講義資料（第二部）

セキュリティTIPSうすしお味
不正アクセス対応 RDP編

～実践的なインシデント対応と調査技術活用の一例～

2019年9月28日

仙台CTF推進プロジェクト

講師自己紹介

名前	五十嵐　良一（いがらし　よしかず）
職業	会社員
趣味	・フォレンジック技術の検証 ・マルウェアの解析

情報セキュリティ担当者のための実験室 セクタンラボ 管理人
http://www.sectanlab.jp/

いきなり体験！ インシデント対応

ストーリー

セキュリティ侵害の原因特定

- **イベントログ解析の結果、開発用サーバは攻撃者（IPアドレス192.168.15.171と10）にRDPログオンされていたことが判明しました。**
- **攻撃者は、Windows標準コマンドである「BitsAdmin」で不審ファイルをダウンロードしたようです。**

◆イベントログからの状況推測

年月日	時刻 (JST)	攻撃者の行動
2019年 8月25日 （日）	① RDPブルートフォース攻撃（1分23秒）	
	22:05:16 - 13:06:14 （58秒）	・開発用サーバへのRDPブルートフォース攻撃（43回、失敗） ・攻撃元IPアドレス 192.168.15.150～171（22個）
	22:06:15 （1秒）	・攻撃元IPアドレス192.168.15.171がRDPログオン成功 ・ログオン直後にログオフ
	22:06:17 - 13:06:39 （22秒）	・開発用サーバへのRDPブルートフォース攻撃（18回、失敗） ・攻撃元IPアドレス 192.168.15.172～180（9個）
	② 攻撃者によるRDPログオン操作（19分31秒）	
	22:20:20	・IPアドレス192.168.15.10によるRDPログオン
	22:23:06	・BitsAdminコマンドによるファイルダウンロード http://c2.attacker.invalid/tools.cab
	22:39:51	・調査者がログオンしたことで強制ログオフ

（参考）BITSAdminコマンドによるファイルダウンロード

名称　：**BITSAdmin.exe**

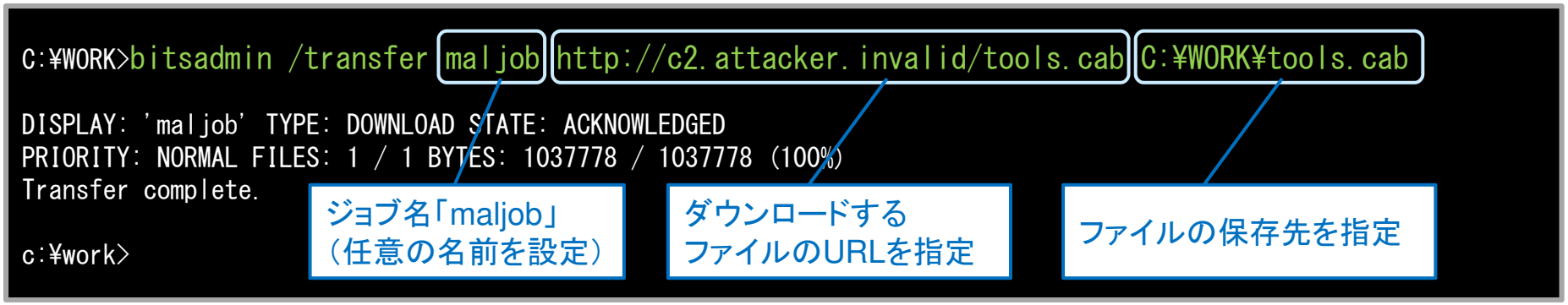
開発元：**Microsoft**

概要　：**Windows Vista以降に標準搭載されたファイルダウンロード/アップロード用ツール。**

［コマンド書式］

bitsadmin /transfer「ジョブ名」「ダウンロードするファイルのURL」「保存先ファイル名」

◆実行例

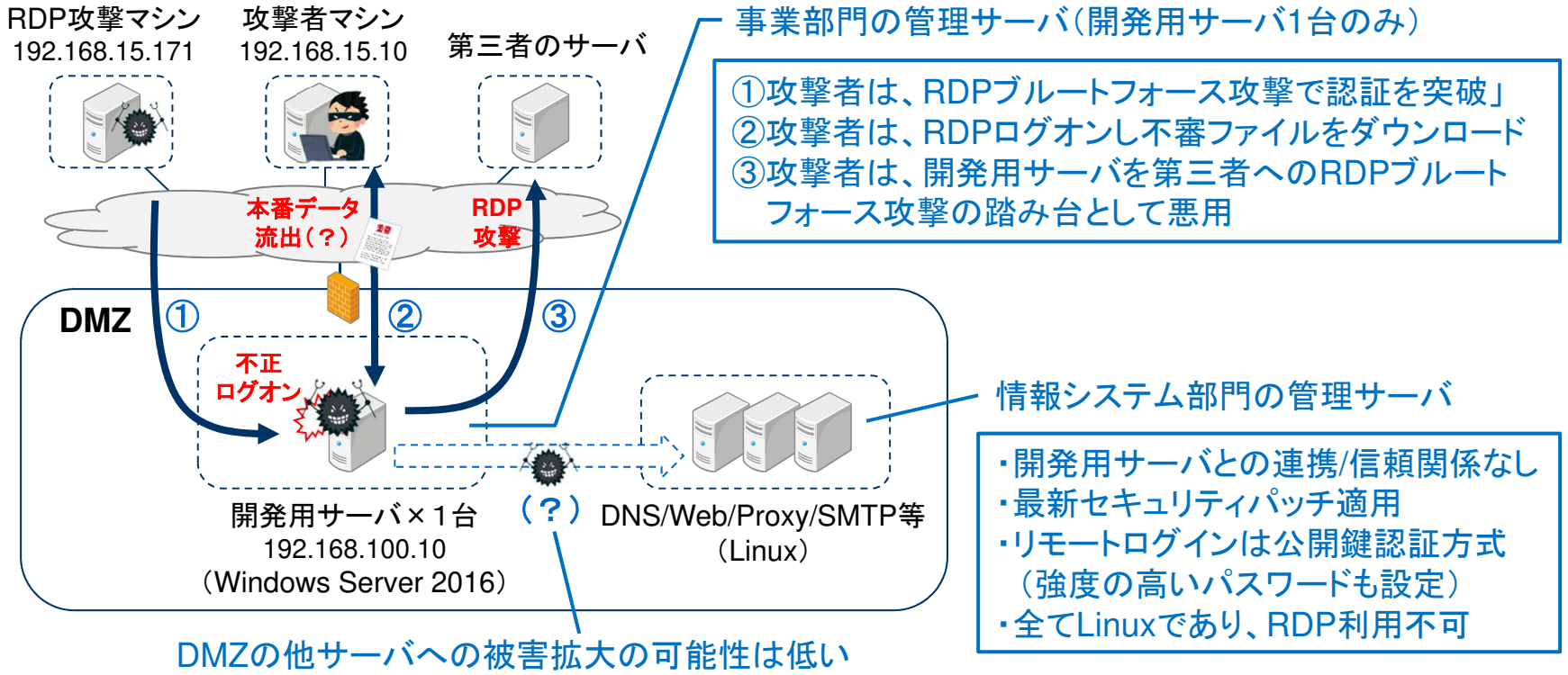


ストーリー

状況整理

- これまでの調査結果から状況を整理します。
  - 開発用サーバは、推測可能なパスワードを設定していたため、攻撃者にRDPログオンされ、第三者へのRDPブルートフォース攻撃の踏み台として悪用された。
  - 開発用サーバには本番データが格納されており、情報流出が懸念される。
  - なお、攻撃の状況から、DMZの他サーバへの被害拡大の可能性は低い。

◆ 状況推測



ストーリー

対策本部への報告準備

- **徹夜で対応にあたり、朝6時となりました。**
- **本日9時から対策本部会議が開催されるため、役員向けの報告資料を整理します。**

分類	報告・提案の例
1.発生した事象 （事実・推測）	• 事業部門の開発用サーバ（1台）が、社員による不適切な設定変更により不正アクセスされ、第三者へのブルートフォース攻撃の踏み台として悪用された。 • SNSには自社から攻撃を受けたとの投稿もされている。SNSの状況を監視しているが今のところ炎上はしていない。 • 開発用サーバには本番データ（顧客情報）が格納されており、不正アクセスにより情報流出が発生した可能性がある。
2.経営への影響 （想定リスク）	**[ただちに顕在化する可能性があるリスク]** • SNSが炎上する。（信用失墜） • 自社からの攻撃が原因で、第三者が不正アクセスを受け被害が発生する。（信用失墜、第三者からの損害賠償請求） **[顧客情報が流出していた場合のリスク]** • 顧客が詐欺被害に遭う。（信用失墜、顧客からの損害賠償請求） • 個人情報保護法違反として、国から是正勧告・改善命令を受ける。
3.官庁対応（案）	• 監督官庁に第一報を報告。同時並行で警察にも相談する。
4.報道対応（案）	• SNSに投稿されたことも踏まえ、社会的説明責任を果たすこと、踏み台攻撃を受けた第三者に注意喚起することを目的にプレス発表の準備を進める。
5.調査予定	• フォレンジック調査を実施し、本番データの情報流出有無を確認する。

ストーリー

対策本部会議

- 対策本部会議で、本日17時にプレス発表することが決定されました。
- 役員からは、プレス発表にあたり、情報流出の可能性について可能な範囲で調査するようにとの指示がありました。

- 情報システム部門で開発用サーバのフォレンジック調査を実施し、15時に中間報告することになりました。

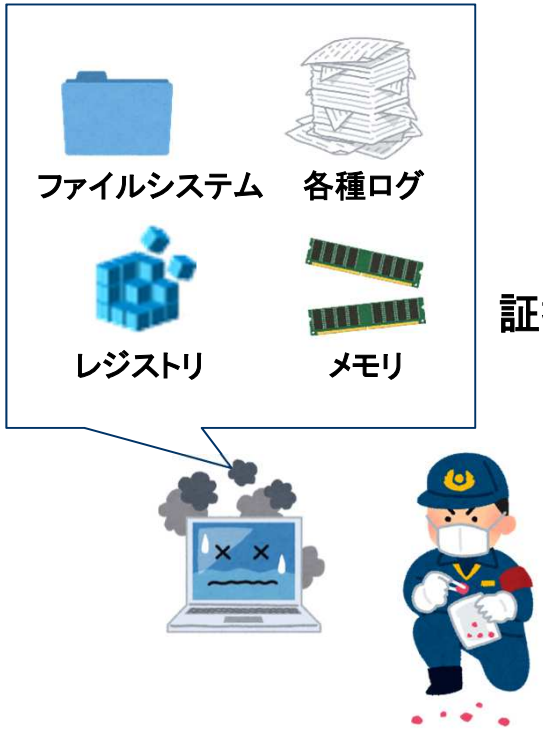
講義

状況把握に役立つ技術「フォレンジック」

- フォレンジック（Forensics）とは、インシデントが発生したコンピュータの解析を行い、「いつ」、「何が起きたのか」を調査する科学捜査手法のことです。
- サイバー攻撃の状況は目に見えづらいですが、フォレンジック技術を活用することで、より正確な状況推測ができるようになります。

◆ フォレンジックのイメージ

解析対象（エビデンス）



証拠保全・解析

解析結果（タイムライン解析）

いつ	何が
○月○日 12:30:50	PC-Aが改ざんされたウェブサイト「http://○○.com」にアクセス
12:30:55	リダイレクトにより、PC-Aが不審サイト「http://□□.ru」にアクセス
12:31:10	Adobe Reader への脆弱性攻撃により、PC-Aで不審プログラム「a.exe」が起動
12:31:12	PC-Aが「a.exe」が「http://△△.cn」との通信を開始
12:32:30	PC-Aから社内サーバに感染が拡大
12:35:00	IDSが、PC-Aの不審通信を検知

本事案の調査方針

- **プレス発表に向けて、情報流出の可能性について優先的に調査します。**

◆ **前提条件**

- 開発用サーバは、LANケーブル抜線後にシャットダウンされている。
- 開発用サーバのハードディスクを抜き出して必要なデータを保全済み。
  現在ハードディスクは複製装置でコピー中。
- 格納されている本番データ（顧客情報）は「C:¥work¥sendaictf.csv」。

