Global Security Days 2012

WebSeekurity - Attaques sur les applications Adobe Flex utilisant le protocole AMF

Julia Benz - 3 avril 2012

Présentation

- Julia Benz
  - Etudiante à l'Ecole Polytechnique Fédérale de Lausanne
  - Projet de Master chez SCRT Information Security

- SCRT Information Security[1]
  - Entreprise spécialisée dans la sécurité des systèmes d'information
  - Société suisse basée à Lausanne, bureau à Paris
  - Propose des services de *Ethical Hacking*

[1] http://www.scrt.ch/

Plan de la présentation

- **INTRODUCTION**
  - **Outil WebSeekurity**
- APPLICATIONS WEB
  - Applications Adobe Flex
  - Exemple
- SÉCURITÉ DES APPLICATIONS
  - Failles potentielles
  - Audits
- WEBSEEKURITY
  - Fonctionnalités
  - Implémentation
- DÉMONSTRATION
  - Présentation de WebSeekurity
  - Attaques contre une application Adobe Flex
- CONCLUSION

Introduction

SCRT effectue des audits de sécurité. Pour cela, des outils spécifiques sont utilisés.

- Si des outils répondant aux attentes de SCRT sont disponibles, ils sont utilisés.
- Sinon, SCRT développe ses propres outils en fonction de ses besoins.
- Outils déjà développés par SCRT:
  - Webshag
  - Mini MySQLat0r
  - XSSploit
  - Fireforce
- Ces outils sont distribués sous licence GNU GPL et sont disponibles sur le site Web de SCRT[1].

[1] http://www.scrt.ch/

Introduction

Les applications développées à l'aide de la technologie Flex de Adobe[2] sont de plus en plus nombreuses.

- Ces applications peuvent communiquer avec le serveur via un nouveau protocole: AMF (*Action Message Format)*[3].
- Aucun outil n'existe pour auditer ces applications.
  - *Deblaze*[4] est disponible mais ne correspond pas aux besoins de SCRT.
- SCRT a décidé de développer un tel outil:

➡ **WebSeekurity**

- Suite de la conférence : présentation de *WebSeekurity* & démonstrations

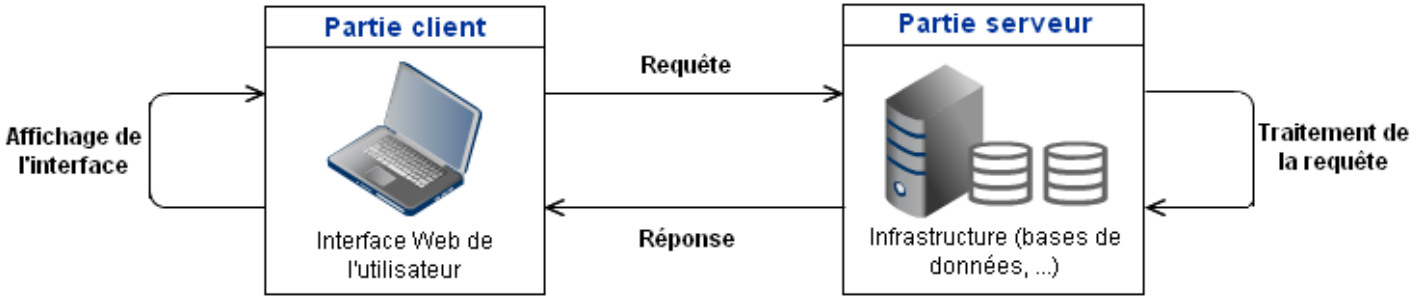
[2] http://www.adobe.com/products/flex.html
[3] http://opensource.adobe.com/wiki/download/attachments/1114283/amf3_spec_05_05_08.pdf
[4] http://deblaze-tool.appspot.com/

Plan de la présentation

- INTRODUCTION
  - Outil WebSeekurity
- **APPLICATIONS WEB**
  - **Applications Adobe Flex**
  - **Exemple**
- SÉCURITÉ DES APPLICATIONS
  - Failles potentielles
  - Audits
- WEBSEEKURITY
  - Fonctionnalités
  - Implémentation
- DÉMONSTRATION
  - Présentation de WebSeekurity
  - Attaques contre une application Adobe Flex
- CONCLUSION

Applications Web

- La partie client interagit avec la partie serveur.
  - Pour accéder à des données du côté du serveur
  - Pour exécuter des tâches du côté du serveur
- Des protocoles de communication spécifiques sont utilisés.



[5] Image conçue via: http://www.gliffy.com/

Applications Adobe Flex

La technologie Flex de Adobe permet de développer des applications Internet riches (*Rich Internet Applications* ou RIA)[2].

