

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

```
(student㉿kalit01)-[~/pytutorial]
$ python3 --version
Python 3.9.9
```

helloworld.py

```
(student㉿kalit01)-[~/pytutorial]
$ leafpad helloworld.py 1
helloworld.py
File Edit Search Options Help
print("Hello, World!") 2
```

Run python file

```
(student㉿kalit01)-[~/pytutorial]
$ python3 helloworld.py
Hello, World!
```

Python interactive model

```
(student@kalit01)-[~/pytutorial]
$ python3
Python 3.9.9 (main, Nov 16 2021, 10:24:31)
[GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello. World!")
Hello. World!
>>> █
```

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

```
▷ ▾
 1 x,y =10,1
 2
 3 ↘ if(x < y):
 4 |   st= "x is less than y"
 5 ↘ else:
 6 |   st= "x is greater than y"
 7 print (st)
 8 print ("x is {} and y is {}".format(x,y))
[3] ✓ 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

```
▶ 
1 # create a month set and
2 # use a for loop over the collection
3 month_set = set()
4 element1 = "Jan"
5 element2 = "Feb"
6
7 month_set.add(element1)
8 month_set.add(element2)
9
10 for m in month_set:
11     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()`

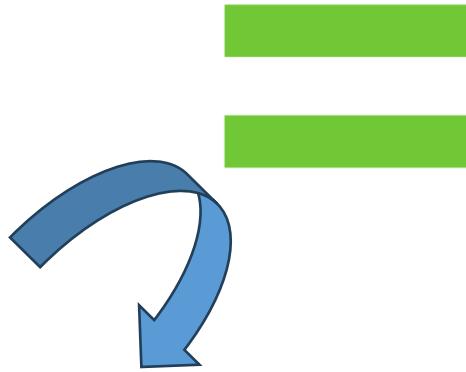
```
pymymodule.py > ...
1 # define a function
2 def square(x):
3     return x * x
4
```

reusing the function by **importing** the module

```
[14] 1 # Import pymymodule as mm (an abbreviation)
      2 import pymymodule as mm
      3
      4 # Use the square function
      5 result = mm.square(5)
      6 print(result) # This will print: 25
      7
      ...
      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!
```



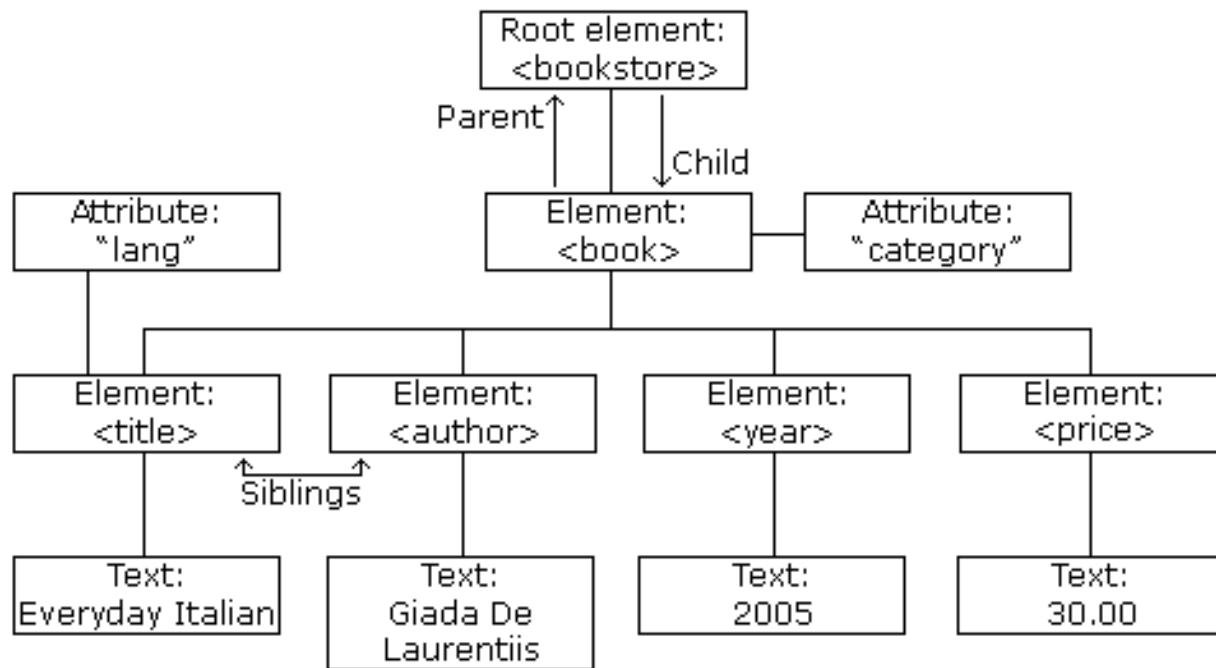
```
1 # Save the updated file to a new file
2 with open("py_tutorial.txt", "w") as f:
3     f.write("The content will be overwritten each time!")
4
5 #open and read the file after the overwriting:
6 with open("py_tutorial.txt", "r") as f:
7     print(f.read())
[24] ✓ 0.0s
... The content will be overwritten each time!
```

