Hacking Web Services with Burp

WSDL (Web Services Description Language) files are XML formatted descriptions about the operations of web services between clients and servers. They contain possible requests along with the parameters an application uses to communicate with a web service. This is great for penetration testers because we can test and manipulate web services all we want using the information from WSDL files. One of the best tools to use for working with HTTP requests and responses for applications is Burp. The only downside with Burp is that it does not natively support parsing of WSDL files into requests that can be sent to a web service. A common work around has been to use a tool such as Soap-UI and proxy the requests to Burp for further manipulation. I've written a plugin for Burp that takes a WSDL request and parses out the operations that are associated with the targeted web service and creates SOAP requests which can then be sent to a web service. This plugin builds upon the work done by Tom Bujok and his soap-ws project which is essentially the WSDL parsing portion of Soap-UI without the UI.

The Wsdler plugin along with all the source is located at the Github repository here: https://github.com/NetSPI/Wsdler.

Wsdler Requirements

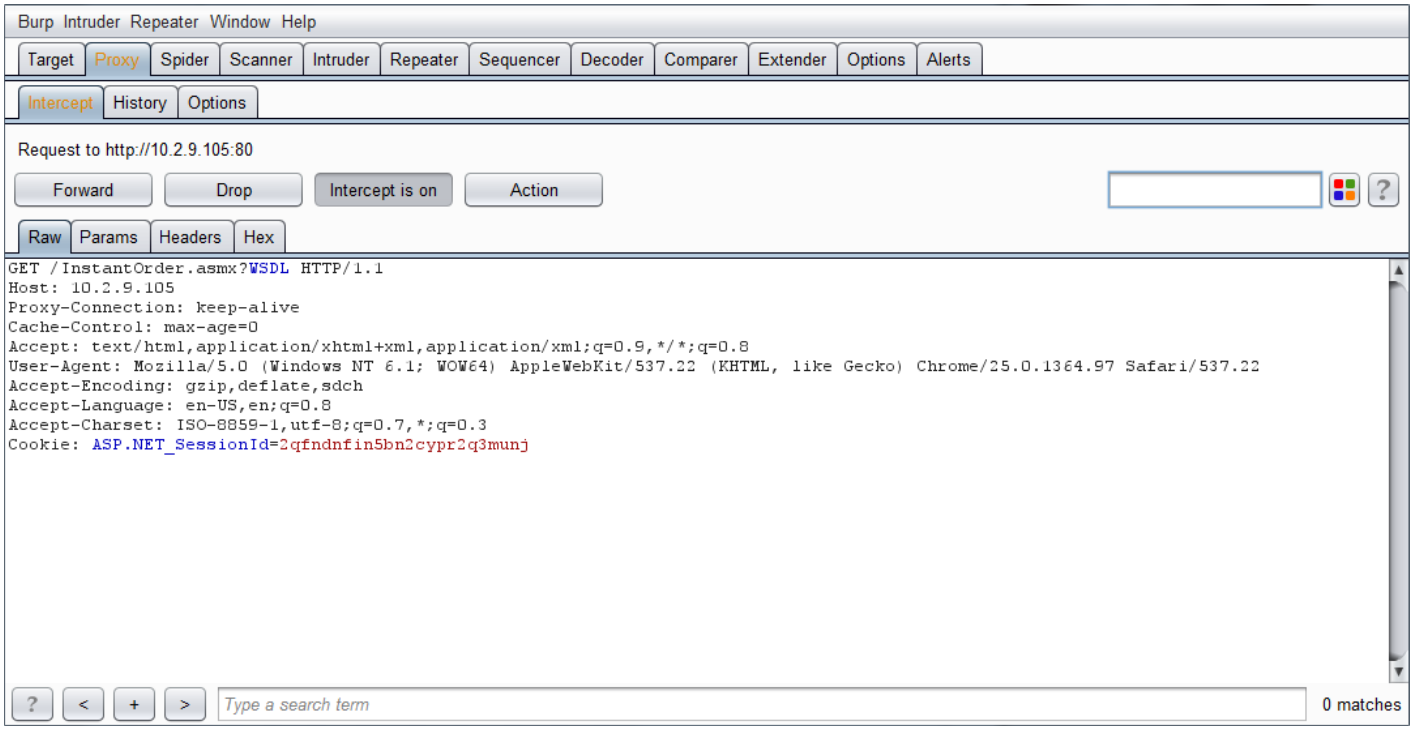
1. Burp 1.5.01 or later
2. Must be run from the command line