Applications Internet riches – Web 2.0:

- Applications plus interactives: affichage et usage de ‘‘contenus riches’’
- Exemples de RIA: Gmail, Twitter
- Se rapprochent des applications de ‘‘bureau’’
- Sont de plus en plus utilisées
- Autres technologies: Silverlight de Microsoft, JavaFX de Sun, etc.

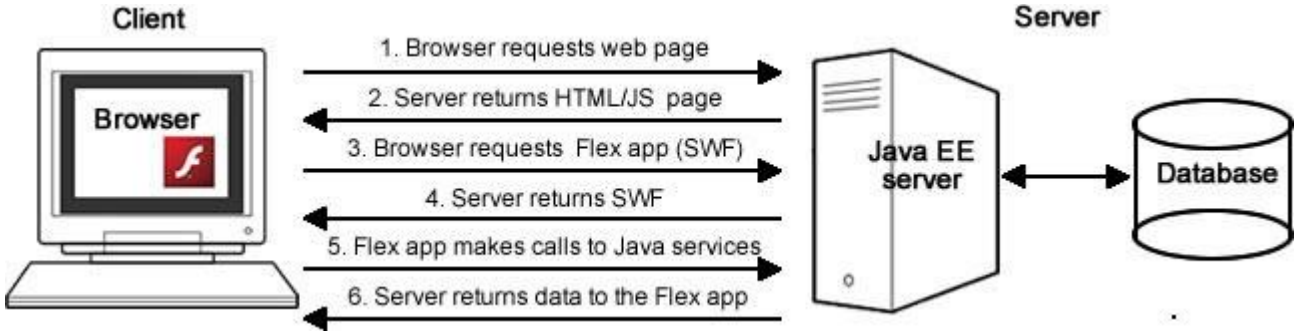
[2] http://www.adobe.com/products/flex.html

Applications Adobe Flex

Les applications Flex suivent une architecture client-serveur.

- Partie cliente:
  - Ecrite généralement en ActionScript
  - Compilée et disponible via un fichier SWF (ShockWave Flash)
  - Fichier SWF interprété par le Flash Player de Adobe
  - Interagit avec la partie serveur

Interaction client-serveur:



[6] Image prise d'un article Adobe: http://www.adobe.com/devnet/flex/articles/flex_java_architecture.html

Applications Adobe Flex

Les communications client-serveur se font via *Remote Procedure Calls* (RPC)[6].

- Le client envoie une requête au serveur contenant les détails de la procédure à exécuter.
- Le serveur traite la requête, exécute la procédure et renvoie une réponse au client.

Avec Flex, trois composants sont à disposition pour effectuer ces RPC[6].

- HTTPService: requêtes HTTP GET ou POST
- WebService: requêtes SOAP via HTTP
- RemoteObject: requêtes AMF via HTTP

[6] http://www.adobe.com/devnet/flex/articles/flex_java_architecture.html

Applications Adobe Flex

AMF: Action Message Format[3]

- Protocole développé par Adobe pour sa technologie Flex
- Dernière version: AMF3
- Format de message binaire pour encoder les RPC (les messages encodés sont ensuite transportés via le protocole HTTP)
- Permet un échange transparent des données et objets
- Avantage: jusqu'à 10 fois + rapide qu'un protocole ''textuel''

[3] http://opensource.adobe.com/wiki/download/attachments/1114283/amf3_spec_05_05_08.pdf

Applications Adobe Flex

Les fonctionnalités ***Flash Remoting*** doivent être présentes du côté du client et du serveur[6].

- Client: fournies par le Flash Player de Adobe
- Serveur: peuvent être déjà intégrées (e.g. *Coldfusion* de Adobe), sinon doivent être installées (e.g. *Zend_AMF* pour les serveurs PHP)