◆ **調査方針**

（1）時間的制約もあることから、保全済みのデータに対して解析作業を実施する。
（ハードディスク全領域に対する解析は実施しない。）

（2）解析により以下2点の痕跡が発見された場合は「情報流出の可能性あり」と判断し、外部のセキュリティ専門家に解析を依頼する。

① 機密データファイル（sendaictf.csv）へのアクセス

② 情報流出につながるプログラムの実行
（情報流出機能のあるマルウェアやFTPなど）

ストーリー

調査手順の概要

- **開発用サーバから取得したエビデンスを解析し、攻撃者が開発用サーバにログオンしていた時間帯「2019年8月25日（日） 22:20:20～22:39:51」の挙動を確認します。**

◆エビデンスと解析ツール

分類	解析の概要	エビデンス	解析ツール
機密データファイルへのアクセス	① MFTのタイムライン解析 ファイルのアクセス状況を確認する。	Master File Table（$MFT）	•MFTECmd •mactime.pl •Timeline Explorer
	② NTFSのログ解析 ファイルのアクセス状況を確認する。	NTFSのログ（$LogFile、$J）	•NTFS Log Tracker
	③ レジストリの解析 本番データのファイル名などが記録されていないか確認する。	レジストリ（system、NTUSER.DAT、UsrClass.dat）	•Registry Explorer
情報流出につながるプログラムの実行※1	④ プログラム実行痕跡の解析 Application Compatibility Cache、UserAssistに記録されたプログラム実行痕跡を確認する。	レジストリ（system、NTUSER.DAT）	•Registry Explorer

（※1） プログラム実行痕跡を確認できるアーティファクトとして「Prefetch」もありますが、Windows Serverの標準設定では無効化されています。

講義

① MFTのタイムライン解析（1）

- **MFTからファイル・フォルダのタイムラインを作成し、攻撃者がログオンした時間帯のファイルのアクセス状況を確認します。**

手順

① 「$MFT」を「mftecmd」コマンドで前処理し、「body」形式の中間ファイルを作成する。
② 「mactime」コマンドで「body」ファイルを整形し、タイムラインを作成する。

コマンド書式

① mftecmd -f 【*$MFTのファイル名*】 --body 【*bodyの出力先フォルダ名*】※1 --bdl 【*ドライブ名*】※2
（※1）指定したフォルダに、ファイル名「YYYYMMDDhhmmss_MFTECmd_Output.body」で出力される。
（※2）ドライブレターとして表示したい任意の文字列を指定する。（例：C）

② mactime -b 【*bodyファイル名*】 -z Japan -m -d > 【*タイムラインの出力ファイル名*】
（①で出力したファイル）

① MFTのタイムライン解析(2)

◆実行例 ① mftecmd

```
caine@caine:$ mftecmd -f $MFT --body . --bdl C
MFTECmd version 0.2.9.1

Author: Eri
https://github.com/EricZimmerman/MFTECmd

Command line: -f $MFT --body . --bdl C

003a:err:winsock:WSAIoctl -> SIO_ADDRESS_LIST_CHANGE request failed with status 0x2733
003a:err:winsock:WSAIoctl -> SIO_ADDRESS_LIST_CHANGE request failed with status 0x2733

Processed '$MFT' in 6.8083 seconds

Bodyfile output will be saved to '.¥20190916063445_MFTECmd_Output.body'
003d:err:winsock:WSAIoctl -> SIO_ADDRESS_LIST_CHANGE request failed with status 0x2733
003d:err:winsock:WSAIoctl -> SIO_ADDRESS_LIST_CHANGE request failed with status 0x2733
003f:err:winediag:SECUR32_initNTLMSP
is in your path. Usually, you can f
caine@caine:$
```

MFTのファイル名「$MFT」を指定

出力先フォルダとして、カレントディレクトリを意味する「.」(ドット)を指定

タイムラインに表示するドライブレターとして「C:」を指定

ファイル名「20190916063445_MFTECmd_Output.body」でbodyファイルが出力された。

① MFTのタイムライン解析(3)

◆実行例 ② mactime

```
caine@caine:$ mactime -b 20190916063445_MFTECmd_Output.body -z Japan -m -d > timeline_mft.txt
caine@caine:$
```

前頁の手順で作成したbodyファイル名

ファイル名「timeline_mft.txt」で
タイムラインを出力

◆タイムライン「timeline_mft.txt」の内容例(抜粋)

```
Date,Size,Type,Mode,UID,GID,Meta,File Name
Sun 08 25 2019 21:36:12 0,mac.,r/rrwxrwxrwx,0,0,109230-144-0,"c:/work"
Sun 08 25 2019 21:36:12,279,.a.b,r/rrwxrwxrwx,0,0,109238-128-1,"c:/work/sendaictf.csv"
S 279 8-2, NAME)"
S 204 -128 eHash{06AE3BCC-7612-39D3-9F3B-
    B6601D877D02}"
Sun 08 25 2019 21:36:12,20480,ma.b,r/rrwxrwxrwx,0,0,109326-128-4,"c:/Windows/Installer/Sour 7612-39D3-9F3B-
    B6601D877D02}"
Sun 08 25 2019 21:36:12,20480,macb,r/rrwxrwxrwx,0,0,109326-48-2,"c:/Windows/Installer/So
    B6601D877D02} ($FILE_NAME)"
```

時刻情報(JST)

タイムスタンプの種類※1

ファイル・フォルダ名※2

解析結果

本番データへのファイルアクセスの痕跡は確認されなかった。

(※1)ファイル・フォルダには、更新日時、作成日時など、複数のタイムスタンプが記録されている。タイムラインでは、同じ時刻のタイムスタンプを一行で表現している。
m:更新日時、a:アクセス日時、c:属性変更日時、b:作成日時

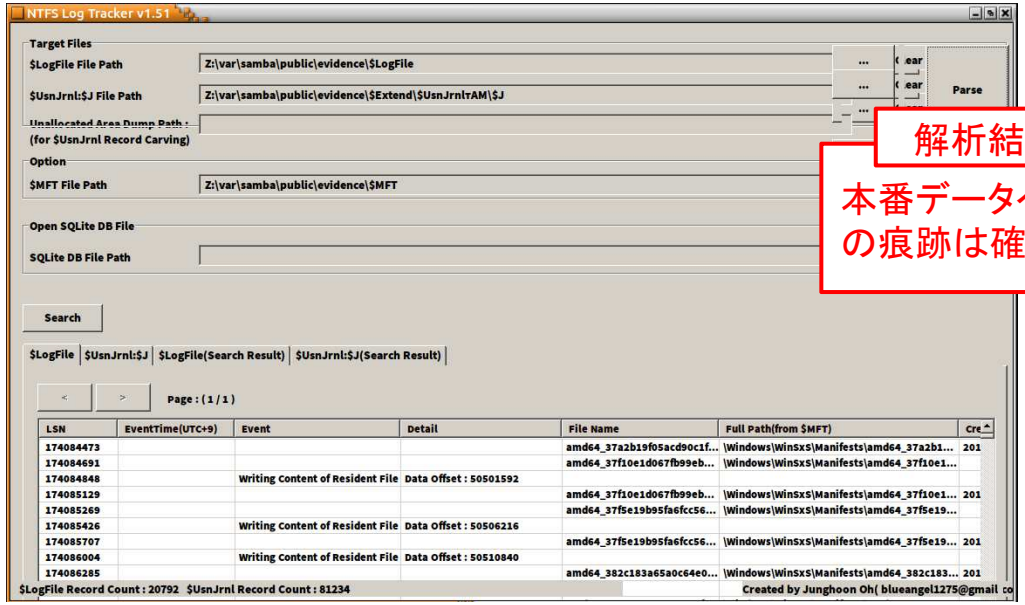
(※2)削除済みファイルは(deleted)が付記される。また、NTFSの「Filename属性」のタイムスタンプは「$FILE_NAME」が付記される。

② NTFSのログ解析

- **NTFSのログファイル「$LogFile」、「$J」を解析し、攻撃者がログオンした時間帯のファイルのアクセス状況を確認します。**

手順

① CAINEの「Main Menu」-「Windows Forensic Tools」から「NTFS Log Tracker」を起動する。
② エビデンスの「$LogFile」、「$J」、「$MFT」を指定し「Parse」ボタンをクリックする。
③ 「Parse Setting」ダイアログが表示されたら「SQLite DB File Name」と「SQLite DB File Path」に任意の名前・ディレクトリを入力し「Start」をクリックする。（解析結果格納用DBが新規作成される。）
④ 解析結果が画面に表示される。「CSV Export」ボタンをクリックすると、CSVで保存できる。

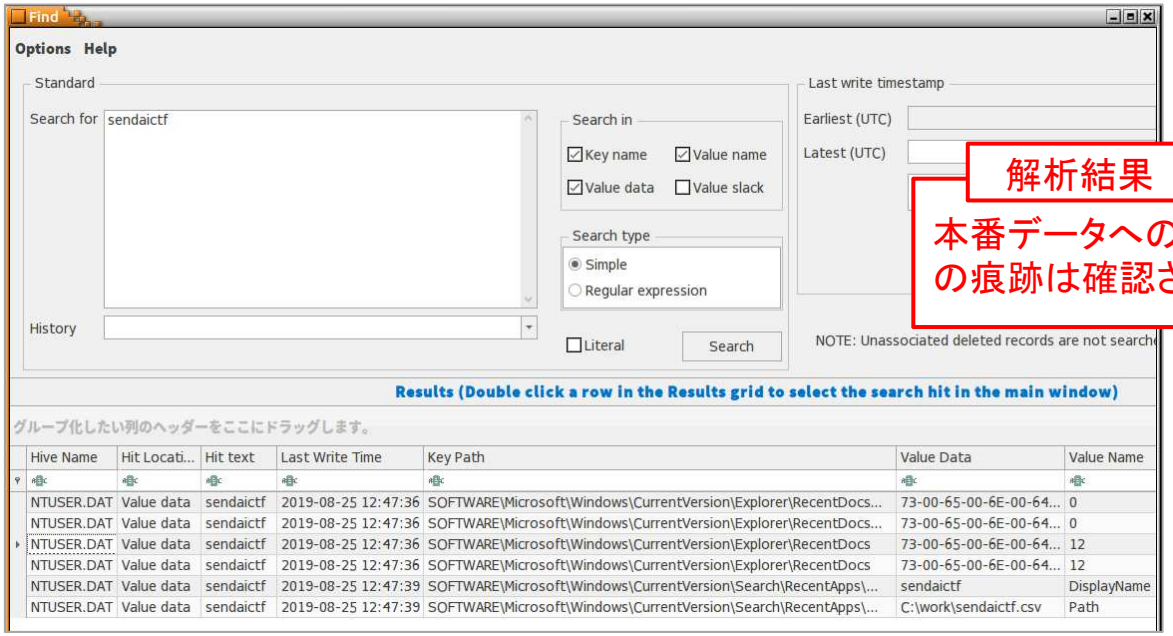


③ レジストリの解析

- レジストリをキーワード検索し、攻撃者がログオンした時間帯に本番データにアクセスした痕跡がないか確認します。

手順

① CAINEの「Main Menu」-「Windows Forensic Tools」から「Registry Explorer」を起動する。
② エビデンスの「system」、「NTUSER.DAT」、「UsrClass.dat」を開く。
③ メニュー「Tools」-「Find」をクリックし、検索ダイアログで本番データのファイル名「sendaictf」を検索する。