Starting Wsdler

The command to start Burp with the Wsdler plugin is as follows:
java -classpath Wsdler.jar;burp.jar burp.StartBurp

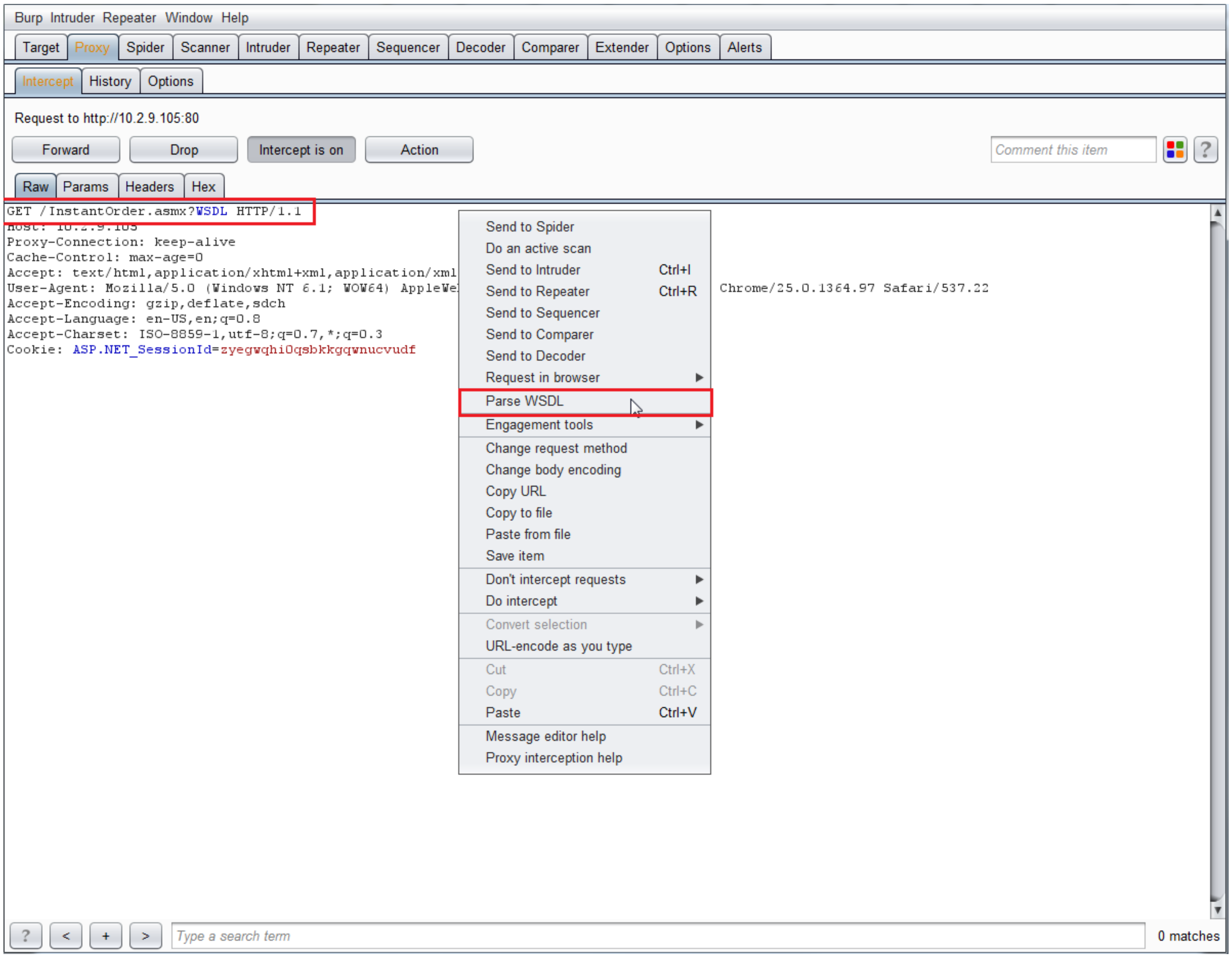
Sample Usage

Here we will intercept the request for a WSDL file belonging to an online store in Burp.