➡ Sérialisation/désérialisation des données échangées

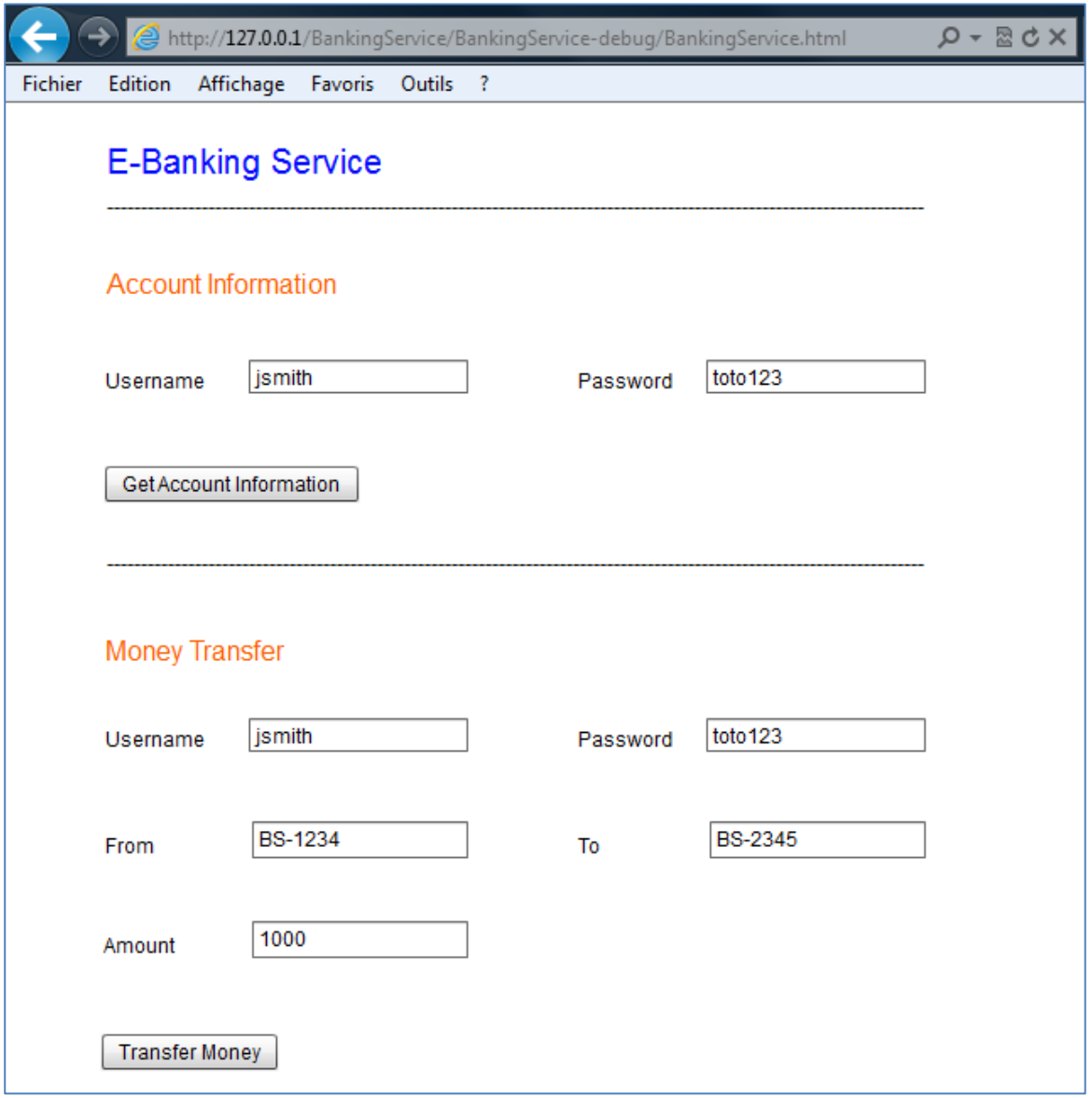
➡ Echange transparent des données et objets, indépendant des langages utilisés pour les parties client et serveur

[6] http://www.adobe.com/devnet/flex/articles/flex_java_architecture.html

Exemple

Application Adobe Flex:

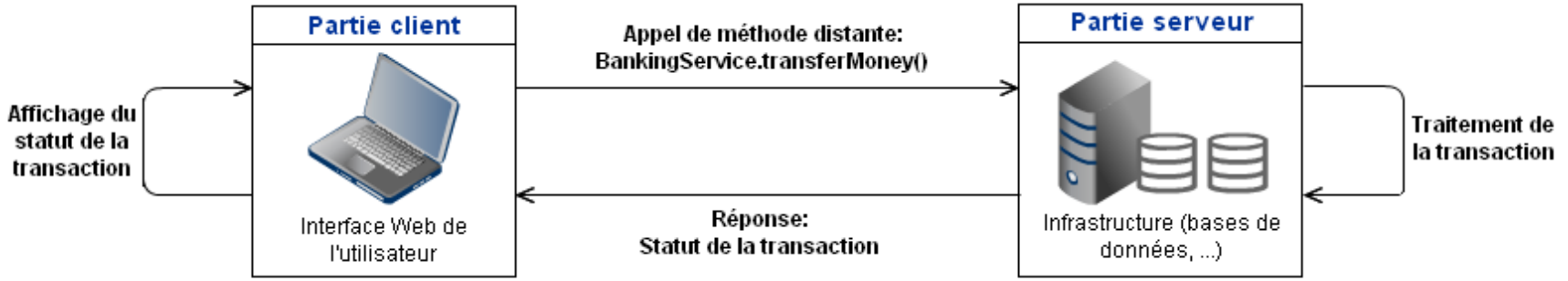
- Développée à l'aide de Flash Builder de Adobe[7]
- Application de e-banking
- Utilise AMF via HTTP pour communiquer avec le serveur



[7] http://www.adobe.com/products/flash-builder.html

Exemple

Exemple de transaction:



