvt_formatted.xml

```xml
<?xml version="1.0" ?>
<Events>
  <Event>
    <System>
      <Provider Guid="{54849625-5478-4994-a5ba-3e3b0328c30d}"
      <EventID Qualifiers="">4608</EventID>
      <Version>0</Version>
      <Level>0</Level>
      <Task>12288</Task>
      <Opcode>0</Opcode>
      <Keywords>0x8020000000000000</Keywords>
      <TimeCreated SystemTime="2015-03-25 10:15:35.248869"/>
      <EventRecordID>1</EventRecordID>
      <Correlation ActivityID="" RelatedActivityID=""/>
      <Execution ProcessID="464" ThreadID="468"/>
      <Channel>Security</Channel>
      <Computer>37L4247F27-25</Computer>
      <Security UserID=""/>
    </System>
    <EventData/>
  </Event>
  <Event>
    <System>
```

Show the first **`event`**

securityevt_show_first_event.py > ...

```python
import xml.etree.ElementTree as ET

tree = ET.parse("securityEvt_formatted.xml")
root = tree.getroot()

# Find the first Event element
first_event = root.find(".//Event")

# Check if a Event element was found
if first_event is not None:
    # Convert the first Event element to a string with pretty formatting
    first_event_str = ET.tostring(first_event, encoding="unicode", method="xml")

    # Print the nicely formatted XML
    print(first_event_str)
else:
    print("No Event elements found in the XML.")
```

Download Python code

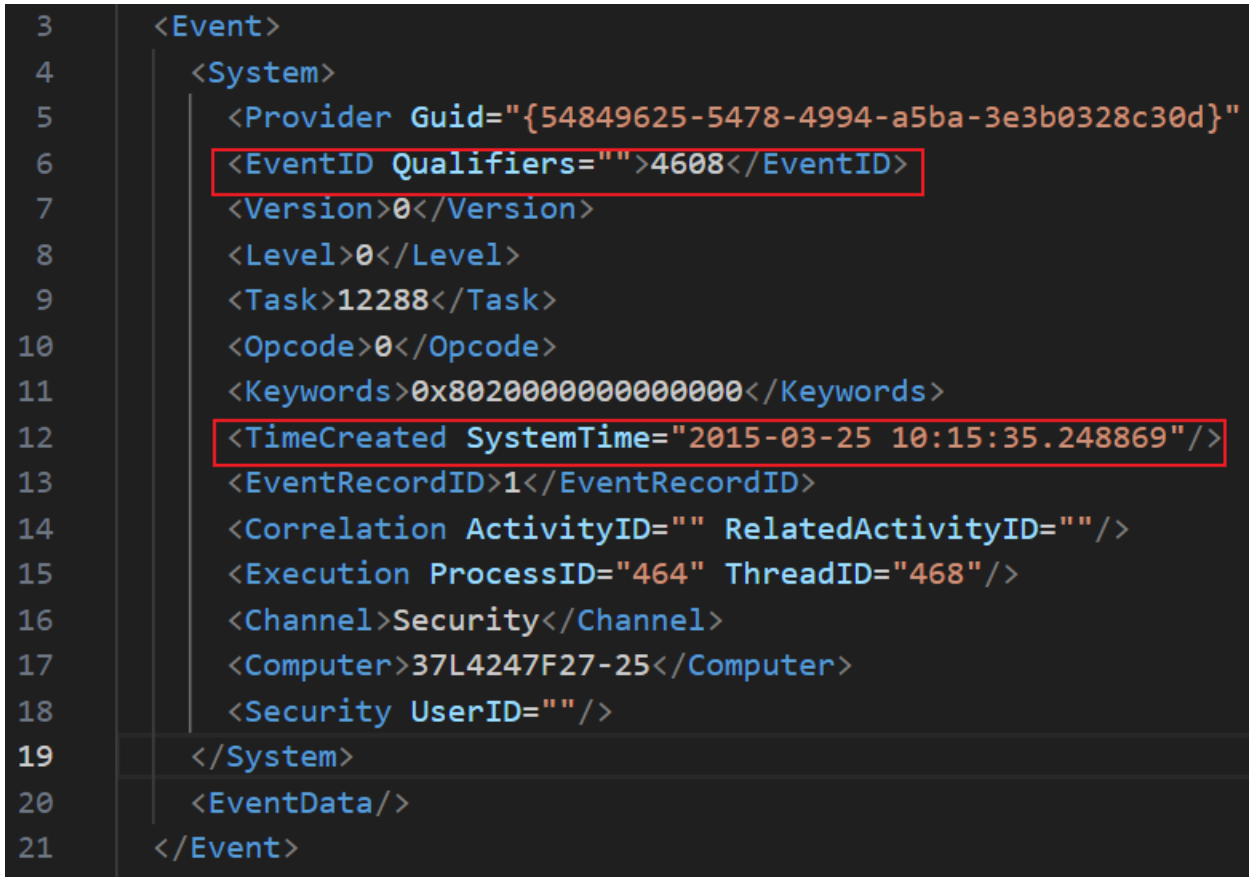
Execution command

```
┌──(student㉿kalit01)-[~/mylab]
└─$ python3 securityevt_show_first_event.py
<Event>
    <System>
      <Provider Guid="{54849625-5478-4994-a5ba-3e3b0328c30d}"
curity-Auditing" />
      <EventID Qualifiers="">4608</EventID>
      <Version>0</Version>
      <Level>0</Level>
      <Task>12288</Task>
      <Opcode>0</Opcode>
      <Keywords>0x8020000000000000</Keywords>
      <TimeCreated SystemTime="2015-03-25 10:15:35.248869" />
      <EventRecordID>1</EventRecordID>
      <Correlation ActivityID="" RelatedActivityID="" />
      <Execution ProcessID="464" ThreadID="468" />
      <Channel>Security</Channel>
      <Computer>37L4247F27-25</Computer>
      <Security UserID="" />
    </System>
    <EventData />
  </Event>
```