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept History Options
Request to http://10.2.9.105:80
Forward Drop Intercept is on Action
Raw Params Headers Hex
Type a search term
0 matches

```
GET /InstantOrder.asmx?WSDL HTTP/1.1
Host: 10.2.9.105
Proxy-Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.22 (KHTML, like Gecko) Chrome/25.0.1364.97 Safari/537.22
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: ASP.NET_SessionId=2qfndnfin5bn2cypr2q3munj
```

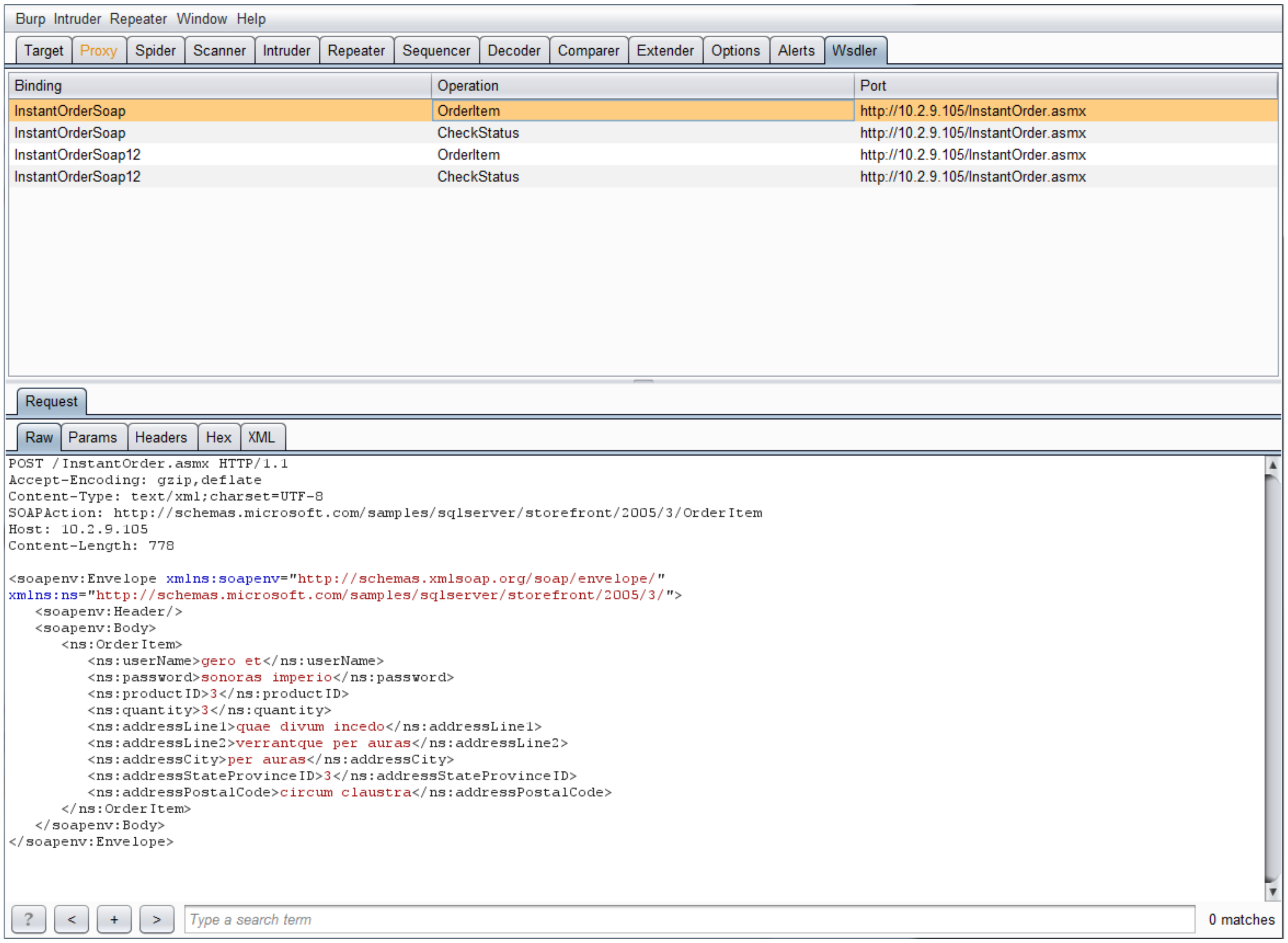
After the request for the WSDL has been intercepted, right click on the request and select Parse WSDL.