[5] Image conçue via: http://www.gliffy.com/

Exemple

Données à disposition:

- Utilisateur: John Smith
- Nom d'utilisateur: jsmith
- Mot de passe: toto123
- Numéro de compte: BS-12
- Autre numéro de compte: BS-23

Plan de la présentation

- INTRODUCTION
  - Outil WebSeekurity
- APPLICATIONS WEB
  - Applications Adobe Flex
  - Exemple
- **SÉCURITÉ DES APPLICATIONS**
  - **Failles potentielles**
  - **Audits**
- WEBSEEKURITY
  - Fonctionnalités
  - Implémentation
- DÉMONSTRATION
  - Présentation de WebSeekurity
  - Attaques contre une application Adobe Flex
- CONCLUSION

Sécurité des applications

- Côté client
- Côté serveur

Failles potentielles[8,9]:

- Mécanismes d'authentification et d'autorisation insuffisants
- Fuite d'informations
- Mauvaise vérification des données récupérées en entrée
  - Injections SQL
  - Autres erreurs: ‘‘format string errors’’, ‘‘buffer overflows’’
- Et beaucoup d'autres…

[8] https://www.owasp.org/index.php/Category:Vulnerability
[9] https://files.pbworks.com/download/OqCOsIJs9i/webappsec/13247059/WASC-TC-v2_0.pdf

Sécurité des applications

Audits:

- Découvrir les services et les méthodes existants, puis trouver de possibles vulnérabilités
- Outil pour automatiser la découverte des services et méthodes et faciliter l'envoi de requêtes et la réception de réponses via le protocole AMF

- **WebSeekurity**

Plan de la présentation

- INTRODUCTION
  - Outil WebSeekurity
- APPLICATIONS WEB
  - Applications Adobe Flex
  - Exemple
- SÉCURITÉ DES APPLICATIONS
  - Failles potentielles
  - Audits
- **WEBSEEKURITY**
  - **Fonctionnalités**
  - **Implémentation**
- DÉMONSTRATION
  - Présentation de WebSeekurity
  - Attaques contre une application Adobe Flex
- CONCLUSION

WebSeekurity

Fonctionnalités de l'outil:

- Génération de requêtes
  - Création manuelle de requêtes
    - Possibilité d'envoyer des objets comme paramètres
  - Découverte automatique des services et méthodes
  - *Fuzzing* des paramètres

  ➡ Plusieurs modes sont proposés:
    - *Manual*
    - *Automatic*
    - *Fuzzing*

WebSeekurity

Fonctionnalités de l'outil (suite):

- Gestion des protocoles par ‘‘plug-in’’
  - Deux protocoles déjà implémentés: AMF et SOAP
  - Possibilité d'ajouter d'autres protocoles par la suite
- Gestion d'un historique
- Création de rapports HTML
- Deux interfaces
  - Ligne de commande: *CLI*
  - Interface graphique: *GUI*
- Multiplateforme (Linux, Mac OS, Windows)

WebSeekurity

Implémentation:

- Développé en python
- A l'aide des bibliothèques *PyAMF*[10] et *SOAPpy*[11]
- Modularité du code
  - Ajout simple de nouveaux protocoles
  - Module pour les protocoles AMF et SOAP
  - Autre module pour chaque protocole ajouté

[10] http://www.pyamf.org/index.html
[11] http://pypi.python.org/pypi/SOAPpy/

Plan de la présentation

- INTRODUCTION
  - Outil WebSeekurity
- APPLICATIONS WEB
  - Applications Adobe Flex
  - Exemple
- SÉCURITÉ DES APPLICATIONS
  - Failles potentielles
  - Audits
- WEBSEEKURITY
  - Fonctionnalités
  - Implémentation
- **DÉMONSTRATION**
  - **Présentation de WebSeekurity**
  - **Attaques contre une application Adobe Flex**
- CONCLUSION

Démonstration

Présentation de l'outil:

- Interface en ligne de commande

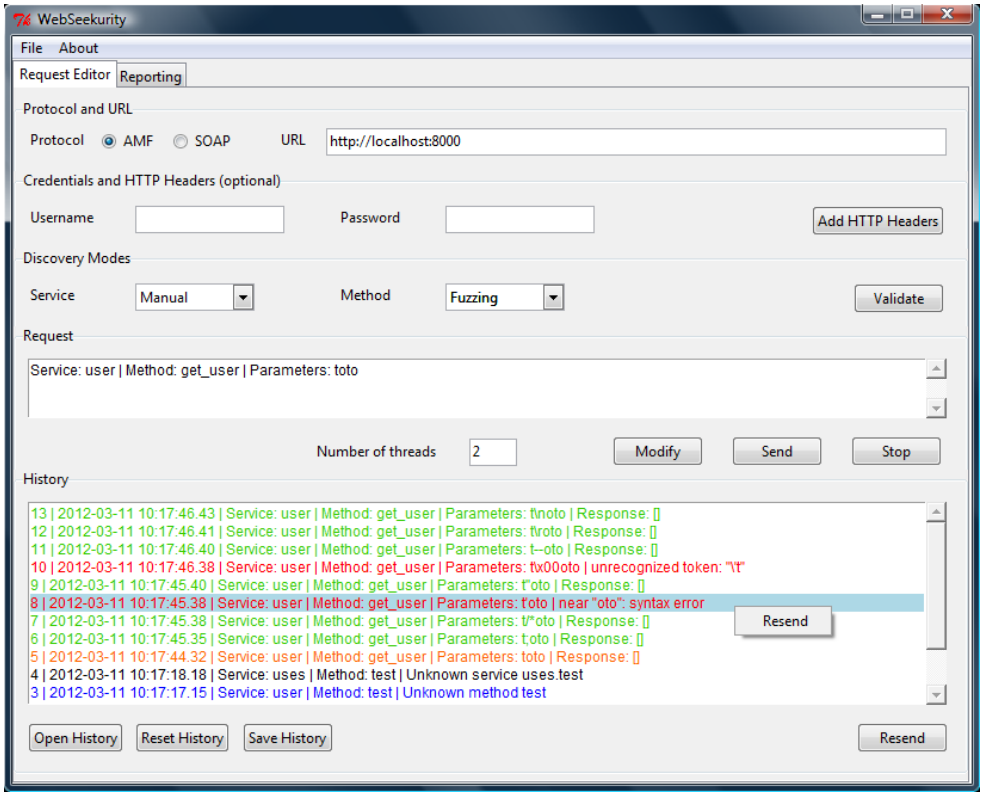
```
C:\Users\julia\Desktop\Workspace\PDM>c:\\python27\python.exe cli.py -h
usage: cli.py [-h] [-d] [-v] -po {AMF,SOAP} -u URL (-s SERVICE_NAME | -sd)
              [-ds SERVICE_DICT] [-rs  REGEX_SERVICE] [-m METHOD_NAME1]
              [-pa PARAMETERS] [-md] [-dm METHOD_DICT] [-rm  REGEX_METHOD]
              [-mf METHOD_NAME2] [-re  REGEX] [-mp MAX_MODIFIED_PARAMETERS]
              [-usr USERNAME] [-pwd PASSWORD] [-he HTTP_HEADERS]
              [-va HTTP_HEADER_VALUES] [-t NUMBER_THREADS] [-hi HISTORY_FILE]
              [-rp REPORT_FILE]

optional arguments:
  -h, --help            show this help message and exit
  -d                    If set, debugging information is displayed.
  -v                    If set, detail of the results is displayed.
  -po {AMF,SOAP}        Protocol to use.
  -u URL                URL via which the application should be reached.
  -s SERVICE_NAME       Service to test.
  -sd                   Automatic discovery of services. If no other argument
                        is specified, default dictionary is used.
  -ds SERVICE_DICT      Dictionary to use for automatic service discovery.
  -rs  REGEX_SERVICE    Regular expression to use for automatic service
                        discovery.
  -m METHOD_NAME1       Method to test.
  -pa PARAMETERS        Parameters to send along with the request. Example if
                        two parameters need to be sent: -pa toto -pa toto123 .
                        If the parameter is an object, it should be specified
                        as follows: [object_name]:object_class,object_namespac
                        e,'param1':'param1'|'param2': number|[param3]...object
                        description...[param3][object_name]
  -md                   Automatic discovery of methods. If no other argument
                        is specified, default dictionary is used.
  -dm METHOD_DICT       Dictionary to use for automatic method discovery.
  -rm  REGEX_METHOD     Regular expression to use for automatic method
                        discovery.
  -mf METHOD_NAME2      Fuzzing mode is used. Parameters or regular expression
                        are needed for this mode.
  -re  REGEX            Regular expression to use for fuzzing.
  -mp MAX_MODIFIED_PARAMETERS
                        Number max of parameters to modify at the same time
                        during fuzzing.
  -usr USERNAME         Username to use if credentials are needed.
  -pwd PASSWORD         Password to use if credentials are needed.
  -he HTTP_HEADERS      HTTP header to add.
  -va HTTP_HEADER_VALUES
                        Value corresponding to the HTTP header to add.
  -t NUMBER_THREADS     Number of threads to start.
  -hi HISTORY_FILE      History file in which to save the session (specify
                        path/history_file).
  -rp REPORT_FILE       HTML report file in which to save the session (specify
                        path/report_file.html).
```

Démonstration

Présentation de l'outil:

- Interface graphique



Démonstration

Scénario de test d'intrusion

Démonstration

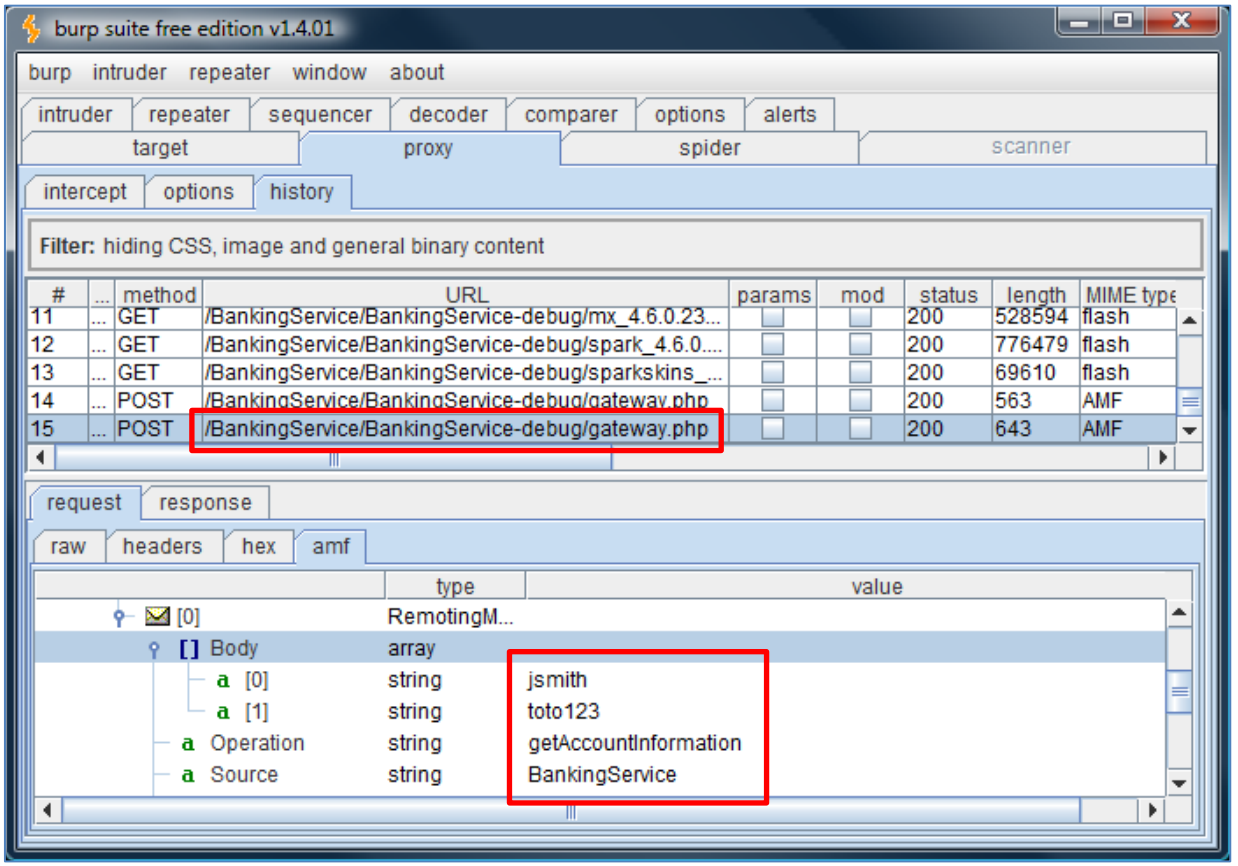
Marche à suivre[12,13]:

- Observer le trafic via un outil de proxy, décompiler le fichier SWF de l'application et analyser les résultats.
  - Déduire le protocole utilisé pour communiquer avec le serveur.
  - Trouver la passerelle à contacter.
  - Identifier les services et méthodes publiques.
- Découvrir de manière automatique quels services et méthodes sont disponibles.
- Une fois que des services et méthodes ont été découverts, effectuer des appels de méthodes.
  - Découvrir les paramètres attendus par les méthodes.
  - Construire des requêtes valides.
- Exploiter les vulnérabilités potentielles.
  - Profiter des services et méthodes non-protégés.
  - Appliquer des techniques de *fuzzing*.