解析結果

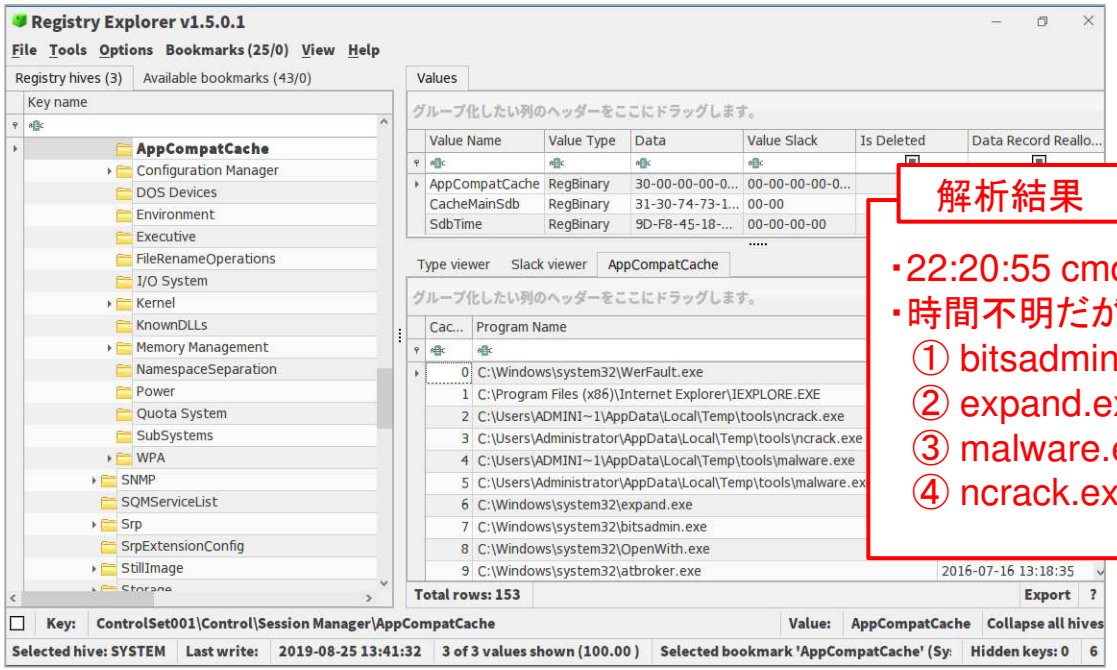
本番データへのファイルアクセスの痕跡は確認されなかった。

④ プログラム実行痕跡の確認

- レジストリのプログラム実行痕跡を解析し、攻撃者がログオンした時間帯に起動されたプログラムを確認します。

手順

① CAINEの「Main Menu」-「Windows Forensic Tools」から「Registry Explorer」を起動する。
② エビデンスの「system」、「NTUSER.DAT」を開く。
③ レジストリ「system」を選択し、メニュー「Bookmarks」から「AppCompatCache」を選択し、プログラム実行痕跡を確認する。同様にレジストリ「NTUSER.DAT」の「UserAssist」も確認する。

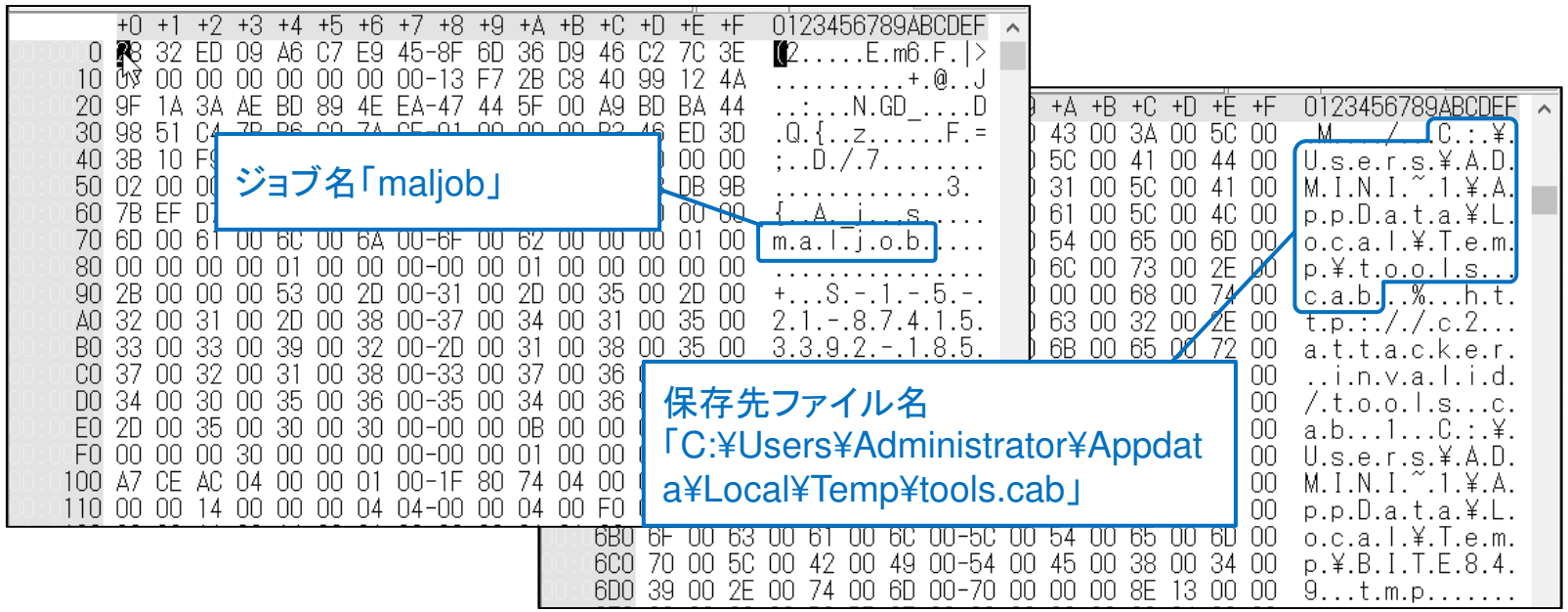


講義

（参考）BITSAdminによるダウンロード痕跡

- イベントログ「Microsoft-Windows-Bits-ClientのイベントID 59にBITSAdminの「ジョブ名」と「ダウンロードするファイルのURL」が記録されます。
- 保存先ファイル名は、Queue Manager Database（QMGR）を解析することで特定できます。
  - QMGR Database
    - フォルダ ：C:¥ProgramData¥Microsoft¥Network¥Downloader¥
    - ファイル：qmgr0.dat、qmgr1.dat

◆QMGR Databaseの内容例



（参考）QMGR Database解析ツール

名称　　：bits_parser

開発元　：ANSSI（Agence nationale de la sécurité des systèmes d'information）
https://github.com/ANSSI-FR/bits_parser

概要　　：QMGR Database解析用Pythonスクリプト。

［コマンド書式］
　bits_parser.py -o「解析結果のファイル名」「QMGR Databaseファイル」

◆実行例

```
caine@caine:$ bits_parser -o qmgr0.csv qmgr0.dat
caine@caine:$
```

解析結果を「qmgr0.csv」というファイル名で出力

解析対象のファイル名

◆解析結果の内容例（CSV形式）

```
job_id,name,desc,type,priority,sid,state,cmd,args,file_count,file_id,dest_fn,src_fn,tmp_fn,download_size,transfer_size,drive,v
ol_guid,ctime,mtime,other_time0,other_time1,other_time2,carved

,,,,,,,,,|>,1,0,C:¥Users¥ADMINI~1¥AppData¥Local¥Temp¥tools.cab,http://c2.attacker.invalid/tools.cab,C:¥Users¥ADMINI~1¥AppData¥L
ocal¥Temp¥BITE849.tmp,0,,C:¥,¥¥?¥Volume{0c398570-0000-0000-0000-501f00000000}¥,2019-08-25 13:23:05.721552,2019-08-25
13:23:05.742742,2019-08-25 13:23:05.742742,2019-08-25 13:23:05.742742,2019-11-23 13:23:05.742742,True
```

BITSAdminでダウンロードした保存先ファイル名

ストーリー

調査結果

- 攻撃者は、開発用サーバに不正ログオンした後、C2サーバからダウンロードした不審プログラムを実行し、第三者にRDPブルートフォース攻撃を実施していました。
- 本番データ（顧客情報）にアクセスした形跡は確認されませんでした。

◆攻撃者の行動のタイムライン[2019年 8月25日（日）]

時刻(JST)	行動	エビデンス
22:20:20	・IPアドレス192.168.15.10から開発用サーバにRDPログオン。	イベントログ
22:20:55	・cmd.exe起動	NTUSER.DAT
22:23:06	・BitsAdminコマンドによるファイルダウンロード。 http://c2.attacker.invalid/tools.cab → C:¥Users¥Administrator¥AppData¥Local¥Temp¥tools.cab	イベントログ、$MFT、QMGR.DAT
22:26:40	・tools.cabをexpandコマンドで展開。	$MFT、AppCompatCache
22:27:10	・IPアドレスが列挙されたテキストファイルを作成（ダウンロード？）。 C:¥Users¥Administrator¥AppData¥Local¥Temp¥targets	$MFT
不明	・不審プログラム実行（2個のプログラムを2回）。 ？？？ C:¥Users¥Administrator¥AppData¥Local¥Temp¥tools¥malware.exe C:¥Users¥Administrator¥AppData¥Local¥Temp¥tools¥ncrack.exe ・ncrackはRDPブルートフォース攻撃機能を有するハッキングツールであり、本ツールが第三者への攻撃に使用された可能性が高い。	AppCompatCache
22:39:51	・調査者がログオンしたことで強制ログオフ。	イベントログ

ストーリー

対策本部会議での判断

- 予定どおり15時に、対策本部会議でフォレンジック調査結果を報告しました。

- 調査結果を踏まえ、プレス発表では「開発用サーバにお客様の個人情報が格納されていたものの、フォレンジック調査により情報流出の痕跡が無いことを確認した。」と宣言することになりました。

ストーリー

対応終了！

- 情報システム部門の活躍により、インシデントを早期に収束することができました。
- あなたは徹夜で対応したためヘトヘトです。自宅に戻りゆっくりと休むことにしました。

ストーリー

・・・対応終了？

はい
いいえ

よくぞ来た！ わしが 上司のなかの上司　城氏である。
わしは 待っておった。 そなたのような若者があらわれることを。

どうじゃ？　もう帰って寝るか？

ストーリー

そうだ、マルウェアを解析してみよう

はい
いいえ

よくぞ来た！ わしが 上司のなかの上司　城氏である。
わしは 待っておった。 そなたのような若者があらわれることを。

どうじゃ？　もう帰って寝るか？

TIPS-3 マルウェア解析

マルウェアのコードを逆アセンブルし挙動を調査する
「静的解析」を体験します。

講義

マルウェア解析の種類

- マルウェア解析の手法は、表層解析、動的解析、静的解析の3種類があり、各手法を組み合わせてマルウェアの挙動を調査します。

◆マルウェア解析の種類

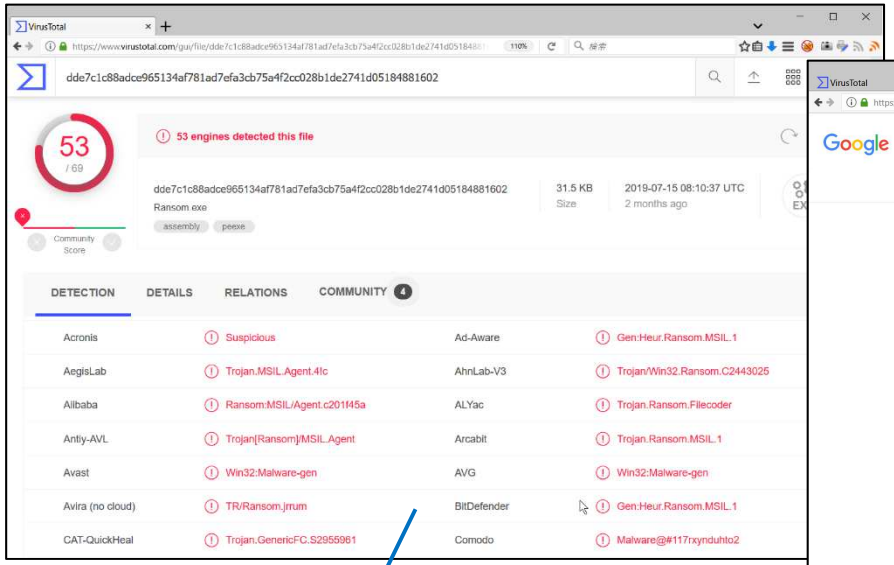
解析手法	概要	メリット・デメリット
表層解析	マルウェアのファイル名、ハッシュ値、ファイル内の文字列など、表層的な特徴をインターネット検索などで調査する手法。	○短時間で簡単に解析できる。 ×必要な情報が得られないことがある。
動的解析	本番環境から隔離された仮想環境などでマルウェアを動作させて、ファイルアクセスや通信などの挙動を観察する手法。	○比較的短時間で多くの情報を得ることができる。 ×マルウェアが解析妨害機能を有する場合、必要な情報を得られないことがある。
静的解析 本勉強会で体験	マルウェアのコードを逆アセンブルし、詳細な挙動を確認する手法。	○特定条件下で実行される処理など、詳細な情報を得ることができる。 ×解析に必要な技術レベルが高く、時間もかかる。

