Burp-Plugin im Eigenbau

Wie ich lernte die API zu lieben

Bernhard Gröhling

19. März 2013

bernhard.groehling@sicsec.de
08142 - 4425 037

sic[!]sec GmbH
Industriestraße 29-31
82194 Gröbenzell

Was erwartet uns?

Das erwartet euch

Theorie:

- Was kann Burp seit v1.5.01
- Kleiner API-Rundgang

Praxis:

- Eclipse-Setup für One-Click Deployment
- Vollständiges Proxy-Plugin mit GUI in 80 Zeilen

Das erwarte ich von euch

- Schonmal mit Burp gearbeitet
- Minimale Java-Kenntnisse

Überblick Burp

Burp ist das „Schweizer Taschenmesser“ zum Pentesten von Webanwendungen

Modularer Aufbau

- Intercepting Proxy
- Spider
- Scanner (aktiv und passiv)
- Intruder
- Repeater
- Sequencer
- Decoder / Comparer
- **Extender**

Burp Extender

Große Änderungen seit v1.5.01

- Extender Tab im GUI
- Mehrere Plugins parallel möglich
- dynamische Laden / Entladen zur Laufzeit
- Unterstützung von Java, Python und Ruby

Burp Extender

Mächtiges API

- Zugriff auf HTTP Request/Responses
- Erweiterungsmöglichkeiten für Burp-Tools wie Intruder und Scanner
- Zugriff auf Laufzeitdaten (Target, History, Scan Issues...)
- Nahtlose Integration eigener GUI-Elemente in Burp-GUI

API Benutzung

Burp Extender API ist Callback-orientiert

1. Callbacks registrieren
2. Interfaces / Callback-Methoden implementieren
3. ...
4. Profit

Code Beispiel

Callbacks registrieren:

```
/*
 * Implement IBurpExtender
 */
public void registerExtenderCallbacks(IBurpExtenderCallbacks
     callbacks)
{
  callbacks.registerProxyListener(this);  // IProxyListener
}
```

Callback-Methoden implementieren:

```
/*
 * Implement IProxyListener
 */
public void processProxyMessage(boolean messageIsRequest,
    IInterceptedProxyMessage message)
{
  if (messageIsRequest) {
    // Dinge mit der ProxyMessage machen
  }
}
```

Hilfsmethoden

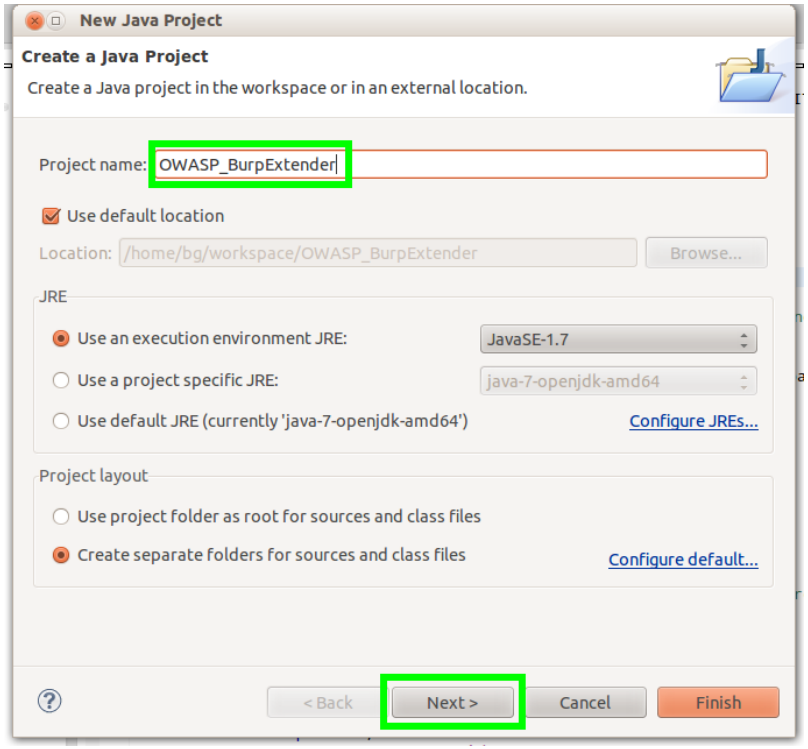
Referenz auf callbacks-Objekt merken:

```
private IBurpExtenderCallbacks callbacks;
public void registerExtenderCallbacks(IBurpExtenderCallbacks
      callbacks) {
  this.callbacks = callbacks;
}
```

