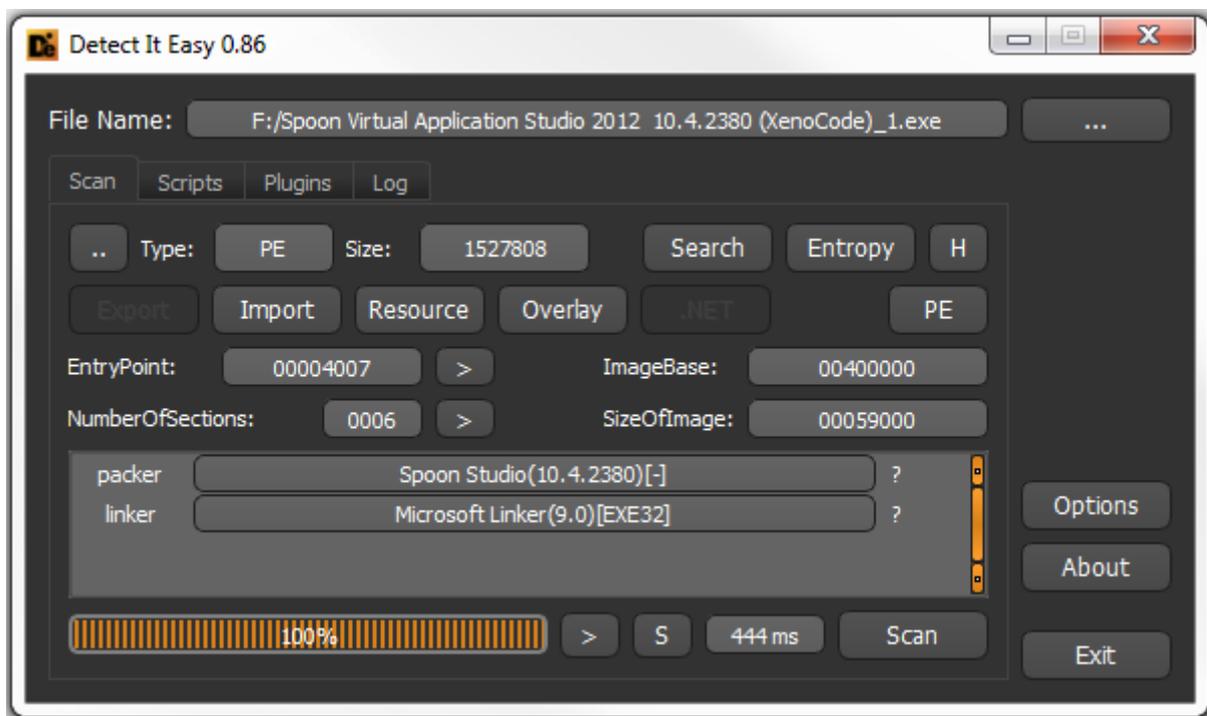


В этой статье будет подробно описано, как создавать сигнатуры для программы „Detect It Easy“. Detect It Easy или сокращённо DIE это программа для определения типов файлов.

DIE – кроссплатформенное приложение и кроме Windows-версии есть версии для Linux и Mac OS.



Многие программы подобного рода(PEID, PE tools) позволяют использовать сторонние сигнатуры. Но к сожалению эти сигнатуры сканируют только байты по заданной маске и невозможно задать дополнительные параметры. Поэтому часто случаются ложные срабатывания. Более надежные алгоритмы обычно жестко прописаны в самой программе. И для добавления нового сложного детекта нужно перекомпилировать весь проект. Никто, кроме авторов, не может изменить алгоритм детекта. И без постоянной поддержки такие программы со временем теряют свою актуальность.