表層解析の一例

- 不審プログラムのハッシュ値をVirus Total、Googleなどで検索してみます。
  SHA-256：dde7c1c88adce965134af781ad7efa3cb75a4f2cc028b1de2741d05184881602

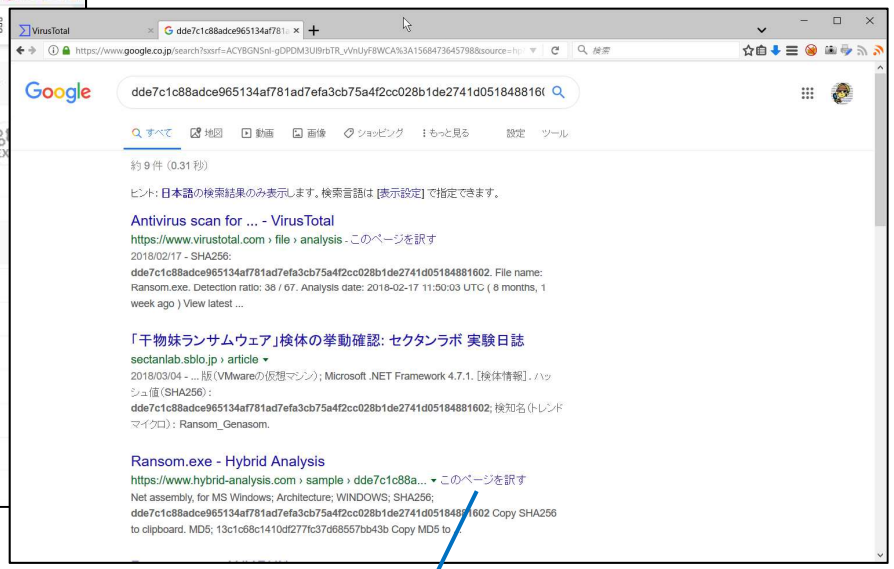
◆Virus Totalでのハッシュ値検索結果

https://www.virustotal.com/



多数のセキュリティ対策ソフトで
「ランサムウェア」として検知しており、
マルウェアである可能性が高い。

◆Googleでのハッシュ値検索結果



解析結果が掲載されたブログによると、
脅迫文が表示されず、暗号化されたファイルを
容易に復号化することができるため、ジョーク
ソフトに近いマルウェアと思われる。

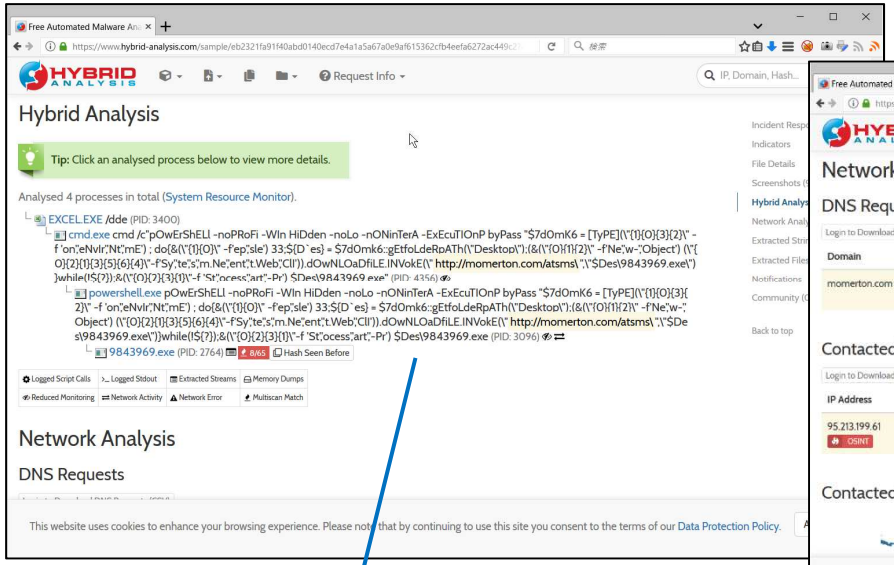
動的解析の一例

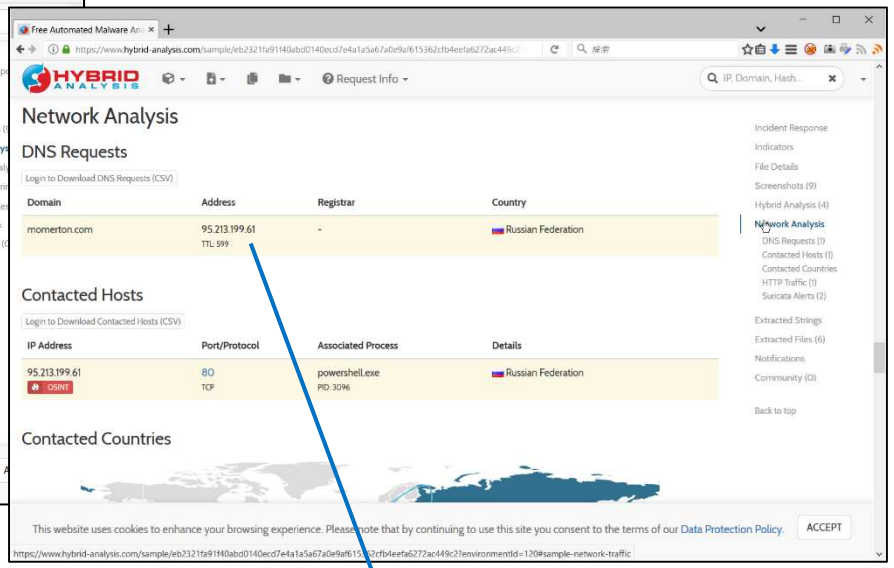
- **不審エクセルファイル「 (73).201805請求データ.xls」を、動的解析クラウドサービスで解析してみます。**

  SHA-256： eb2321fa91f40abd0140ecd7e4a1a5a67a0e9af615362cfb4eefa6272ac449c2

◆Hybrid Analysisでの解析結果

https://www.hybrid-analysis.com/





不審エクセルファイルを開くと、cmd.exeから
PowerShellが起動され、不審URLから、
マルウェアをダウンロードし実行することが判明。

動的解析で判明した通信先IPアドレスは、ファイアウォールなどで遮断するとともに、社内からのアクセス有無（＝不審ファイル開封有無）をプロキシログ調査を実施。

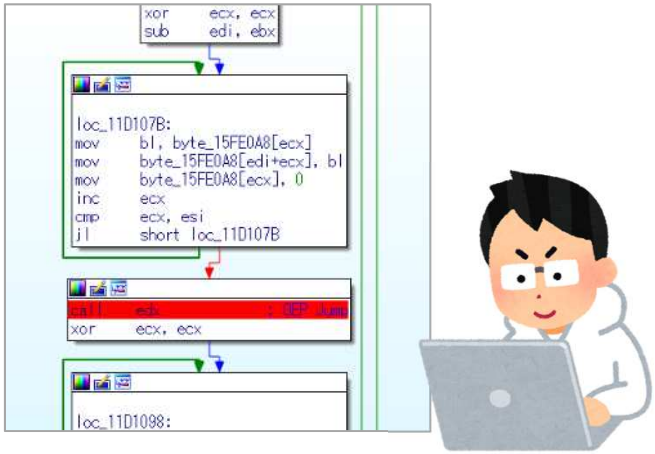
講義

静的解析は難しい？

- 逆アセンブルリストを読み解く必要があるため、難しいというイメージがありますが、「大まかな挙動を、なんとなく見てみる」レベルであれば、初心者でも大丈夫です。
- 本勉強会では、学習のキッカケ作りのため、敬遠されがちな静的解析を「体験」します。

◆静的解析スキルの活用例（学習のメリット）

- メモリダンプから、マルウェアの挙動を詳細に確認することもできる。
- 「この業務プログラムには、暗号化したうえでパスワードを埋め込んであります。」と言われた場合に「いや、ただのXORですよね。見えてますよ・・・。」と指摘できる。
- 静的解析ができるとカッコイイ！？ ← 重要



静的解析に必要な知識

- **C言語とWindows API、アセンブリ言語のちょっとした基礎知識があれば、静的解析の学習を始めることができます。**

C言語とWindows APIの基礎知識

・関数（printfなど）の呼び出し方、条件分岐（if文）
・いくつかのWindows APIの関数　など

アセンブリ言語の基礎知識

・いくつかの基本的な命令
・レジスタとスタックメモリ　など

これだけ理解すればOK!? C言語（1）

- C言語は、さまざまな「関数」を呼び出して処理を実行します。
- 関数名の英単語から動作を推測しましょう。
  - 最初から全てを理解しようとすると挫折します。理解できなくても気にしない！

◆C言語のソースコード

サンプル：hello.c

```
#include<stdio.h>
int main( void ) {
    printf( "Hello World!" );
    return 0;
}
```

起動時に自動的に呼び出しされる「main」関数に処理を記載。

print：印字する
「printf」関数を呼び出し。
関数の引数として、印字したい文字列「Hello World!」を指定。

コンパイル・実行結果

```
C:¥WORK>hello.exe
Hello World!
C:¥work>
```

これだけ理解すればOK!? C言語(2)

- 処理を条件分岐したい場合は「if - else」文などを使います。

◆C言語のソースコード

サンプル: **atari.c**

```
#include<stdio.h>
int main( void ) {
    int a = 2;
    if ( a == 7 ){
        printf("あたり！");
    }
    else {
        printf("はずれ！");
    }
    return 0;
}
```

if: もし～ならば
変数aの値が「7」ならば
「あたり！」を印字。

else: そうでなければ
(if に合致しなければ)
「はずれ！」を印字。

コンパイル・実行結果

```
C:¥WORK>atari.exe
はずれ！
C:¥work>
```

これだけ理解すればOK!? C言語(3)

- 本勉強会の実習では、下表の関数が登場します。
- 英単語から動作を推測し、詳細を知りたい場合はインターネットで調べてください。

◆本勉強会に登場する関数の一例

関数名	概要
printf	• 文字列を指定した書式で印字する。 • 関数名末尾の「f」は、書式を意味する「format」
fopen、fread、fclose	• ファイルの読み込みを行う。 • fopenでファイルをオープン状態にした後、freadで読み込みする。処理終了後はfcloseでファイルをクローズする。
WSAStartup、socket、connect、send、closesocket、WSACleanup	• ネットワーク接続しデータを送信する。 • connectで指定したIPアドレスに接続し、sendでデータを送信する。その他の関数は前処理・事後処理で使用する。
gethostbyname	• ホスト名の名前解決を行い、IPアドレスを得る。
htons	• ポート番号の前処理に使う。 （ホストバイトオーダーからネットワークバイトオーダーに変換する。）

これだけ理解すればOK!? Windows API

- **Windows APIは、ファイルの読み書きなど、Windows OSのさまざまな機能を利用するための関数のことです。**
- **本勉強会の実習では、下表のWindows APIが登場します。**
- **英単語から動作を推測し、詳細を知りたい場合はインターネットで調べてください。**

◆本勉強会に登場するWindows APIの一例

API	概要
URLDownloadToFile	• 指定したURLからファイルをダウンロードする。 • 引数としてURL、ダウンロードしたファイルを保存するパスを指定する。
GetLocalTime	• 現在の時刻（日本時間の年月日、時分秒）を取得する。 • 引数で指定した変数（構造体）に時刻情報が格納される。
WinExec	• 引数で指定したプログラムを起動する。
GetCurrentDirectory	• カレントディレクトリを取得する。