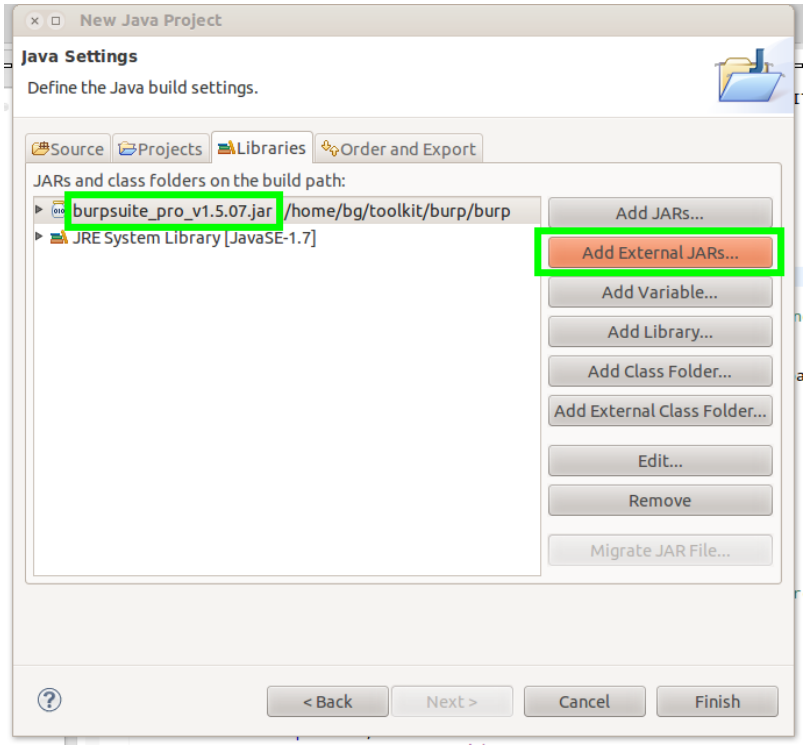
Verfügbare Hilfsmethoden:

- `analyzeRequest`
- `base64(De|En)code`
- `buildHttp(Message|Request|Service)`
- `(build|update)Parameter`
- `getRequestParameter`
- `makeScannerInsertionPoint`
- `removeParameter`
- `toggleRequestMethod`
- `url(De|En)code`