[12] http://www.ivizsecurity.com/blog/web-application-security/testing-flash-applications-pen-tester-guide/
[13] http://www.gdssecurity.com/l/OWASP_NYNJMetro_Pentesting_Flex.pdf

Démonstration

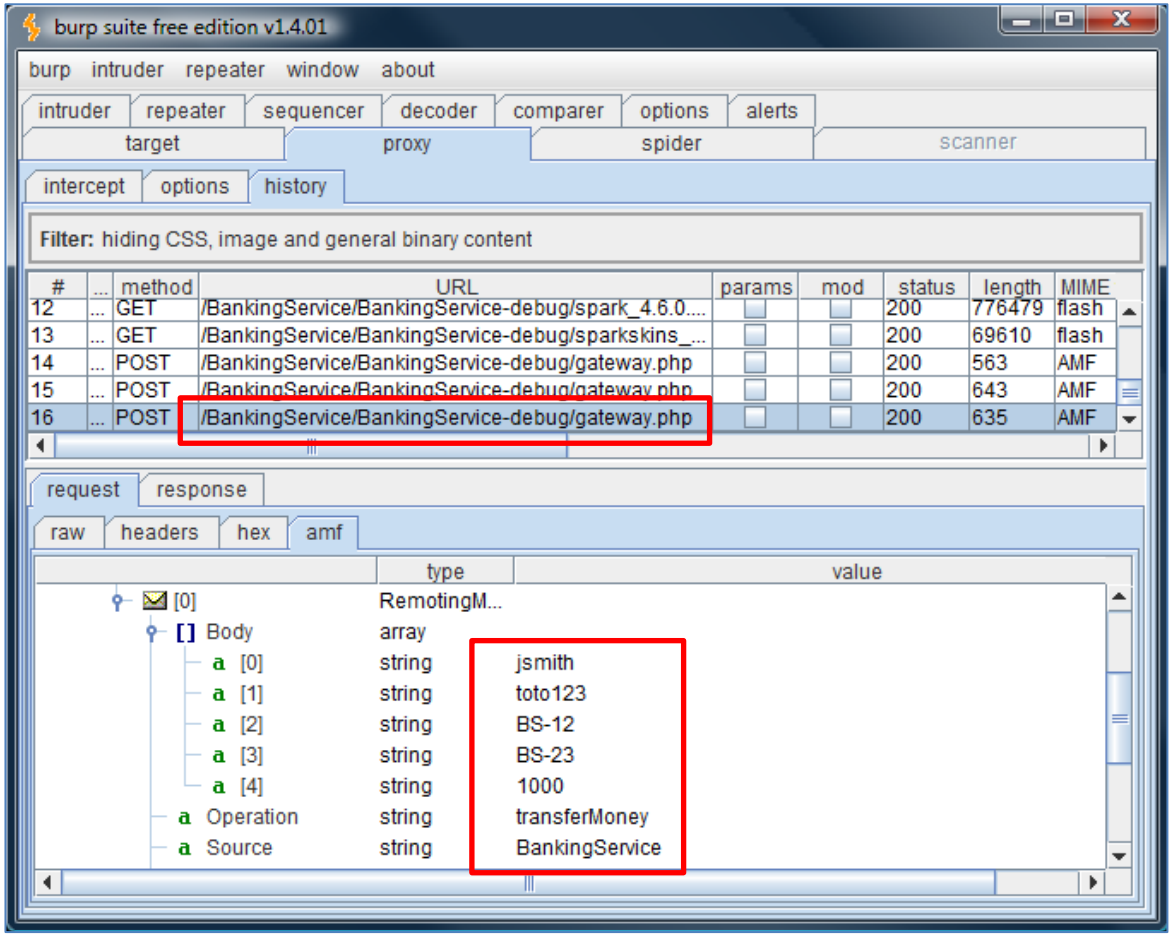
Observation du trafic[14]: Get Account Information



[14] http://www.portswigger.net/

Démonstration

Observation du trafic[14]: Transfer Money



[14] http://www.portswigger.net/

Démonstration

Déduction de l'observation:

- URL: http://127.0.0.1/BankingService/BankingService-debug/gateway.php
- Service:
  - BankingService
- Méthodes publiques:
  - getAccountInformation avec 2 paramètres:
    - nom d'utilisateur
    - mot de passe
  - transferMoney avec 5 paramètres:
    - nom d'utilisateur
    - mot de passe
    - numéro du compte à débiter
    - numéro du compte à créditer
    - montant à transférer

Démonstration

Découverte automatique des services:

- Mode pour la découverte des services: *Automatic*
- Mode pour la découverte des méthodes: *Manual*

- Service: découverte automatique grâce à un dictionnaire de noms de services
- Méthode: test

Services découverts:

- BankingService

Démonstration

Découverte automatique des méthodes:

- Mode pour la découverte des services: *Manual*
- Mode pour la découverte des méthodes: *Automatic*

- Service: BankingService
- Méthode: découverte automatique grâce à un dictionnaire de noms de méthodes

Méthodes découvertes:

- createClient
- deleteClient
- getAccount
- getClient

Démonstration

Envoi d’une requête simple:

- Mode pour la découverte des services: *Manual*
- Mode pour la découverte des méthodes: *Manual*

- Service: BankingService
- Méthode: getAccountInformation
- Nom d’utilisateur: jsmith
- Mot de passe: toto123

Démonstration

Envoi d'une requête simple:

- Mode pour la découverte des services: *Manual*
- Mode pour la découverte des méthodes: *Manual*

- Service: BankingService
- Méthode: transferMoney
- Nom d'utilisateur: jsmith
- Mot de passe: toto123
- Numéro du compte à débiter: BS-12
- Numéro du compte à créditer: BS-23

➡ Attaque: Essayer de transférer un montant d'argent négatif.