これだけ理解すればOK!? アセンブリ言語（1）

- C言語など、人間が理解しやすい「高級言語」で作成したプログラムも、最終的には「0」、「1」の機械語（マシン語）に変換されてから実行されます。
- アセンブリ言語は、機械語とほぼ1対1で対応した命令を記載する「低級言語」ですが、解析ツールを使えばプログラムの動作を推測することは決して難しくはありません。
  - ただし、「面倒くさい」、「時間がかかる」のは事実です・・・。

◆C言語とアセンブリ言語のコード

C言語

ソースコード
printf("Hello World!");
return 0;

アセンブリ言語（解析ツールによる表示例）

命令（オペコード）	パラメーター（オペランド）
PUSH	s_Hello_World!_0040c000
CALL	_printf
XOR	EAX,EAX
RET	

これだけ理解すればOK!? アセンブリ言語(2)

- **本勉強会の実習にあたり覚えたほうが良いアセンブリ言語の主な特徴や命令は下表のとおりです。**
  - ざっくりとイメージを掴んでいただくため、正確性を割り切った表現としています。

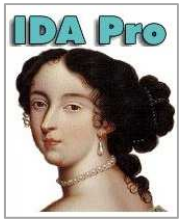
◆アセンブリ言語の主な特徴や命令

項目	説明	
関数 呼び出し	• CALL命令で関数を呼び出しする。 • 引数は、関数呼び出し前にPUSH命令でスタックメモリに格納する。 (引数と逆の順番でPUSHされるため、第一引数が最後にPUSHされる。)	
スタック、 レジスタ	• データなどを保存する領域のこと。 • スタックは、PUHS命令でデータを格納、POP命令でデータを取り出しする。PUSHで積み木のようにデータを積み上げ、POPで上から順にデータを取り出しする。 • レジスタは、CPUに設置された容量が小さいが高速に読み書きできるメモリ。EAX、EBP、ESPなどのレジスタがあり、用途に応じて利用される。	
命令	関数呼び出し	PUSH(引数設定)、CALL(関数呼び出し)、RET(return文) ※関数の戻り値はEAXレジスタに格納。 例: PUSH s_Hello_World_0040c000、 CALL printf
	IF文	CMP(比較命令)、JNZ(比較結果に応じたジャンプ命令) など 例: CMP ECX,7(レジスタECXを7と比較)、JNZ LAB_00401040(7以外ならジャンプ)
	変数代入	MOV(代入)、LEA(代入とほぼ同じ) など 例: MOV ECX, 7(レジスタECXに7を代入)

静的解析ツール

- 代表的な静的解析ツールとして「IDA Pro」、「Ghidra」などがあります。
- 本勉強会では「Ghidra」を利用します。

◆静的解析ツール

名称（開発元）	ライセンス	特徴
IDA Pro （Hex-ray社） 	有償※ ※機能制限された フリー版もある	○ 製品として成熟しており、高速・高機能。 ○ デバッガ機能がある。 × 高価。デコンパイル機能は別売。 × アンドゥ（取り消し）機能が無い！
Ghidra （NSA） 	オープンソース	○ 十分に高機能。 ○ デコンパイル機能がある。 ○ アンドゥ（取り消し）機能がある。 × 正しくコードを解析できない部分がある。

実習　Ghidraの操作練習

（別紙. TIPS-3実習資料 参照）

- 実習を通じて、Ghidraの操作方法を確認します。

実習1　ダウンロードファイルの特定

実習用プログラム「/var/samba/public/tips3/re01_urldownload.exe」を解析し、
① 不審プログラムがダウンロードしたファイル名（保存先のフルパス）、
② 上記①のダウンローと元URLを特定してください。

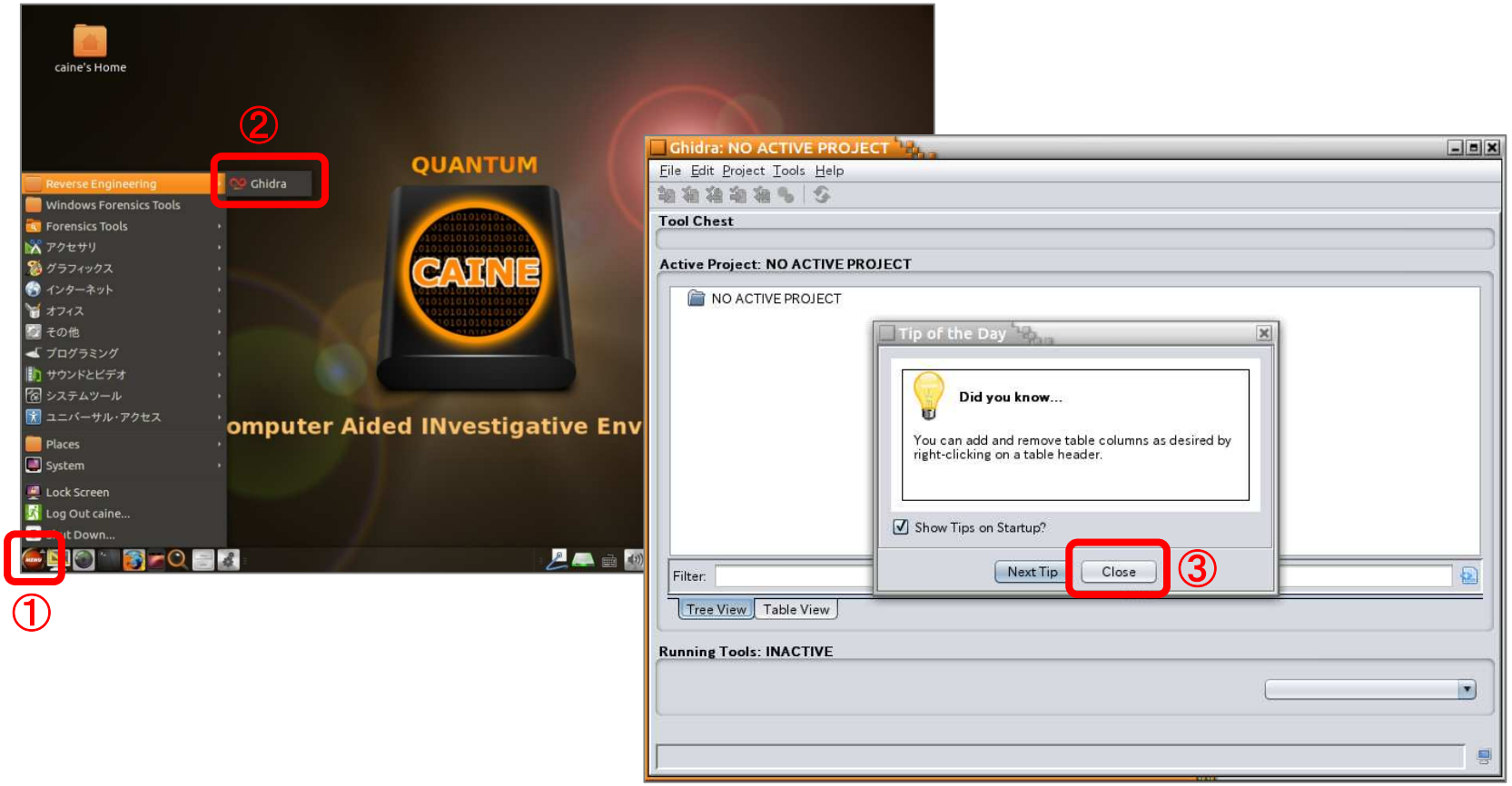
［ヒント］ 実習1のソースコード

```
#include <windows.h>
#include <stdio.h>
#pragma comment(lib, "urlmon")

void main()
{
    printf("Q1.What URL do I connect to?¥nQ2.What is the path of the downloaded file?¥n");
    URLDownloadToFile(NULL, "http://[redacted]", "C:¥¥[redacted]", 0, NULL);

    return;
}
```

実習1

Ghidraの操作（1）起動

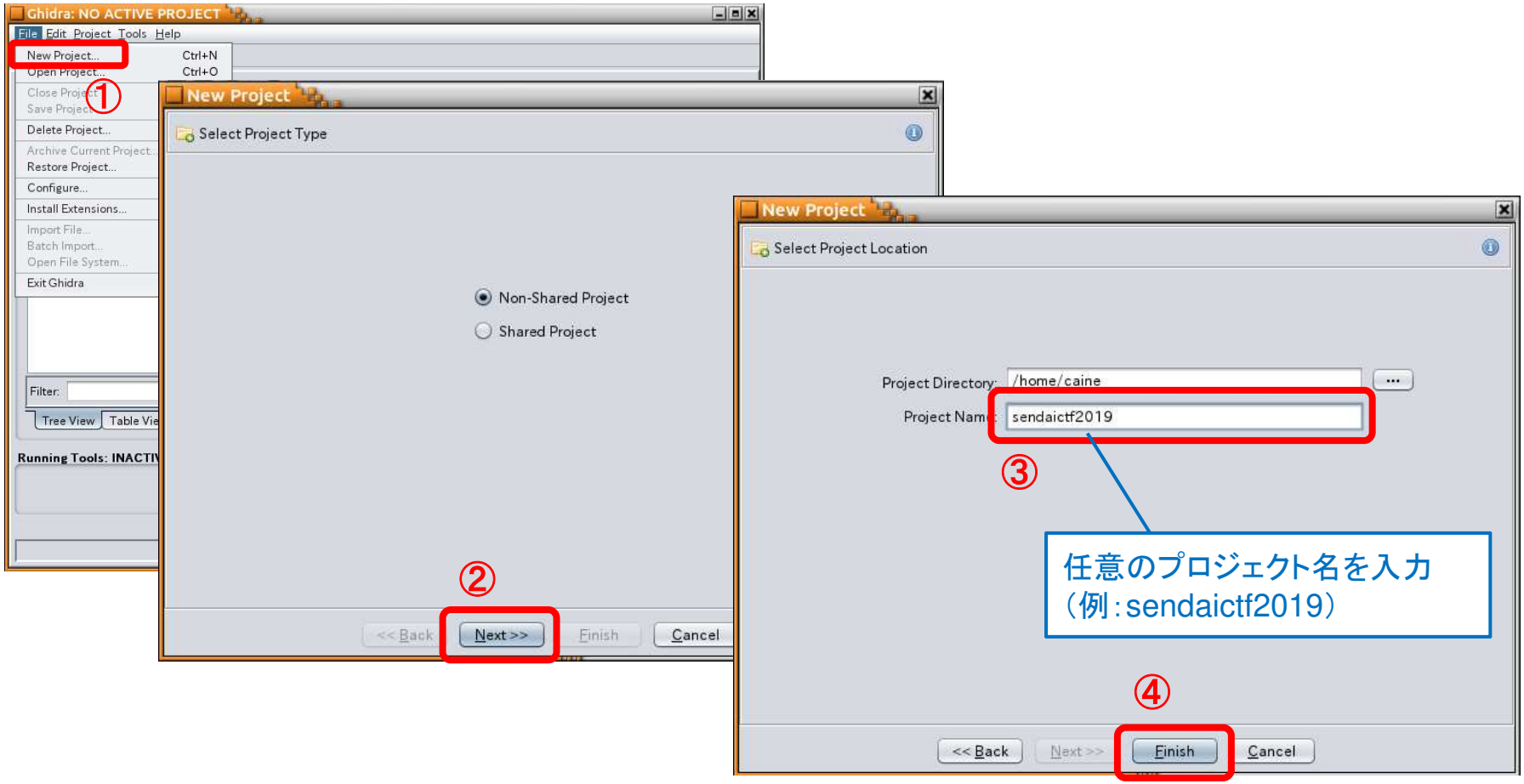
- 実習用仮想マシン「Caine」の「Main Menu」からGhidraを起動します。
- 「Tip of the Day」ダイアログは「Close」をクリックして閉じてください。