Neues Java Projekt anlegen



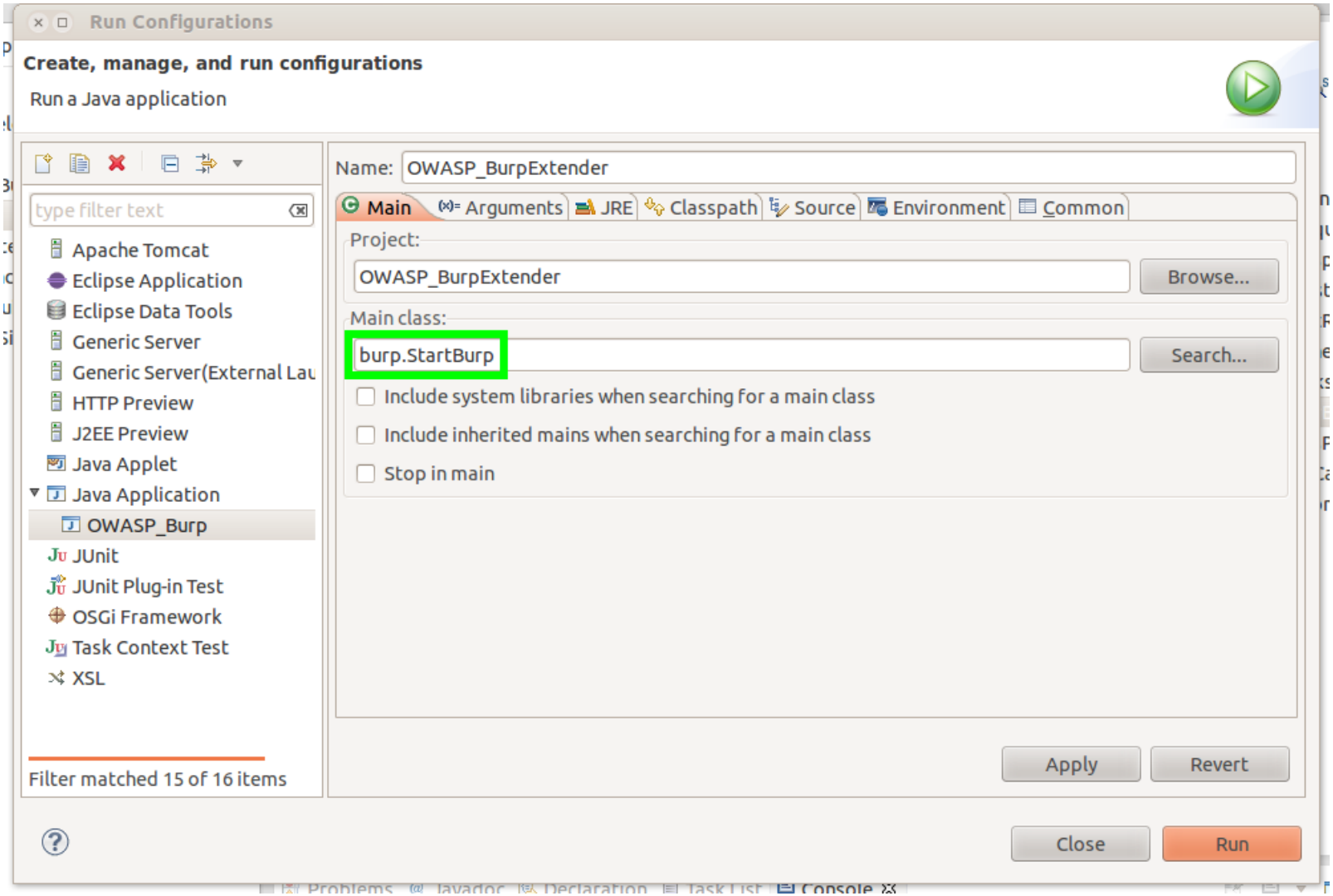
File -> New -> Java Project