Introduction to XML

What is XML

- An XML file is an eXtensible Markup Language 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

```
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>
```

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 > ...

```
1 import xml.etree.ElementTree as ET
2
3 tree = ET.parse("bookstore.xml")
4 root = tree.getroot()
5
6 # Create an empty set to store unique tag names
7 tag_names = set()
8
9 # Iterate through the elements and collect unique tag names
10 for element in root.iter():
11     tag_names.add(element.tag)
12
13 # Convert the set to a sorted list and print the tag names
14 tag_list = sorted(tag_names)
15 for tag in tag_list:
16     print(tag)
17
```

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>
```

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
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>
```

List a book attribute: category

```
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>
```

```
bookstore_list_book_category_attrib.py > ...
1  import xml.etree.ElementTree as ET
2
3  tree = ET.parse("bookstore.xml")
4  root = tree.getroot()
5
6  # Iterate through the book elements and print their category attributes
7  for book in root.findall("./book"):  # Find all 'book' elements at any depth
8      # Get book category attribute
9      cate = book.attrib.get("category")
10     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

Execution command

```
$ python3 bookstore_list_book_category_attrib.py
```

Execution results

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

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>
```

List a book tag: title and its text

bookstore_list_titles_v1.py > ...

```
1 import xml.etree.ElementTree as ET
2
3 tree = ET.parse("bookstore.xml")
4 root = tree.getroot()
5
6 # Iterate through the book elements and print their titles
7 for book in root.findall("./book"): # Find all 'book' elements at any depth
8     # Find the first 'title' elements at current depth
9     title_element = book.find("title")
10    if title_element is not None:
11        print("Book Title: {}".format(title_element.text))
12
```

```
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>
```



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

Execution command

```
$ python3 bookstore_list_titles_v1.py
```

Execution results

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

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>
```

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

```
(bookstore_list_titles_v2.py) > ...
1 import xml.etree.ElementTree as ET
2
3 tree = ET.parse("bookstore.xml")
4 root = tree.getroot()
5
6 # Find and print all the "year" elements
7 for title_element in root.findall("./title"):
8     print(title_element.text)
9
```

Show the first book details

```
(bookstore_show_first_book.py) ...  
1 import xml.etree.ElementTree as ET  
2  
3 tree = ET.parse("bookstore.xml")  
4 root = tree.getroot()  
5  
6 # Access the first child element of the root directly using indexing  
7 first_element = root[0]  
8  
9 # Print the tag name and text content of the first element  
10 print("Tag Name:", first_element.tag)  
11 for child in first_element:  
12     print(f"{child.tag}: {child.text}")
```

```
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>
```



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

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
```

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 one author

"Giada Laurentiis"

bookstore_update_one_author.py > ...

```
1 import xml.etree.ElementTree as ET
2
3 tree = ET.parse("bookstore.xml")
4 root = tree.getroot()
5
6 # Find the "author" element with the current name and update it
7 for author_element in root.findall("./author"):
8     if author_element.text == "Giada De Laurentiis":
9         author_element.text = "Giada Laurentiis"
10
11 # Serialize the updated XML to a string
12 updated_xml_content = ET.tostring(root, encoding="utf-8")
13
14 # Print the updated XML content
15 print(updated_xml_content.decode("utf-8"))
```

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>
```



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 > ...