実習1

Ghidraの操作（2）プロジェクトの作成

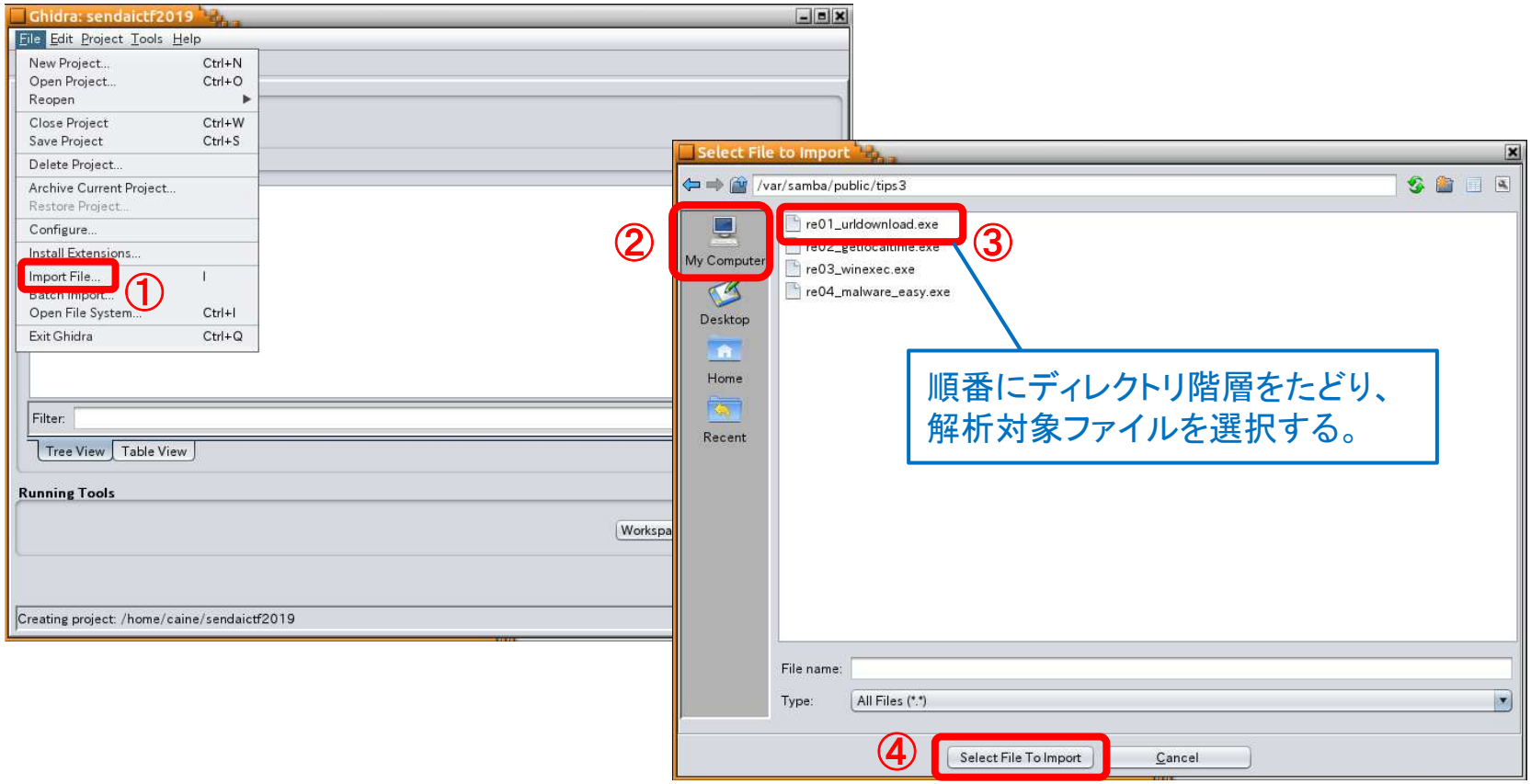
- メニュー「File」-「New Project」をクリックします。
- 「Non-Shared Project」が選択されていることを確認し「Next」をクリックします。
- 「Project Name」に任意のプロジェクト名を入力し「Finish」をクリックします。



実習1

Ghidraの操作（3）解析対象ファイルの追加

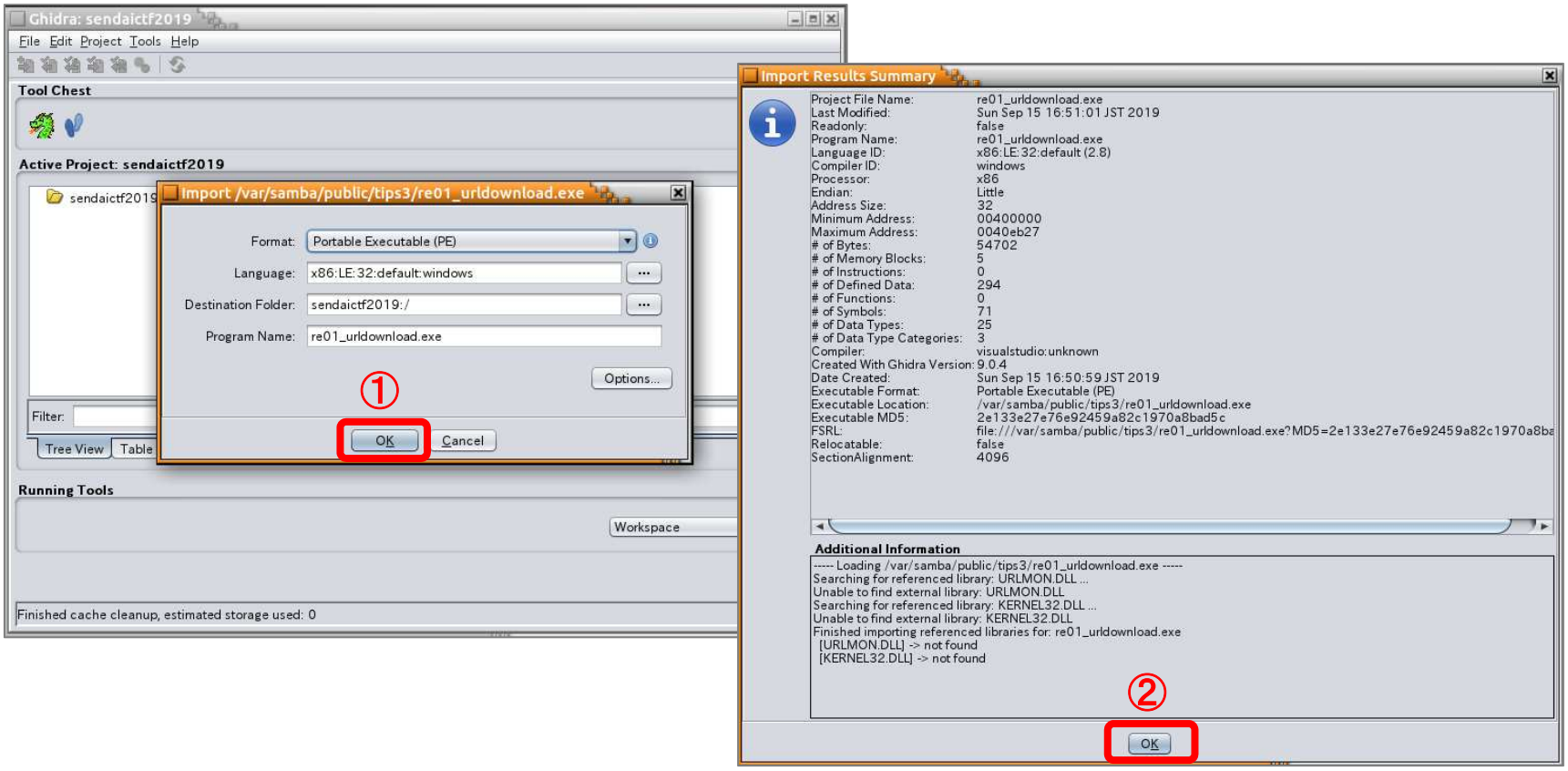
- メニュー「File」-「Import File」をクリックします。
- 「Select File to Import」ダイアログで「My Computer」をクリックし、ディレクトリ「/var/samba/public/tips3」にある「re01_urldownload.exe」を選択し「Select File To Import」をクリックします。



実習1

Ghidraの操作（4）解析対象ファイルの確認

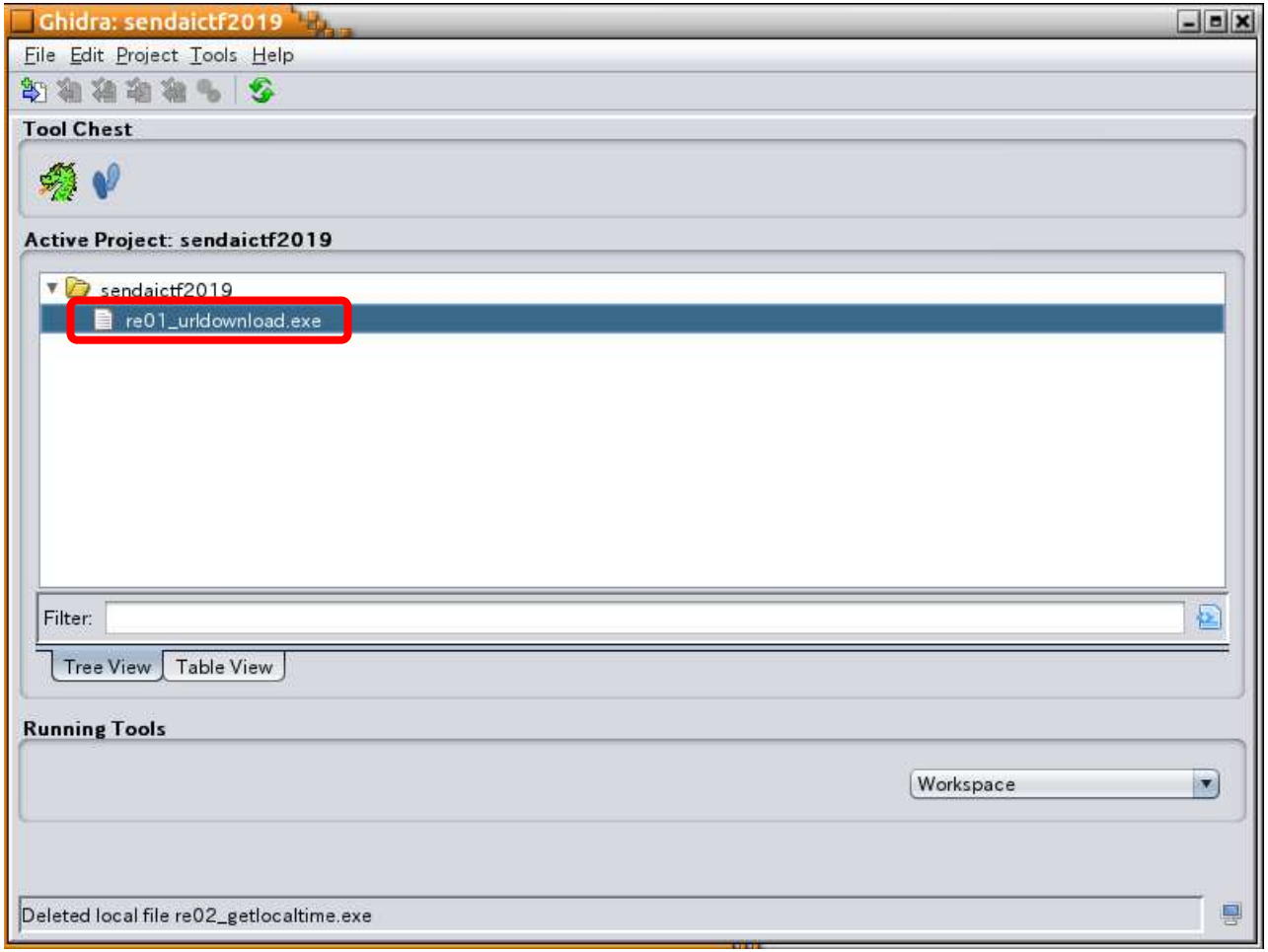
- 解析対象ファイルの確認ダイアログが2回表示されますが、そのまま「OK」をクリックします。