Démonstration

Rappel: injections SQL

- Insertion de requêtes SQL dans les données d'entrée d'une application
- Conséquences: une injection SQL réussie peut permettre d'accéder à une base de donnés et de pouvoir exécuter des opérations (par exemple lire, modifier, insérer ou supprimer des ressources de la base de données)
- Exemple: démonstration qui suit

Démonstration

Attaque par fuzzing:

- Mode pour la découverte des services: *Manual*
- Mode pour la découverte des méthodes: *Fuzzing*

- Service: BankingService
- Méthode: getAccountInformation
- Paramètres: nom d'utilisateur = test, mot de passe = test

Failles découvertes:

- Vulnérabilité aux injections SQL

Démonstration

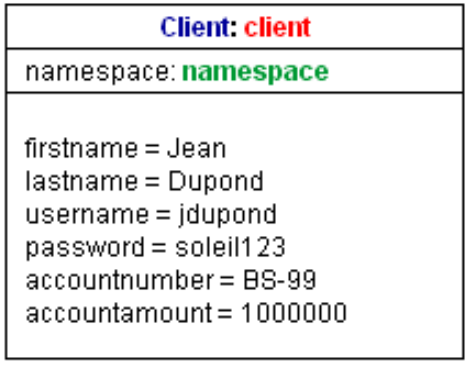
Autres fonctionnalités de WebSeekurity

Démonstration

Envoi d'une requête simple, le paramètre de la méthode est un objet:

- Mode pour la découverte des services: *Manual*
- Mode pour la découverte des méthodes: *Manual*

- Service: BankingService
- Méthode: createClient
- Objet client: [client]:Client,namespace,'firstname':'Jean'|'lastname':'Dupond'|
  'username':'jdupond'|'password':'soleil123'|'accountnumber':'BS-99'|
  'accountamount':1000000[client]



Démonstration

Fuzzing:

- Mode pour la découverte des services: *Manual*
- Mode pour la découverte des méthodes: *Fuzzing*

- Service: BankingService
- Méthode: getAccount
- Paramètres: expression régulière = BS-[0-9]{2}

Numéros de compte découverts:

- BS-12, BS-23, BS-34, BS-45, BS-99

Démonstration

Protocole SOAP, requête simple:

- Pas de service avec SOAP
- Méthode: echo
- Paramètre: toto

Démonstration

Sauvegarde de l'historique de la session:

- Base de données SQLite[15]

Génération d'un rapport HTML:

- Services et méthodes découverts
- Historique de la session enregistrée

[15] http://www.sqlite.org/

Plan de la présentation

- INTRODUCTION
  - Outil WebSeekurity
- APPLICATIONS WEB
  - Applications Adobe Flex
  - Exemple
- SÉCURITÉ DES APPLICATIONS
  - Failles potentielles
  - Audits
- WEBSEEKURITY
  - Fonctionnalités
  - Implémentation
- DÉMONSTRATION
  - Présentation de WebSeekurity
  - Attaques contre une application Adobe Flex
- **CONCLUSION**

Conclusion

WebSeekurity:

- Facilite l'audit d'applications utilisant AMF/SOAP via HTTP pour communiquer avec le serveur et en particulier des applications Adobe Flex.
- Distribué sous licence GNU GPLv2 (gratuit).
- Disponible sur le site Web de SCRT sous:

www.scrt.fr ou www.scrt.ch

- D'autres protocoles pourront être intégrés par la suite à l'outil en fonction des besoins.

Références principales

[1] http://www.scrt.ch/

[2] http://www.adobe.com/products/flex.html

[3] http://opensource.adobe.com/wiki/download/attachments/1114283/amf3_spec_05_05_08.pdf

[4] http://deblaze-tool.appspot.com/

[5] http://www.gliffy.com/

[6] http://www.adobe.com/devnet/flex/articles/flex_java_architecture.html

[7] http://www.adobe.com/products/flash-builder.html

[8] https://www.owasp.org/index.php/Category:Vulnerability

[9] https://files.pbworks.com/download/OqCOsIJs9i/webappsec/13247059/WASC-TC-v2_0.pdf

[10] http://www.pyamf.org/

[11] http://pypi.python.org/pypi/SOAPpy/

[12] http://www.ivizsecurity.com/blog/web-application-security/testing-flash-applications-pen-tester-guide/

[13] http://www.gdssecurity.com/l/OWASP_NYNJMetro_Pentesting_Flex.pdf

[14] http://www.portswigger.net/

[15] http://www.sqlite.org/