```
1  import xml.etree.ElementTree as ET
2
3  tree = ET.parse("bookstore.xml")
4  root = tree.getroot()
5
6  # Find and update all the "price" elements
7  for price_element in root.findall(".///price"):
8      current_price = float(price_element.text)
9      new_price = current_price + 1
10     price_element.text = str(new_price)
11
12  # Serialize the updated XML to a string
13  updated_xml_content = ET.tostring(root, encoding="utf-8")
14
15  # Save the updated XML to a new file
16  with open("bookstore_updated_price.xml", "wb") as f:
17      f.write(updated_xml_content)
```

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>
```



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
1 <bookstore>
2   <book category="cooking">
3     <title lang="en">Everyday Italian</title>
4     <author>Giada De Laurentiis</author>
5     <year>2005</year>
6     <price>31.0</price>
7   </book>
8   <book category="children">
9     <title lang="en">Harry Potter</title>
10    <author>J. K. Rowling</author>
11    <year>2005</year>
12    <price>30.99</price>
13  </book>
14  <book category="web">
15    <title lang="en">Learning XML</title>
16    <author>Erik T. Ray</author>
17    <year>2003</year>
18    <price>40.95</price>
19  </book>
20 </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>
```

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
1  <?xml version="1.0" encoding="UTF-8"?>
2  <bookstore xmlns="http://schemas.example.com">
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>
```