実習1

Ghidraの操作（5）プロジェクトウィンドウ

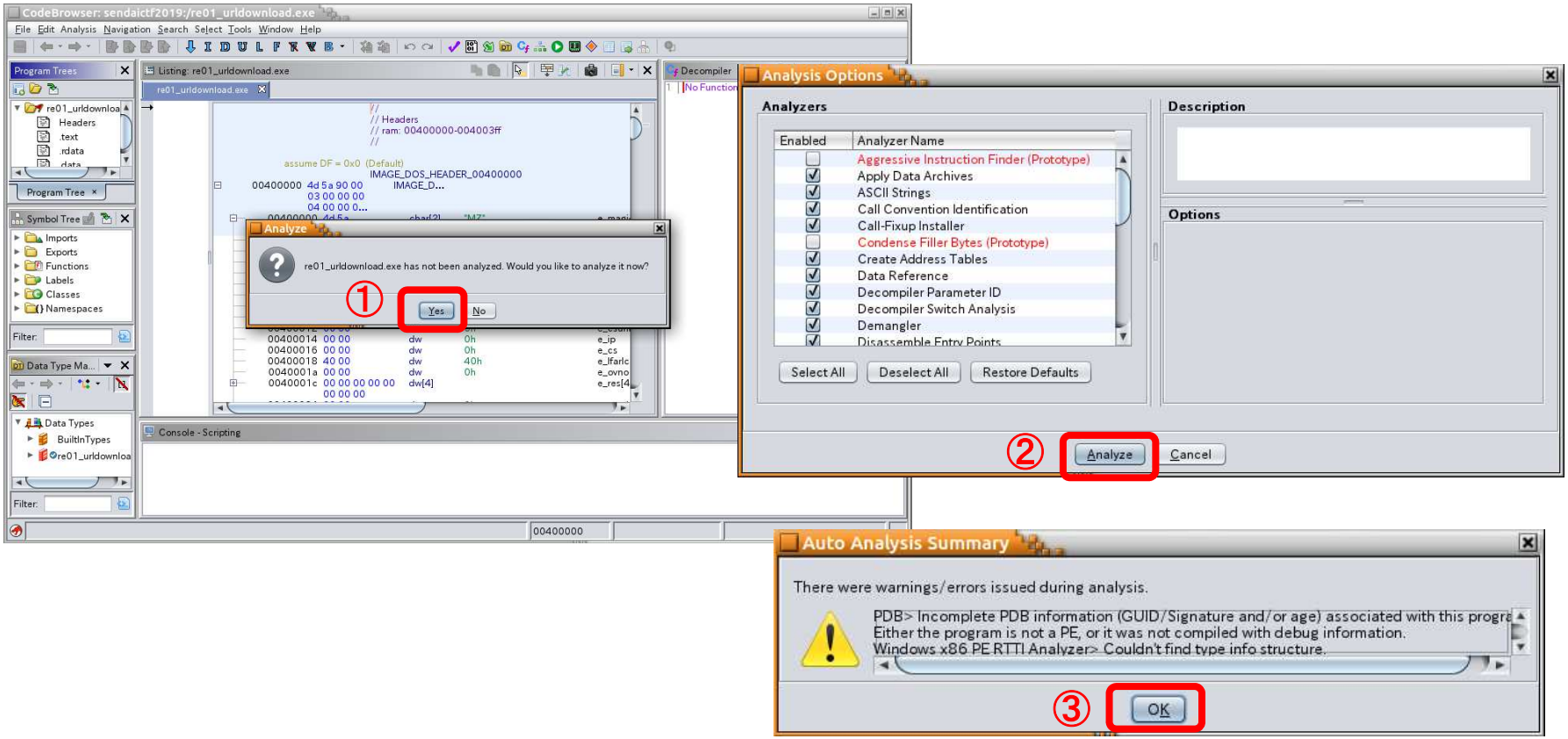
- プロジェクトウィンドウに解析対象ファイルが追加されました。
- 解析対象ファイルをダブルクリックします。



実習1

Ghidraの操作（6）自動解析

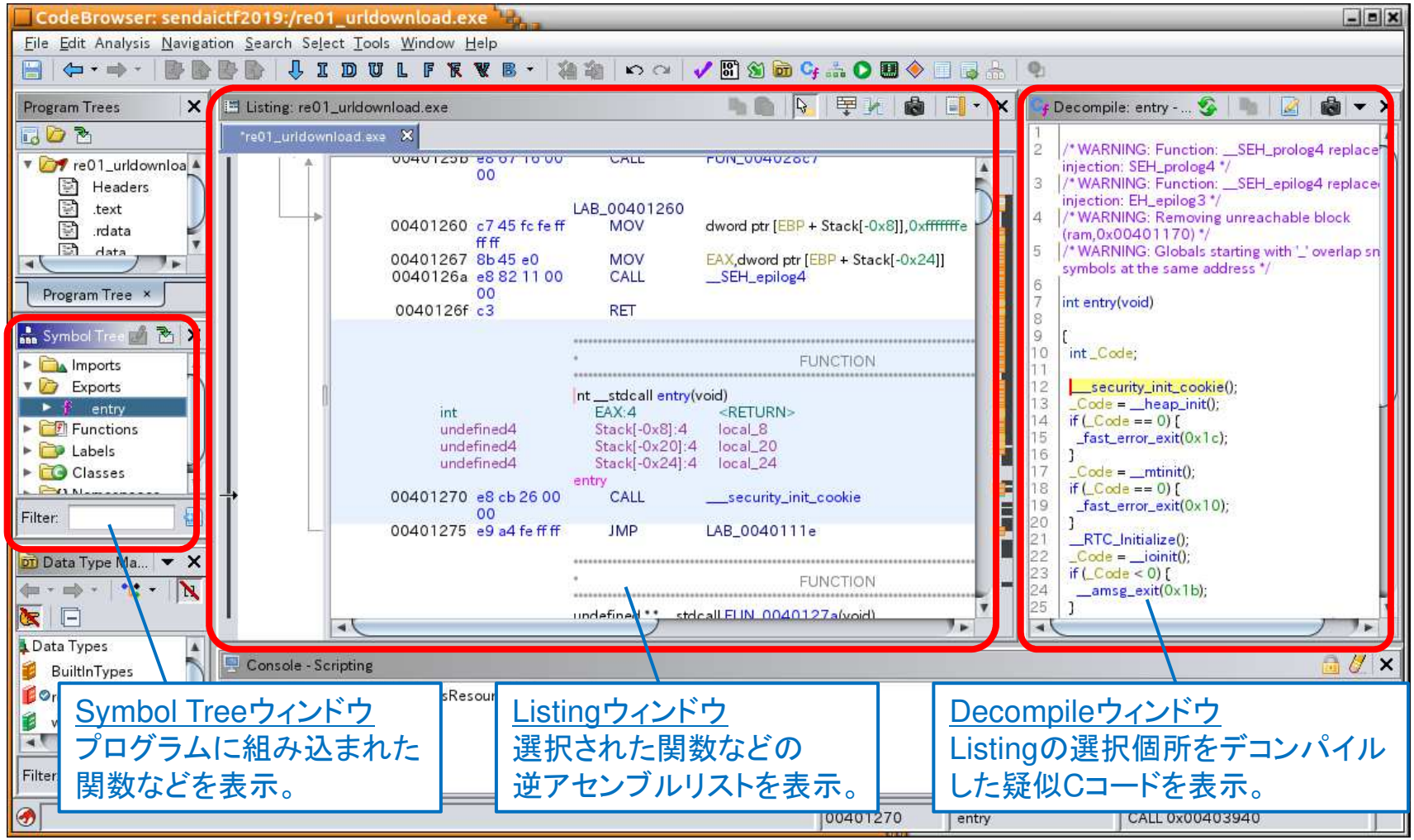
- 「Analyze」ダイアログが表示されたら「Yes」をクリックします。
- 「Analysis Options」ダイアログは、そのまま「Yes」をクリックします。
- 自動解析処理が終了し「Auto Analysis Summary」ダイアログが表示されたら「OK」をクリックします。（PDBに関する警告が表示されますが問題ありません。）



実習1

Ghidraの画面構成

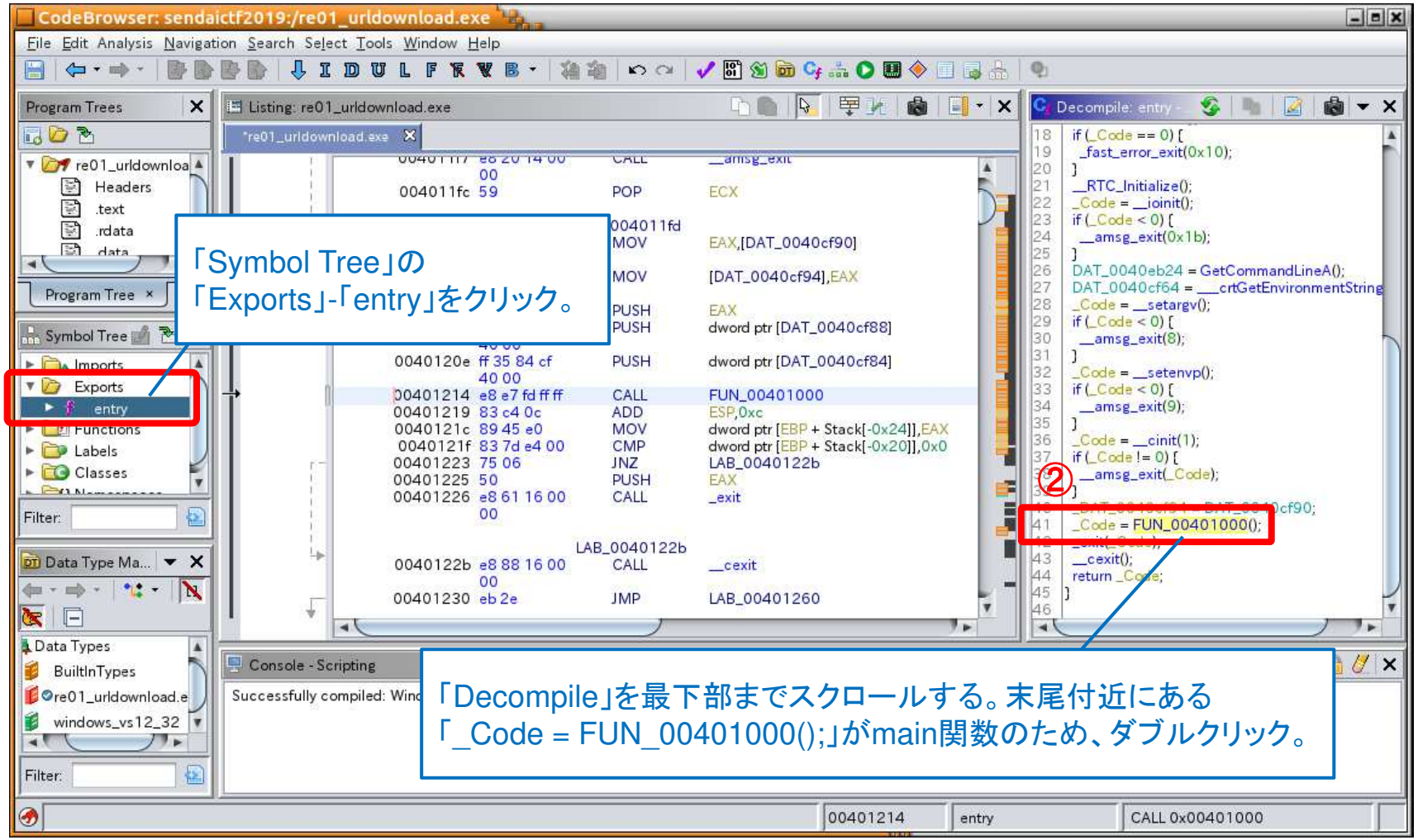
- 本実習では、主に「Symbol Tree」、「Listing」、「Decompile」の3種類のウィンドウを利用します。