Request to http://10.2.9.105:80

```
GET /InstantOrder.asmx?WSDL HTTP/1.1
Host: 10.2.9.105
Proxy-Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWe                    Chrome/25.0.1364.97 Safari/537.22
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: ASP.NET_SessionId=zyegwqhi0qsbkkgqwnucvudf
```

A new Wsdler tab will open with the parsed operations for the WSDL, along with the bindings and ports for each of the operations. Operations are synonymous with the requests that the application supports. There are two operations in this WSDL file, OrderItem and CheckStatus. Each of these operations has two bindings, for simplicity's sake, bindings describe the format and protocol for each of the operations. The bindings for both of the operations are InstantOrderSoap and InstantOrderSoap12. The reason there are two bindings for each of the operations is because the WSDL file supports the creation of SOAP 1.1 and 1.2 requests. Finally, the "Port" for each of the operations is essentially just the URL the request will be sent to. The full specification for each of the Objects in WSDL files can be read here: http://www.w3.org/TR/wsdl.



Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts Wsdler
Request
Raw Params Headers Hex XML
? < + > Type a search term 0 matches

Binding	Operation	Port
InstantOrderSoap	OrderItem	http://10.2.9.105/InstantOrder.asmx
InstantOrderSoap	CheckStatus	http://10.2.9.105/InstantOrder.asmx
InstantOrderSoap12	OrderItem	http://10.2.9.105/InstantOrder.asmx
InstantOrderSoap12	CheckStatus	http://10.2.9.105/InstantOrder.asmx

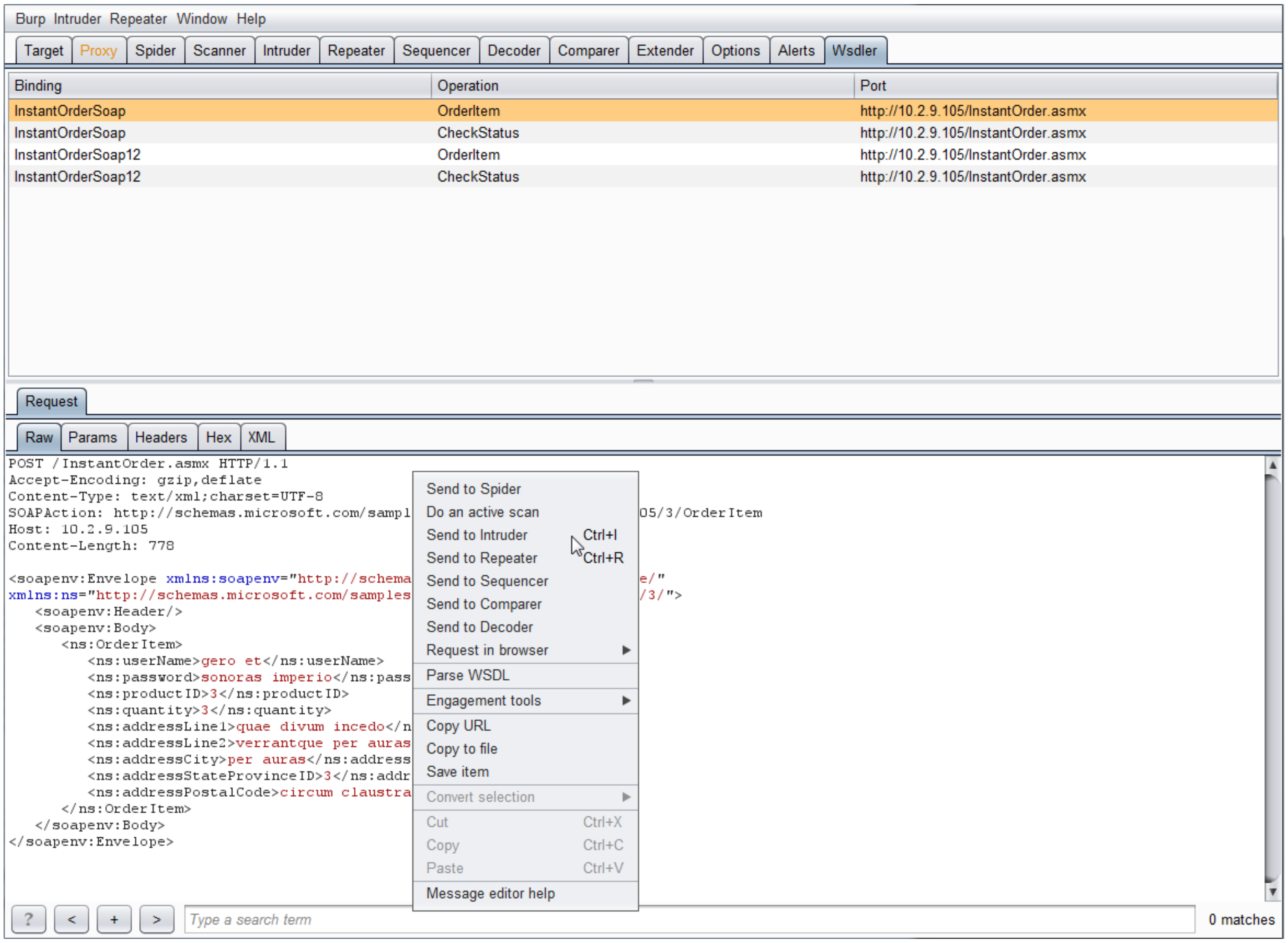
```
POST /InstantOrder.asmx HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: http://schemas.microsoft.com/samples/sqlserver/storefront/2005/3/OrderItem
Host: 10.2.9.105
Content-Length: 778

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://schemas.microsoft.com/samples/sqlserver/storefront/2005/3/">
   <soapenv:Header/>
   <soapenv:Body>
      <ns:OrderItem>
         <ns:userName>gero et</ns:userName>
         <ns:password>sonoras imperio</ns:password>
         <ns:productID>3</ns:productID>
         <ns:quantity>3</ns:quantity>
         <ns:addressLine1>quae divum incedo</ns:addressLine1>
         <ns:addressLine2>verrantque per auras</ns:addressLine2>
         <ns:addressCity>per auras</ns:addressCity>
         <ns:addressStateProvinceID>3</ns:addressStateProvinceID>
         <ns:addressPostalCode>circum claustra</ns:addressPostalCode>
      </ns:OrderItem>
   </soapenv:Body>
</soapenv:Envelope>
```