- **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_D
ata_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) > ...
1 import xml.etree.ElementTree as ET
2
3 tree = ET.parse("bookstore.xml")
4 root = tree.getroot()
5
6
7 # Define a function to recursively remove all namespace prefixes
8 def remove_namespace_prefix(element):
9     print(element.tag)
10    element.tag = element.tag.split("}", 1)[-1] # Remove namespace prefix
11    for child in element:
12        remove_namespace_prefix(child)
13
14
15 # Remove namespace prefixes from the root element and its descendants
16 remove_namespace_prefix(root)
17
18 # Convert the modified XML tree to a string
19 modified_xml = ET.tostring(root, encoding="utf-8")
20
21 # Save the updated XML to a new file
22 with open("bookstore_removed_ns.xml", "wb") as f:
23     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
```

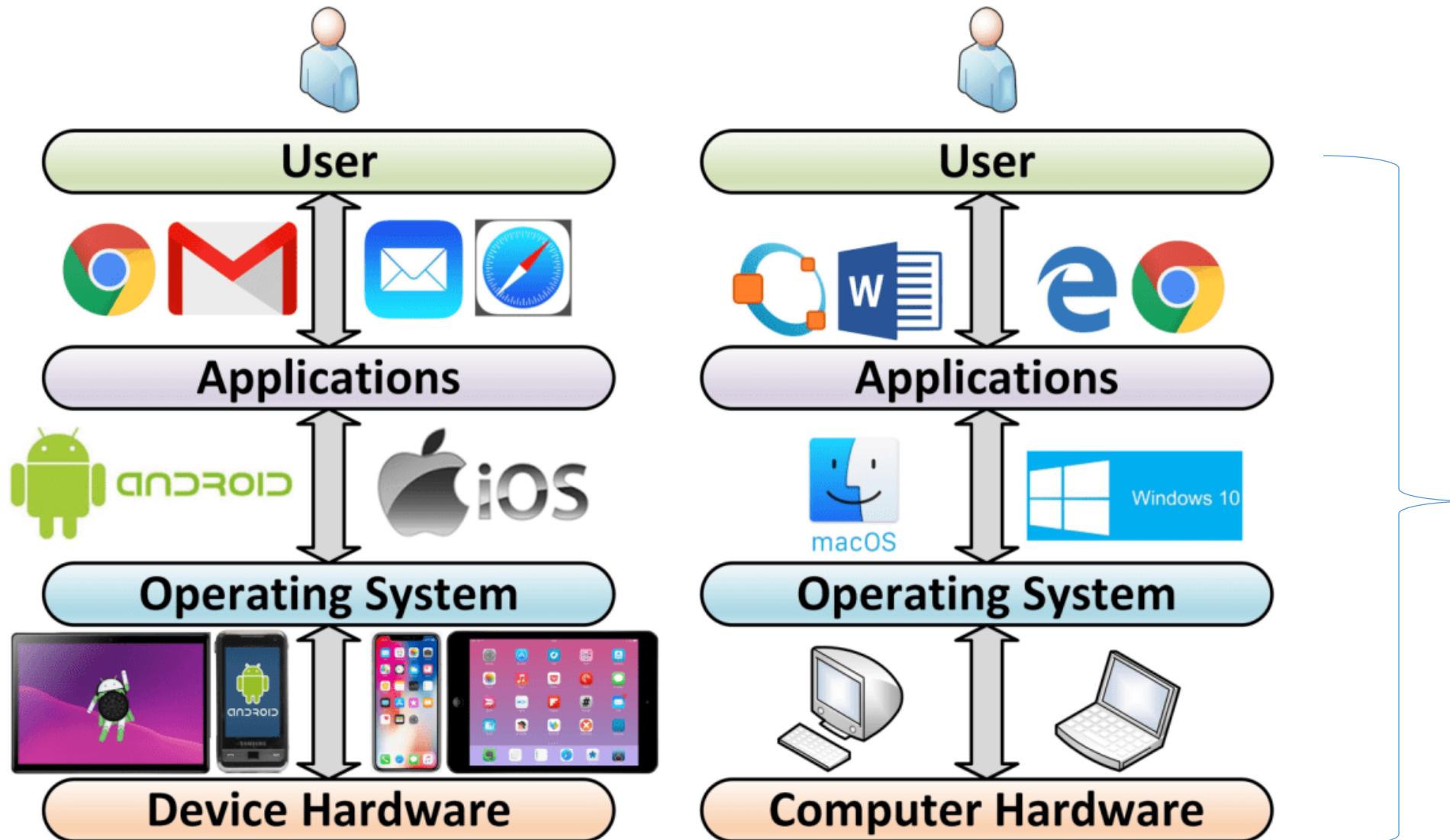
Execution results

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

Investigate Windows Security Event Logs

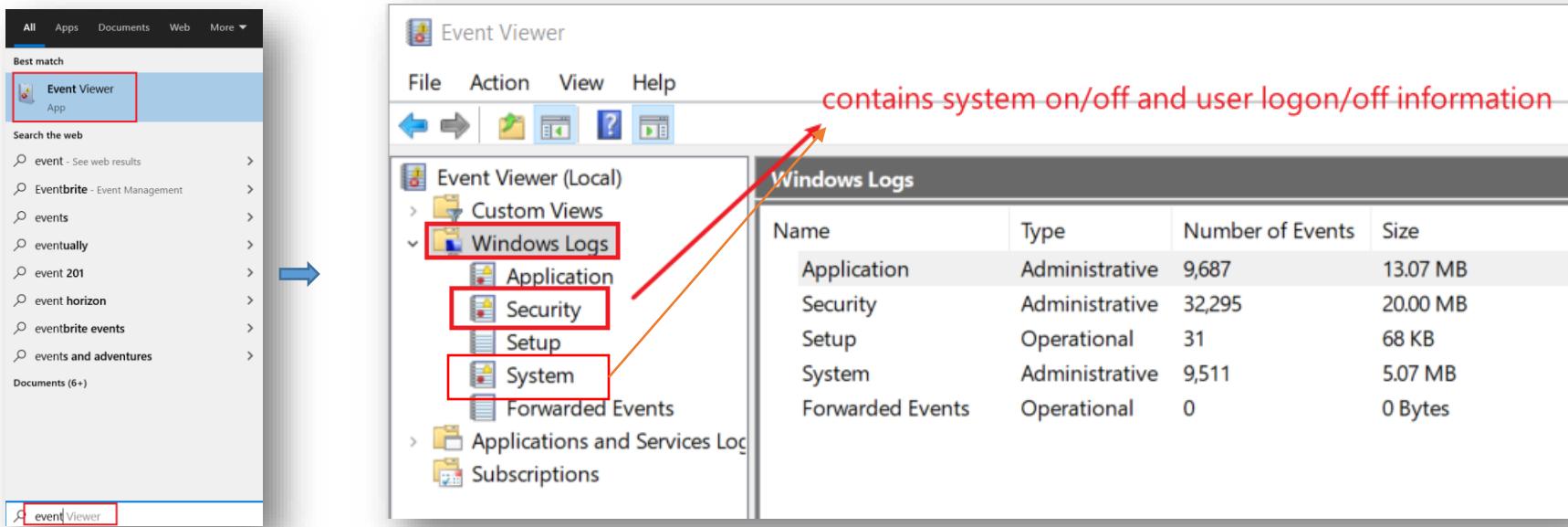
Layers of a computer system and evidence generation



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

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

The screenshot shows the Windows Event Viewer interface. The left pane displays a tree view of logs: Event Viewer (Local), Custom Views, Windows Logs (with Application, Security, Setup, System, Forwarded Events expanded), Applications and Services Logs, and Subscriptions. The Security node under Windows Logs is highlighted with a red box. The main pane shows a table of events with columns: Keywords, Date and Time, Source, Event ID, and Task Category. A specific event is selected and highlighted with a red box: "Audit Success" on 1/26/2021 8:54:03 AM from Microsoft Windows security auditing. The event ID is 4624 and the task category is Logon. Below the table, a details pane for "Event 4624, Microsoft Windows security auditing" is open, showing the General tab. The message "An account was successfully logged on." is highlighted with a red box. The subject information includes Security ID: SYSTEM, Account Name: RYZEN3790-XU\$, Account Domain: WORKGROUP, and Logon ID: 0x3E7. Logon Information shows Logon Type: 5, Restricted Admin Mode: -, Virtual Account: No, and Elevated Token: Yes. The Impersonation Level is Impersonation. The event properties at the bottom are also highlighted with a red box, listing Log Name: Security, Source: Microsoft Windows security auditing, Logged: 1/26/2021 8:54:03 AM, Event ID: 4624, Task Category: Logon, Level: Information, Keywords: Audit Success, User: N/A, Computer: ryzen3790-xu, OpCode: Info, and More Information: [Event Log Online Help](#).

Keywords	Date and Time	Source	Event ID	Task Category