実習1

main関数の表示

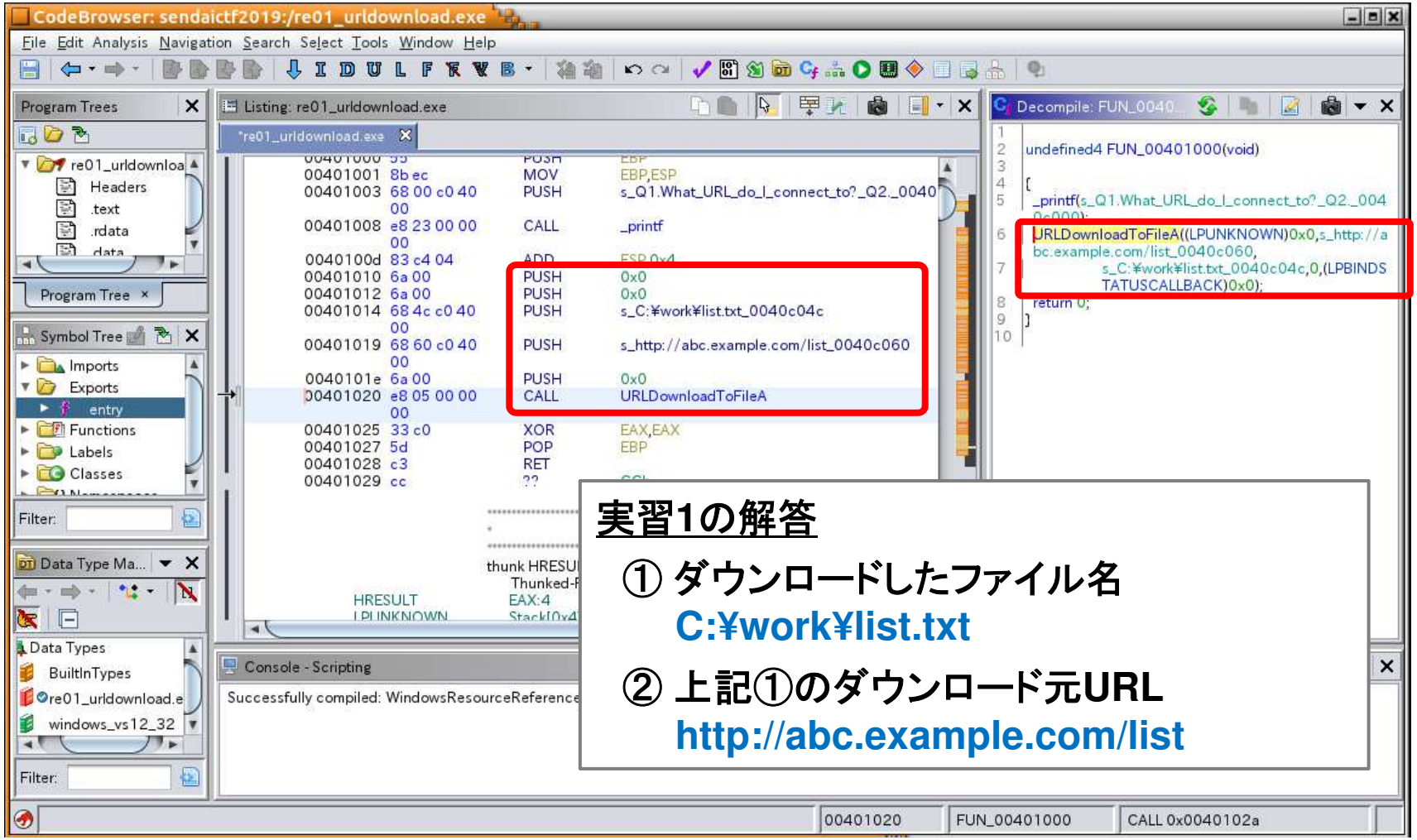
- Ghidraは、main関数を認識してくれないため、下図手順でmain関数を表示します。
  （他のプログラムでも同様の手順でmain関数を表示できます。）



実習1

main関数の逆アセンブルリスト

- 「Listing」と「Decompile」から、Windows API「URLDownloadToFileA」の引数を確認します。



実習　不審プログラムの調査

（別紙. TIPS-3実習資料 参照）

実習2　動作条件の特定

実習用プログラム「/var/samba/public/tips3/re02_getlocaltime.exe」を解析し、
プログラムが動作する条件を特定してください。

実習3　起動される外部プログラムの特定

実習用プログラム「/var/samba/public/tips3/re03_winexec.exe」を解析し、
同プログラムから起動される外部プログラムを特定してください。

実習4　情報流出したファイルの特定

実習用プログラム「/var/samba/public/tips3/re04_malware_easy.exe」を解析し、
情報流出したファイル名および通信先FQDNを特定してください。

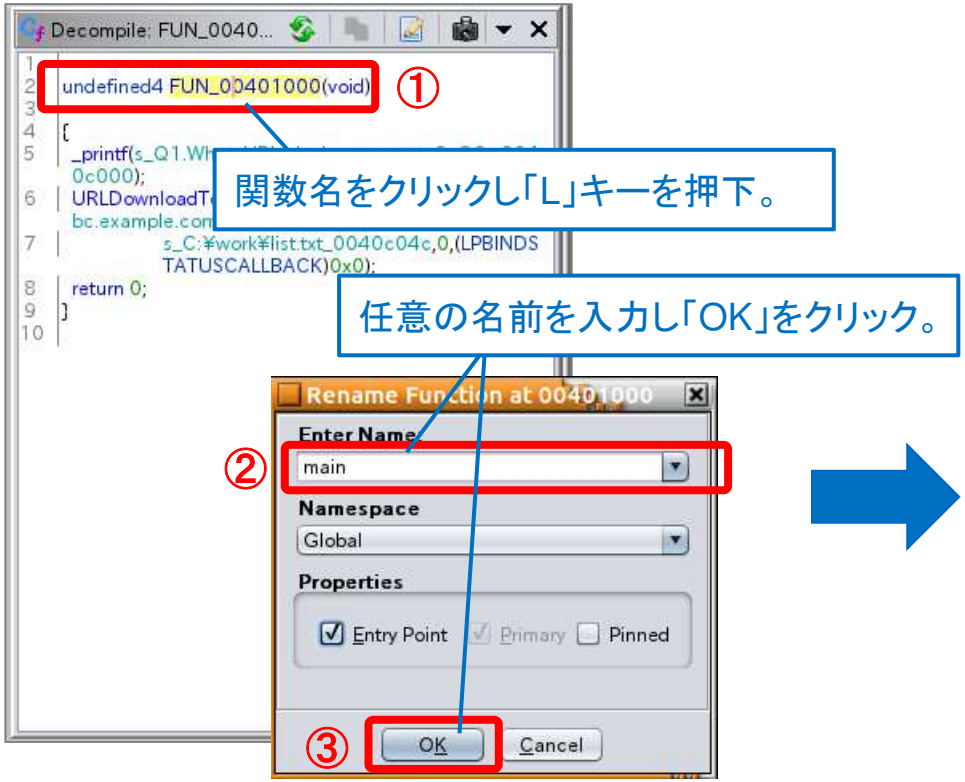
（注記）本プログラムは、開発用サーバで起動されたプログラムを実習で解析
しやすいよう修正したものです。

解析に便利な機能（1）関数名・変数名の変更

- 関数名・変数名（Ghidraではラベルと呼ぶ）を分かりやすい名前に変更すると、コードが読みやすくなります。

◆関数名・変数名の変更方法

（Before）

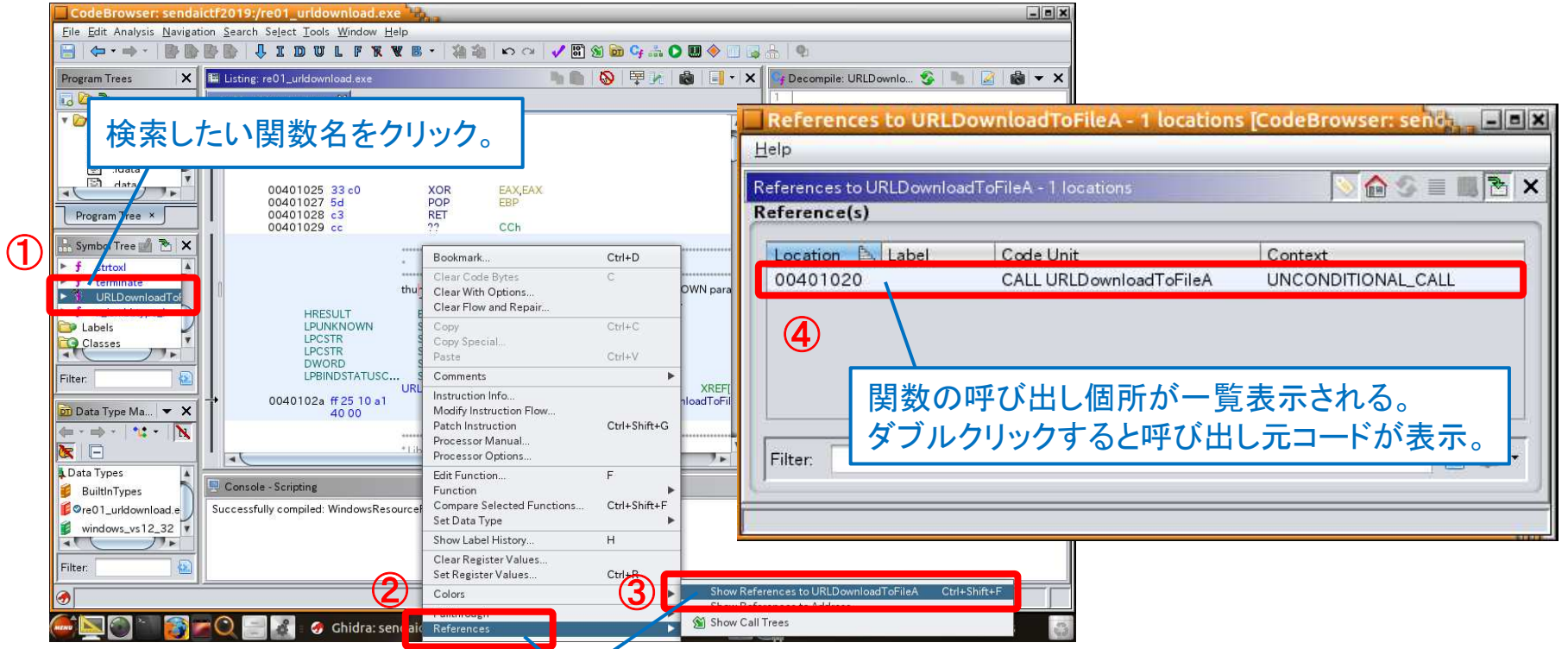


（After）

解析に便利な機能（2）参照機能

- 参照機能（Reference）を使うと、関数の呼び出し元・文字列の参照元コードを簡単に探し出すことができます。

◆関数の呼び出し元の検索



講義

実戦におけるマルウェア解析

- 本物のマルウェアは解析妨害機能があり、一筋縄ではいきません。

- 実戦では時間の制約もあり、静的解析を実施できる機会は少ないかもしれませんが、静的解析の知識（アセンブリの知識）は、脆弱性攻撃やセキュリティ対策製品の動作原理の理解にもつながるため、ぜひ学習に挑戦してみてください。

いきなり体験！ インシデント対応

講義

開発用サーバのインシデント対応の振り返り

- **マルウェア解析の結果、開発用サーバ内の本番データ（顧客情報）が流出した可能性が高いことが判明しました。**

  本事案では外部セキュリティ専門家の支援を受けるべきだったかもしれない。
  しかし、組織・事案ごとに状況が異なり、時間的制約やコストの問題もあり、絶対的な正解はない。**自組織の危機管理体制のなかで判断するしかない。**

  （注記）本勉強会の実習では、マルウェア解析により情報流出の有無を特定したが、
  一般的には、通信ログ、メモリ解析なども含め総合的に判断する。
  （マルウェアの静的解析だけで判断することは無い。）

- **本事案では、開発用サーバのメモリを保全せずにシャットダウンしてしまいました。**

  事象究明のために、メモリ解析が必要な場合も多い。
  インシデント発生時には、メモリも含め必要なエビデンスを速やかに保全できるよう事前に初動対応手順を整備しておくことが望ましい。

講義

重大インシデントへの備え

- サイバー攻撃によるインシデント対応には、多大な労力・コストがかかります。

- 自組織のシステム対策、運用体制などを再確認し、必要な対策を計画的に実施することでインシデントの未然防止を図ることが大切です。
- また、インシデント発生時に迅速・的確に対応するためには、事前に対応手順を整備しておくことも大切です。

まとめ

まとめ

インシデント発生時は、自組織への影響を最小化するための
「危機管理」として対応

ネットワーク、ログ解析、マルウェア解析などの知識があると
原因調査に役立つため、実務者のスキルアップを図る

インシデント発生時に迅速・的確に対応するためには、
事前準備が大切