The SOAP requests for the operations will be in the lower part of the Burp window. The parsing functionality will also automatically fill in the data type for each of the parameters in the WSDL operation. In this example, strings are filled in with parts of the Aeneid and integers are filled in with numbers.

The request that Wsdler creates is a standard Burp request, so it can be sent to any other Burp function that accepts requests (intruder, repeater, etc.).

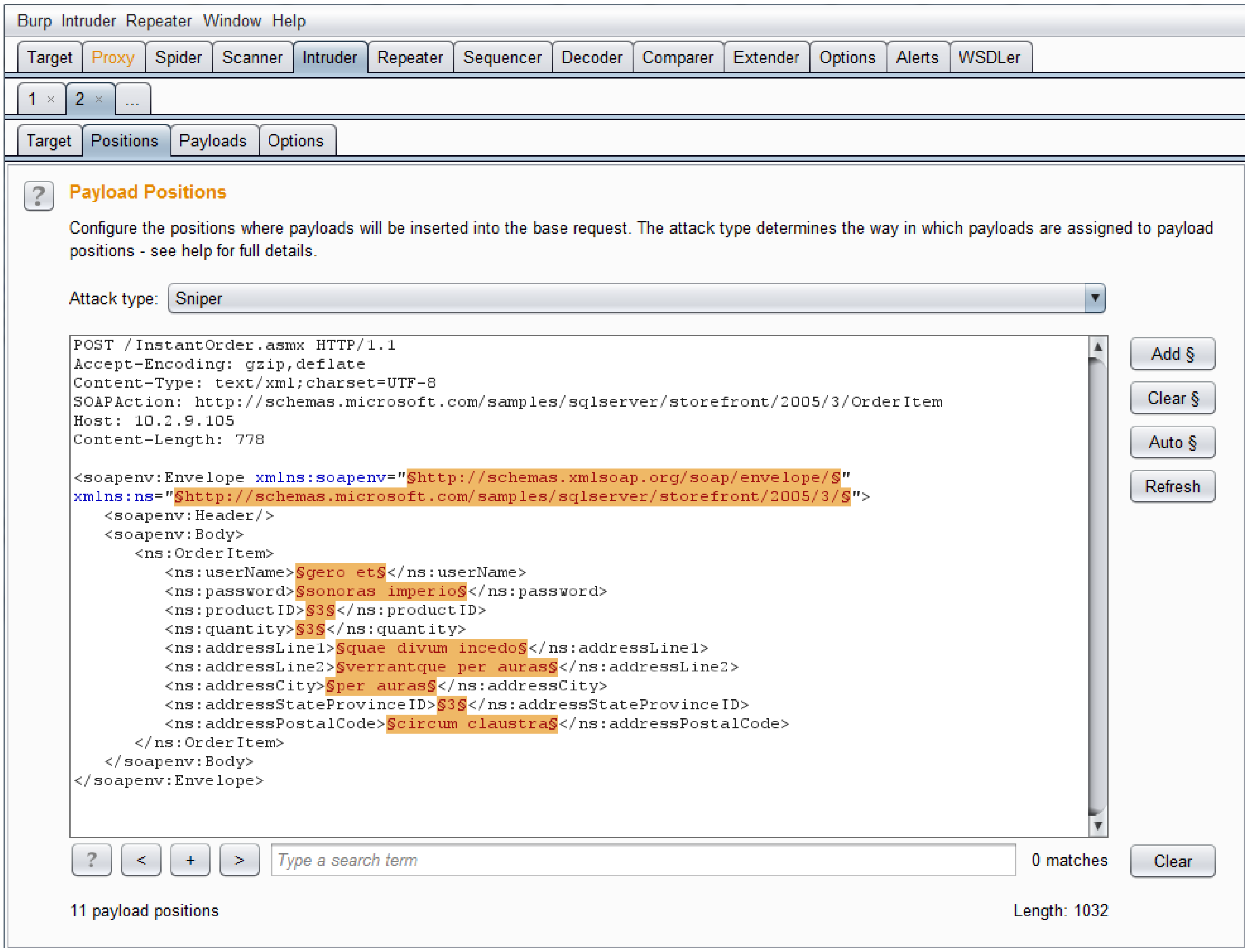
Here the request is sent to intruder for further testing. Because the request is XML, Burp automatically identifies the parameters for intruder to use.