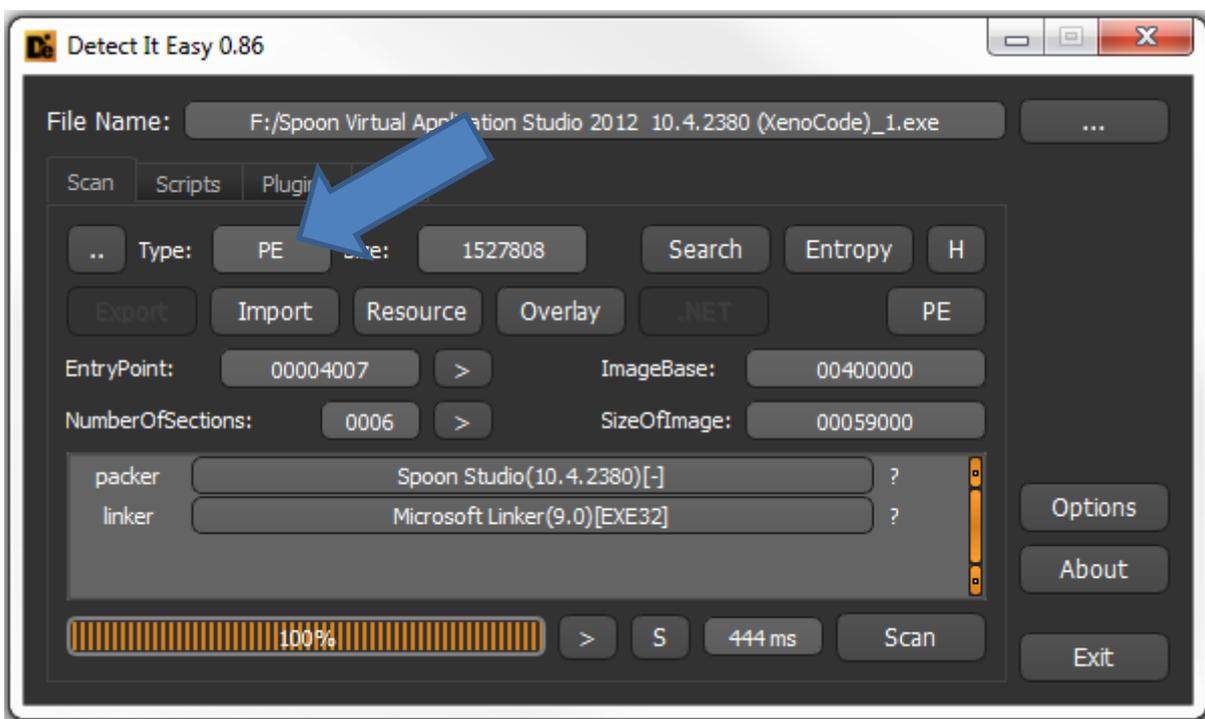
Detect It Easy имеет полностью открытую архитектуру сигнатур. Можно легко добавлять свои алгоритмы детекта или изменять уже имеющиеся. Это достигается использованием скриптов. Язык скриптов очень похож на JavaScript и любой человек, понимающий основы программирования, без труда разберется, как это работает. Возможно кому-то покажется, что скрипты работают очень медленно. Действительно скрипты работают медленнее скомпилированного кода, но благодаря хорошей оптимизации Script Engine это не причиняет особых неудобств. А возможности открытой архитектуры компенсируют недостатки.

DIE существует в трех версиях. Обычная версия(DIE), версия Lite(DIEL) и консольная(DIEC). Все эти три версии используют одни и те же сигнатуры, которые лежат в папке «db». Если открыть эту папку, то можно найти вложенные подпапки(„Binary“, „PE“ и другие). Имена