Audit Success	1/26/2021 8:54:03 AM	Microsoft Windows security auditing.	4672	Special Logon
Audit Success	1/26/2021 8:54:03 AM	Microsoft Windows security auditing.	4624	Logon
Audit Success	1/26/2021 8:50:11 AM	Microsoft Windows security auditing.	4672	Special Logon
Audit Success	1/26/2021 8:50:11 AM	Microsoft Windows security auditing.	4624	Logon
Audit Success	1/26/2021 8:45:07 AM	Microsoft Windows security auditing.	4798	User Account Management
Audit Success	1/26/2021 8:45:07 AM	Microsoft Windows security auditing.	4798	User Account Management

An account was successfully logged on.

Subject:

Security ID:	SYSTEM
Account Name:	RYZEN3790-XU\$
Account Domain:	WORKGROUP
Logon ID:	0x3E7

Logon Information:

Logon Type:	5
Restricted Admin Mode:	-
Virtual Account:	No
Elevated Token:	Yes

Impersonation Level: Impersonation

Log Name: Security
Source: Microsoft Windows security auditing
Event ID: 4624
Level: Information
User: N/A
OpCode: Info
Logged: 1/26/2021 8:54:03 AM
Task Category: Logon
Keywords: Audit Success
Computer: ryzen3790-xu
More Information: [Event Log Online Help](#)

How to search events?

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

The screenshot shows the homepage of the Ultimate Windows Security Log Encyclopedia. At the top, there are banners for "LOGbinder" and "SUPERCHARGE for Windows Event Collection". Below the banner, the navigation menu includes "Home", "Security Log", "Encyclopedia" (which is currently selected), "Training", "Tools", "Webinars", "Blog", and "Forum". A sidebar on the left has links for "All Event IDs" and "Audit Policy", and a "Go To Event ID:" input field with a "Go" button. A "Security Log Quick Reference Chart" is also visible. The main content area is titled "Windows Security Log Events". It features several dropdown menus and radio buttons for filtering events:

- Source:** Radio buttons for "All Sources" (unchecked) and "Windows Audit" (checked).
- Windows Audit Categories:** A dropdown menu set to "System" with a "Subcategories" dropdown below it set to "All subcategories".
- Windows Versions:** Radio buttons for "All events", "Win2000, XP and Win2003 only", and "Win2008, Win2012R2, Win2016 and Win10+, Win2019" (checked).

Below these filters, a section titled "Category: System" lists several event IDs with their descriptions:

- Windows 4608 Windows is starting up
- Windows 4609 Windows is shutting down
- Windows 4610 An authentication package has been loaded by the Local Security Authority
- Windows 4611 A trusted logon process has been registered with the Local Security Authority

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 `securityEvt_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 directly 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_xm/l/securityEvt_formatted.xml
```