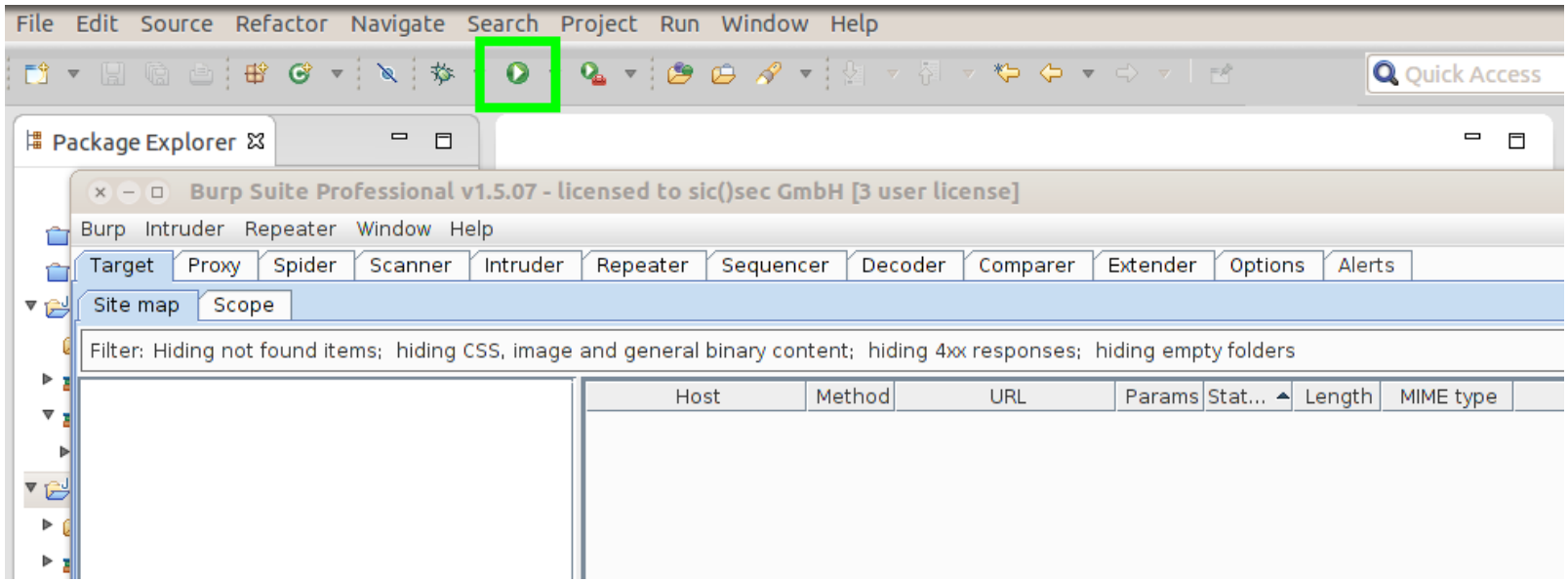
Burp als Library hinzufügen

Run Configuration

Run -> Run Configurations...