Find when Windows started

4608 (Windows is starting up)
1100 (service shutdown)
4624 (successful logon)
4634 (logoff)
4625 (logon failure)
4647 (a user initiated the logoff process)...

- In digital forensics, determining when Windows started (boot time) can be crucial
  - Timeline Analysis
  - Incident Reconstruction
  - Malware Analysis
  - Unauthorized Access
  - System Stability and Integrity
  - Alibis and Altered System Times
  - Correlation with Other Artifacts



Find all Event ID 4608 (Windows is starting up) and its timestamps

```python
import xml.etree.ElementTree as ET
import xml.dom.minidom as minidom

tree = ET.parse("SecurityEvt_formatted.xml")
root = tree.getroot()

# Iterate through all System elements
for system_element in root.findall(".//System"):
    event_id_element = system_element.find("EventID")
    time_created_element = system_element.find("TimeCreated")

    # Check if EventID and TimeCreated elements exist
    if (
        event_id_element is not None
        and event_id_element.text == "4608"
        and time_created_element is not None
    ):
        event_id = event_id_element.text
        system_time = time_created_element.get("SystemTime")

        # Print the lists of EventID and TimeCreated values
        print("EventIDs: {} and SystemTimes: {}".format(event_id, system_time))
```

Execution command

```
$ python3 securityevt_findall_eventid_time.py
```

Execution results

```
EventIDs: 4608 and SystemTimes: 2015-03-25 10:15:35.248869
EventIDs: 4608 and SystemTimes: 2015-03-25 10:19:26.671669
EventIDs: 4608 and SystemTimes: 2015-03-22 14:51:14.039225
EventIDs: 4608 and SystemTimes: 2015-03-22 15:22:31.655058
EventIDs: 4608 and SystemTimes: 2015-03-22 15:43:36.516434
EventIDs: 4608 and SystemTimes: 2015-03-23 17:24:23.645645
EventIDs: 4608 and SystemTimes: 2015-03-24 13:21:29.399240
EventIDs: 4608 and SystemTimes: 2015-03-25 13:05:41.968834
```

Assignments

Assignments

- Practice the lab by following PPTs
- Find the EventId “4634” (logoff) and its timestamp using Python
- Substitute the EventId “4634” with the string “logoff”
- Find the string “logoff” with timestamps