Binding	Operation	Port
InstantOrderSoap	OrderItem	http://10.2.9.105/InstantOrder.asmx
InstantOrderSoap	CheckStatus	http://10.2.9.105/InstantOrder.asmx
InstantOrderSoap12	OrderItem	http://10.2.9.105/InstantOrder.asmx
InstantOrderSoap12	CheckStatus	http://10.2.9.105/InstantOrder.asmx

```
POST /InstantOrder.asmx HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: http://schemas.microsoft.com/sampl          05/3/OrderItem
Host: 10.2.9.105
Content-Length: 778

<soapenv:Envelope xmlns:soapenv="http://schema          e/"
xmlns:ns="http://schemas.microsoft.com/samples          /3/">
   <soapenv:Header/>
   <soapenv:Body>
      <ns:OrderItem>
         <ns:userName>gero et</ns:userName>
         <ns:password>sonoras imperio</ns:pass
         <ns:productID>3</ns:productID>
         <ns:quantity>3</ns:quantity>
         <ns:addressLine1>quae divum incedo</n
         <ns:addressLine2>verrantque per auras
         <ns:addressCity>per auras</ns:address
         <ns:addressStateProvinceID>3</ns:addr
         <ns:addressPostalCode>circum claustra
      </ns:OrderItem>
   </soapenv:Body>
</soapenv:Envelope>
```


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts WSDLer
1 × 2 × ...
Target Positions Payloads Options
Add §
Clear §
Auto §
Refresh
Clear

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
POST /InstantOrder.asmx HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: http://schemas.microsoft.com/samples/sqlserver/storefront/2005/3/OrderItem
Host: 10.2.9.105
Content-Length: 778

<soapenv:Envelope xmlns:soapenv="§http://schemas.xmlsoap.org/soap/envelope/§"
xmlns:ns="§http://schemas.microsoft.com/samples/sqlserver/storefront/2005/3/§">
   <soapenv:Header/>
   <soapenv:Body>
      <ns:OrderItem>
         <ns:userName>§gero et§</ns:userName>
         <ns:password>§sonoras imperio§</ns:password>
         <ns:productID>§3§</ns:productID>
         <ns:quantity>§3§</ns:quantity>
         <ns:addressLine1>§quae divum incedo§</ns:addressLine1>
         <ns:addressLine2>§verrantque per auras§</ns:addressLine2>
         <ns:addressCity>§per auras§</ns:addressCity>
         <ns:addressStateProvinceID>§3§</ns:addressStateProvinceID>
         <ns:addressPostalCode>§circum claustra§</ns:addressPostalCode>
      </ns:OrderItem>
   </soapenv:Body>
</soapenv:Envelope>
```

Type a search term — 0 matches

11 payload positions — Length: 1032

Conclusion

Currently, the plugin only supports WSDL specification 1.1, but there is work on supporting 1.2 / 2.0. Also, I will be adding the option to specify your own strings and integers when the plugin automatically fills in the appropriate data type for each of the parameters in the parsed operations. If there are any bugs or features that you would like to see added, send me an email or create a ticket on Github.