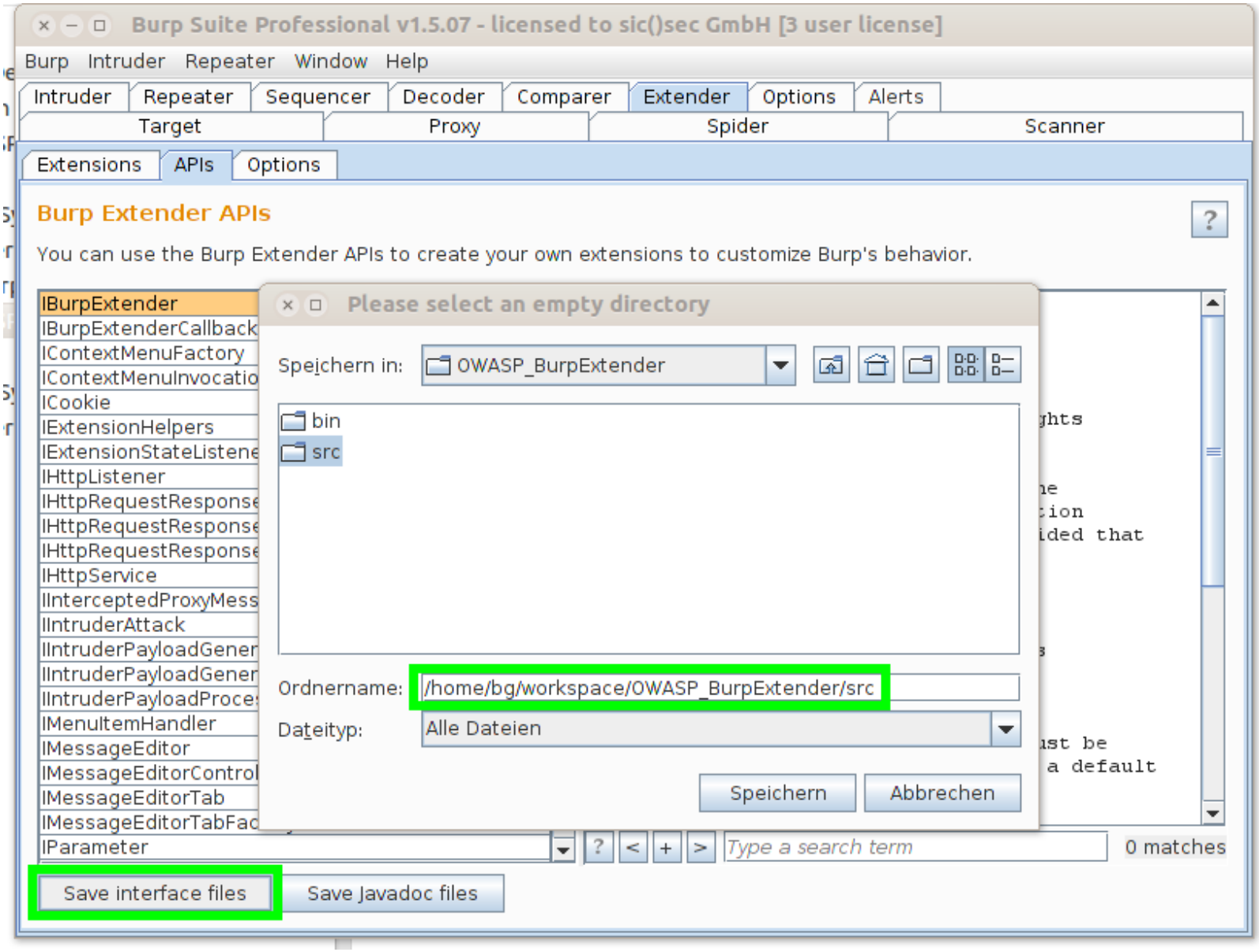


Burp starten



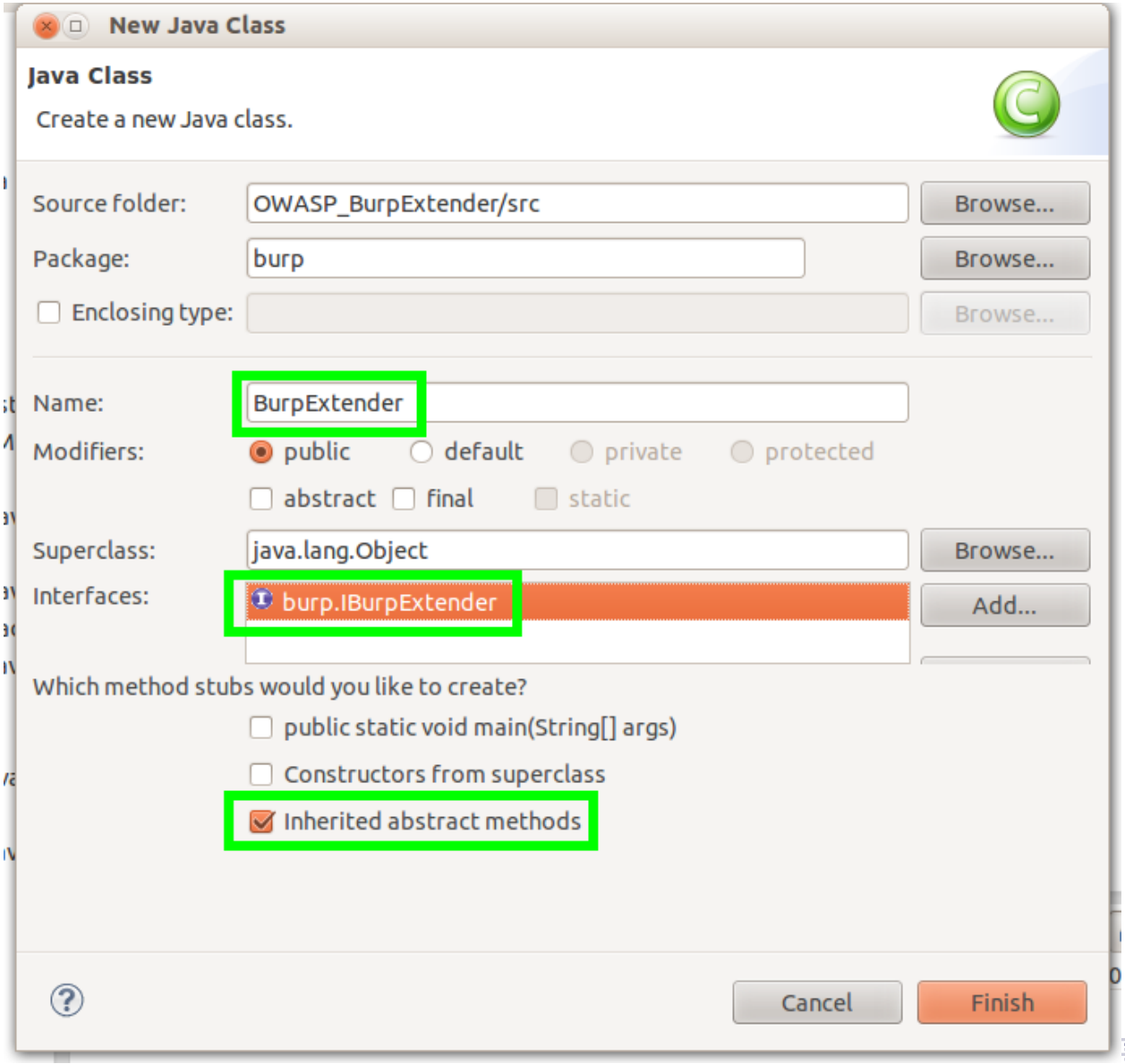
Burp Extender API Interface exportieren

Extender -> APIs -> Save interface files



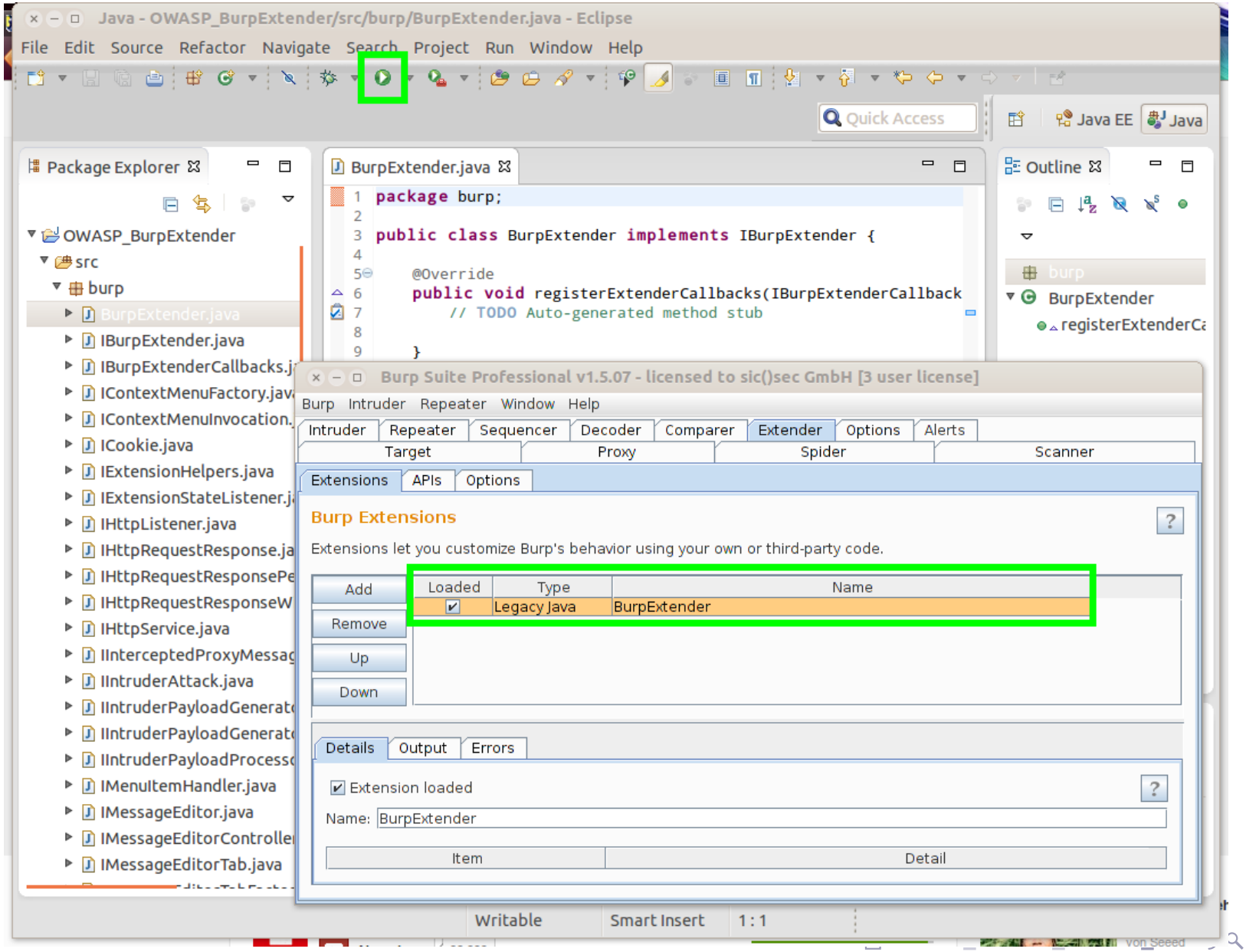
Eigenes Plugin entwickeln

- Burp Projekt aktualisieren (F5)
- Neue Klasse BurpExtender anlegen New -> Class



Eigenes Plugin testen

Burp mit „Play Button“ starten



Beispiel Plugin

Simple Statistics

- Einfache Statistik über Requests führen
- Zählt POST- und GET-Requests, Responses
- Stellt Daten in eigenem Tab dar

Benötigte Interfaces

- IBurpExtender
- IProxyListener
- ITab

Implementierung

Klasse BurpExtender mit Member Variablen

```
public class BurpExtender implements IBurpExtender,
    IProxyListener, ITab {
  private int numRequests = 0;
  private int numResponses = 0;
  private int numPostReq = 0;
  private int numGetReq = 0;