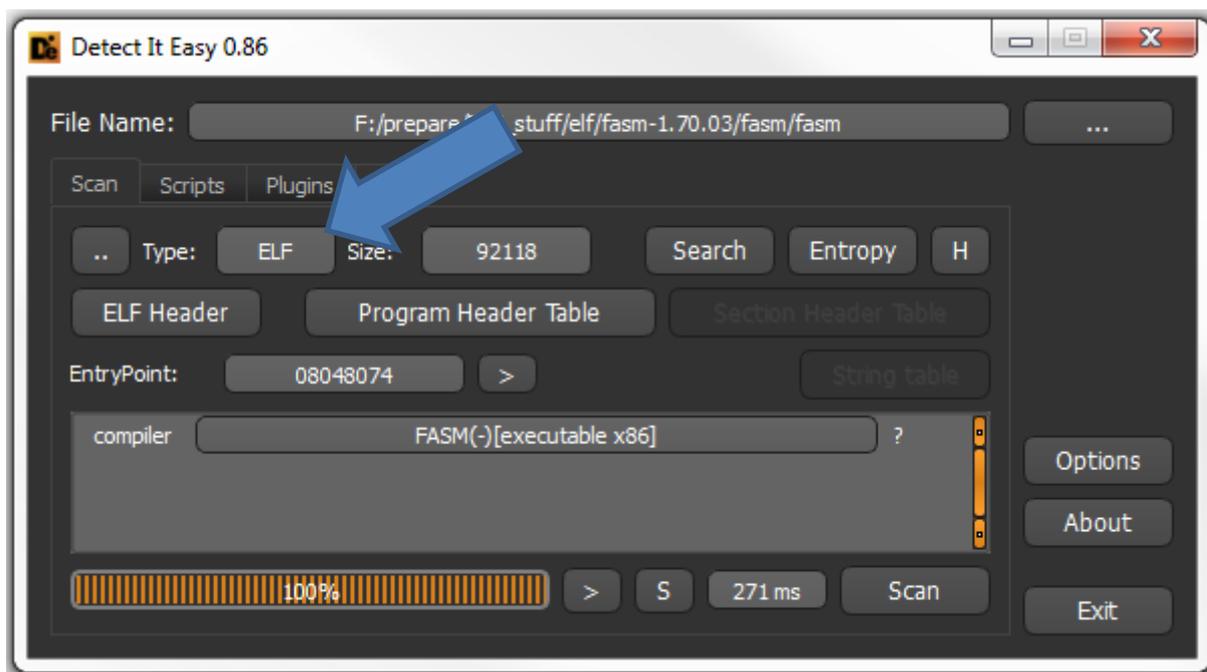
подпапок соответствуют типам файлов. Сначала DIE определяет тип файла, а затем последовательно загружает все сигнатуры, которые лежат в соответствующей папке. В настоящее время программа определяет следующие типы:

- **MSDOS** исполняемые файлы MS-DOS
- **PE** исполняемые файлы Windows
- **ELF** исполняемые файлы Linux
- **MACH** исполняемые файлы Mac
- **Text** текстовые файлы
- **Binary** все остальные файлы

Открыв файл в DIE можно узнать тип файла:



Центральная часть окна также меняется в зависимости от типа:



Сами сигнатуры являются обыкновенными текстовыми файлами, только с расширением (*.sg). Их можно редактировать в любом текстовом редакторе. Каждая сигнатура должна иметь в своем теле функцию detect, которая запускается при загрузке. Прототип этой функции такой:

function detect(bShowType,bShowVersion,bShowOptions)

где bShowType, bShowVersion и bShowOptions флаги, которые могут иметь значение 0 или 1. Если сигнатура успешно сработала, то функция detect должна вернуть строку в соответствии с установленными флагами.

Покажем это на примере.

Если все три флага установлены, то для упаковщика UPX возвратится примерно такая строка:

packer: UPX(3.09)[DLL32]

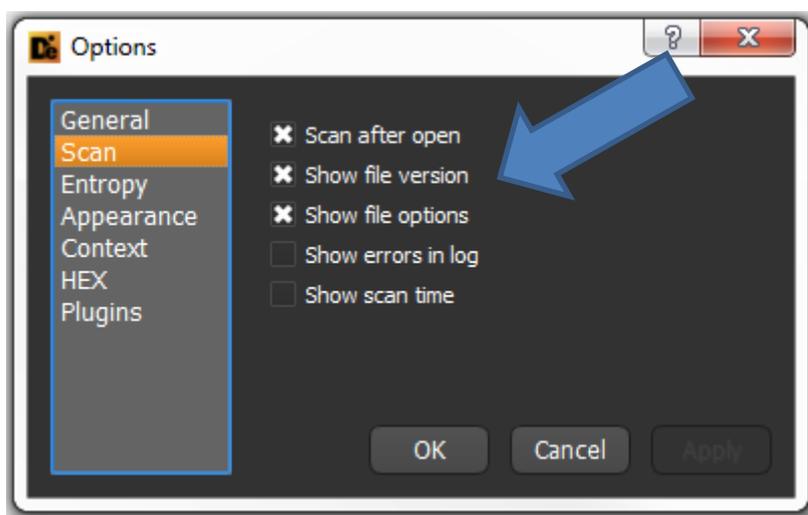
- **packer** - тип сигнатуры(Type). Не надо путать тип сигнатуры с типом файла. Например для ASPack <http://www.aspack.com/aspack.html> тип файла PE, а тип сигнатуры packer.
- **UPX** - имя(Name)
- **3.09** - версия(Version)
- **DLL32** - опции(Options)

Можно проигнорировать такую рекомендацию и просто возвращать строку произвольного формата. Но тогда возможно обычная версия(DIE) не сможет корректно отобразить результат(для версии lite и консольной версии это не имеет значения). Чтобы DIE смог правильно показать результат, надо чтобы строка хотя бы имела тип и имя, разделенные «:».