Download cleaned security event log

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

```
Welcome to Terminator's documentation!
```

```
(student@kalit01)-[~/mylab]$ ls securityEvt_formatted.xml -l  
-rw-r--r-- 1 student student 1617236 Sep 15 21:35 securityEvt_formatted.xml
```

View the cleaned security event log

```
❶ securityEvt_formatted.xml
 1   <?xml version="1.0" ?>
 2   <Events>
 3     <Event>
 4       <System>
 5         <Provider Guid="{54849625-5478-4994-a5ba-3e3b0328c30d}">
 6           <EventID Qualifiers="">4608</EventID>
 7           <Version>0</Version>
 8           <Level>0</Level>
 9           <Task>12288</Task>
10           <Opcode>0</Opcode>
11           <Keywords>0x8020000000000000</Keywords>
12           <TimeCreated SystemTime="2015-03-25 10:15:35.248869" />
13           <EventRecordID>1</EventRecordID>
14           <Correlation ActivityID="" RelatedActivityID="" />
15           <Execution ProcessID="464" ThreadID="468" />
16           <Channel>Security</Channel>
17           <Computer>37L4247F27-25</Computer>
18           <Security UserID="" />
19       </System>
20       <EventData />
21     </Event>
22     <Event>
23       <System>
24         <Provider Guid="{54849625-5478-4994-a5ba-3e3b0328c30d}">
```

Show the first event

```
securityevt_show_first_event.py > ...
1 import xml.etree.ElementTree as ET
2
3 tree = ET.parse("securityEvt_formatted.xml")
4 root = tree.getroot()
5
6 # Find the first Event element
7 first_event = root.find("./Event")
8
9 # Check if a Event element was found
10 if first_event is not None:
11     # Convert the first Event element to a string with pretty formatting
12     first_event_str = ET.tostring(first_event, encoding="unicode", method="xml")
13
14     # Print the nicely formatted XML
15     print(first_event_str)
16 else:
17     print("No Event elements found in the XML.")
18
```

[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

```
3   <Event>
4     <System>
5       <Provider Guid="{54849625-5478-4994-a5ba-3e3b0328c30d}">
6         <EventID Qualifiers="">4608</EventID>
7           <Version>0</Version>
8           <Level>0</Level>
9           <Task>12288</Task>
10          <Opcode>0</Opcode>
11          <Keywords>0x8020000000000000</Keywords>
12          <TimeCreated SystemTime="2015-03-25 10:15:35.248869">
13            <EventRecordID>1</EventRecordID>
14            <Correlation ActivityID="" RelatedActivityID="">
15              <Execution ProcessID="464" ThreadID="468"/>
16              <Channel>Security</Channel>
17              <Computer>37L4247F27-25</Computer>
18              <Security UserID="">
19            </System>
20            <EventData/>
21          </Event>
```

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

```
1 import xml.etree.ElementTree as ET
2 import xml.dom.minidom as minidom
3
4 tree = ET.parse("SecurityEvt_formatted.xml")
5 root = tree.getroot()
6
7 # Iterate through all System elements
8 for system_element in root.findall("./System"):
9     event_id_element = system_element.find("EventID")
10    time_created_element = system_element.find("TimeCreated")
11
12    # Check if EventID and TimeCreated elements exist
13    if (
14        event_id_element is not None
15        and event_id_element.text == "4608"
16        and time_created_element is not None
17    ):
18        event_id = event_id_element.text
19        system_time = time_created_element.get("SystemTime")
20
21        # Print the lists of EventID and TimeCreated values
22        print("EventIDs: {} and SystemTimes: {}".format(event_id, system_time))
23
```

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