  private JTextPane textPane = new JTextPane();
  private IBurpExtenderCallbacks callbacks;
```

Implementierung

Callbacks registrieren

```
/*
 * Implement IBurpExtender
 */
@Override
public void registerExtenderCallbacks(IBurpExtenderCallbacks
      callbacks) {
  this.callbacks = callbacks;
  callbacks.setExtensionName("Simple Statistics");
  callbacks.registerProxyListener(this);  // IProxyListener
  callbacks.addSuiteTab(this);  // ITab
  callbacks.customizeUiComponent(textPane);
  this.textPane.setEnabled(false); // Kein UserInput
}
```

Implementierung

ProxyMessage verarbeiten

```
/* Implement IProxyListener */
@Override
public void processProxyMessage(boolean messageIsRequest,
    IInterceptedProxyMessage message) {
  if (messageIsRequest) { // Message is Request
    numRequests++;
    IRequestInfo reqInfo = this.callbacks.getHelpers().
        analyzeRequest(message.getMessageInfo());
    if(reqInfo.getMethod().equalsIgnoreCase("POST")) {
      this.numPostReq++;
    }
    if(reqInfo.getMethod().equalsIgnoreCase("GET")) {
      this.numGetReq++;
    }
  } else { // Message is Response
    numResponses++;
  }
  this.textPane.setText("Total GET:        "+numGetReq+"\n"+
    "Total POST :      "+numPostReq+"\n"+
    "Total Requests:   "+numRequests+"\n"+
    "Total Responses:  "+numResponses);
}
```

Implementierung

ITab implementieren

```
/*
 * Implement ITab
 */
@Override
public String getTabCaption() {
  return "Statistics";
}

/*
 * Implement ITab
 */
@Override
public Component getUiComponent() {
  return this.textPane;
}
```

Referenzen

Portswigger http://portswigger.net/

Dokumentation und Beispiele Code
http://portswigger.net/burp/extender/

Burp-Extensions im User Forum
http://forum.portswigger.net/index.cgi?
board=extensions