Например:

Type: Name

В консольной и обычной версии (DIE и DIEЛ) можно управлять флагами bShowVersion и bShowOptions. В обычной версии это можно сделать в настройках.



В консольной версии это ключи -showoptions и -showversion (по умолчанию оба флага установлены в 1).

Если воспользоваться готовым шаблоном сигнатуры, то не надо думать как правильно сформировать возвращаемую строку. В шаблоне это все уже реализовано.

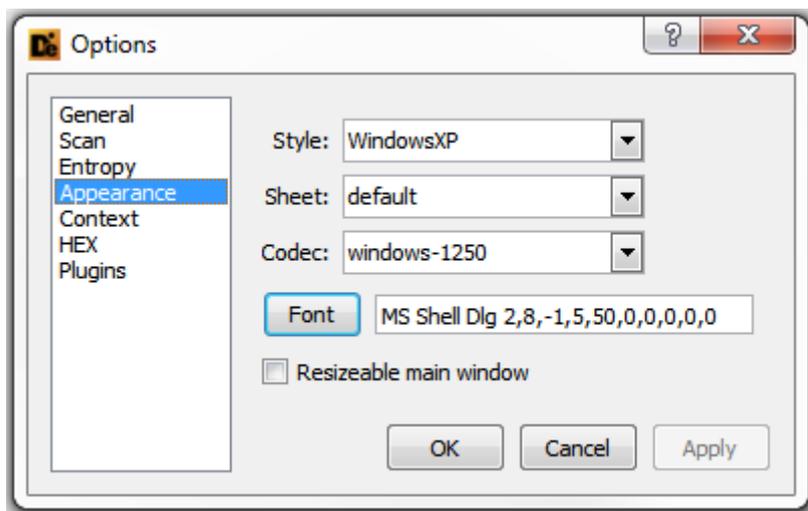
Присутствие функции detect в сигнатуре обязательно, но кроме этой функции в файле могут находиться и пользовательские функции. Например:

```
function sum(arg1, arg2)
{
    return arg1+arg2;
}
function detect (bShowType, bShowVersion, bShowOptions)
{
    var k=1;
    var l=3;
    var s=sum(k, l);
```

Как уже говорилось, для редактирования и создания новых сигнатур можно использовать любой текстовый редактор. Также это осуществимо и средствами самой программы. Для этого запустим обычную версию DIE и откроем какой-нибудь файл.

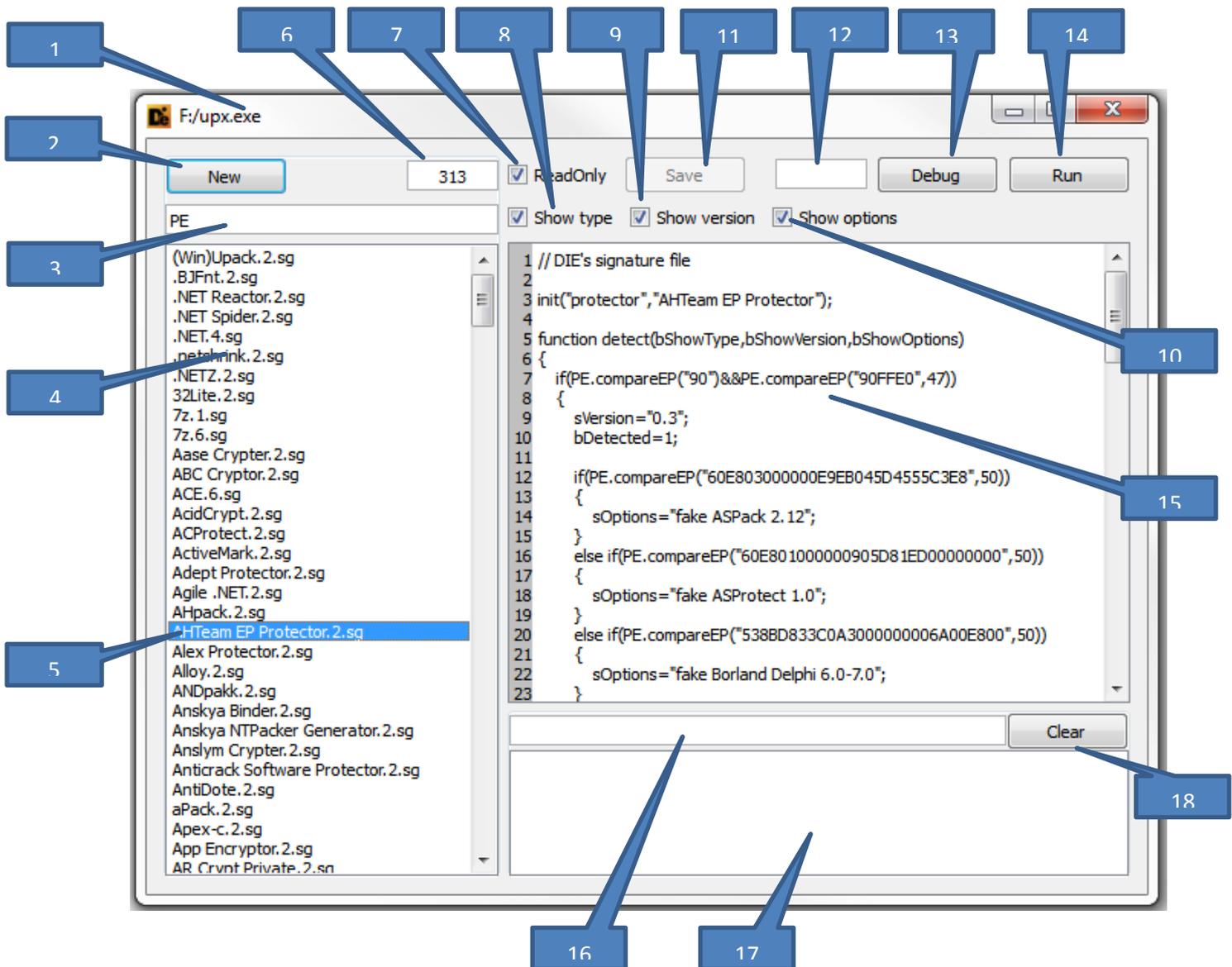
Небольшое замечание, для комфортной работы с сигнатурами (особенно с отладчиком) лучше отключить все пользовательские стили оформления. Для этого переходим на вкладку «Appearance» меню настроек и переключаем стили на используемые по умолчанию:

Style	WindowsXP
Style sheet	default



Затем нажмем кнопку с надписью «S» внизу окна.

Откроется редактор сигнатур



- 1) Заголовок окна с именем открытого файла.
- 2) Кнопка создания новой сигнатуры.
- 3) Тип открытого файла.
- 4) Список имеющихся сигнатур для этого типа.
- 5) Выбранная сигнатура.
- 6) Количество имеющихся сигнатур.
- 7) Включение режима редактирования.
- 8) Установка флага bShowType.
- 9) Установка флага bShowVersion.
- 10) Установка флага bShowOptions.

- 11) Сохранение сигнатуры.
- 12) Время выполнения сигнатуры.
- 13) Запуск выполнения кода сигнатуры в отладчике.
- 14) Запуск выполнения кода сигнатуры.
- 15) Код выбранной сигнатуры.
- 16) Строка результата выполнения сигнатуры.
- 17) Окно лога.
- 18) Очистка строки результата и окна лога.

Каждый тип файла имеет собственные встроенные функции. Они вызываются как

<тип файла>.<имя функции>

Например Binary.readDword или PE.compareEP. Список встроенных функций постоянно расширяется. Полное описание всех функций можно найти в SDK/signatures/index.html

Список встроенных функций постоянно расширяется и если есть идеи, какие функции нужно еще добавить, пишите на horsicq@gmail.com

Кроме того, существуют и так называемые «глобальные функции» которые можно вызывать везде.

Самой важной из этих функций является “includeScript”. С помощью этой функции можно включать в скрипт другие скрипты.. „\$DIEAPP/db“ является папкой по умолчанию для включения скриптов.

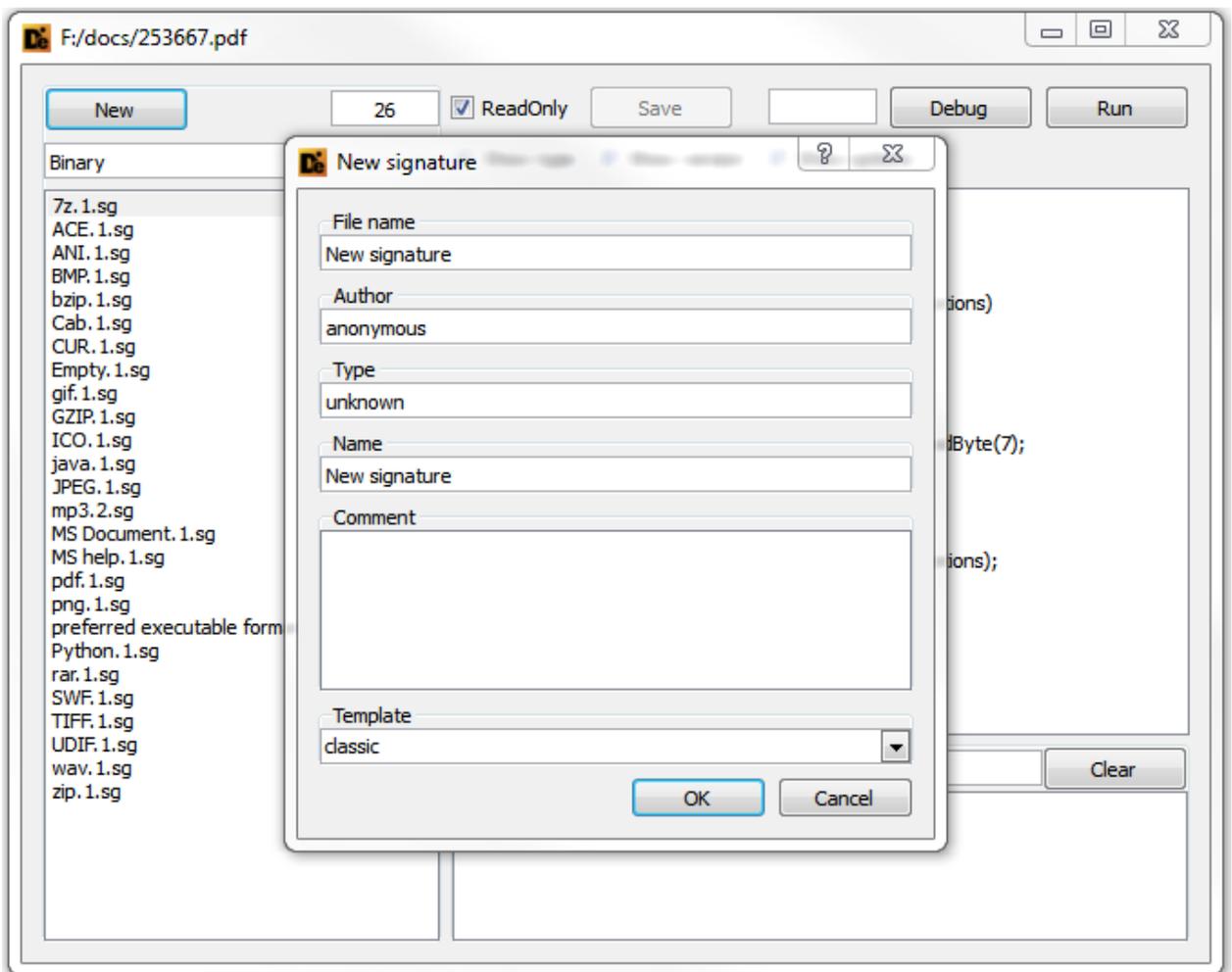
```
includeScript("result"); // добавляет скрипт $DIEAPP/db/result.
```

Также удобно использовать функцию «_log» для вывода отладочных сообщений.

```
_log("Hello world!"); // Выведет строку.  
_log(123); // Выведет число.
```

Существует особый файл «_init» лежащий в папке каждого типа. Например “\$DIEAPP/db/PE/_init”. Этот файл загружается по умолчанию перед выполнением всех скриптов и туда можно поместить функции и глобальные переменные общие для этого типа файлов(в данном случае «PE»)

Теперь, в качестве примера, попробуем создать сигнатуру для PDF. Для этого откроем какой-нибудь файл этого типа. Затем перейдем к редактору сигнатур и нажмем кнопку «New». Появится окно создания новой сигнатуры.



В качестве имени файла можно задать любую строку, но чтобы потом было легче разбираться с сигнатурами, лучше чтобы имя файла совпадала с именем определяемой сигнатуры. То есть если создается для PDF, то и файл лучше всего назвать „PDF“, а не к примеру „new3“. Если файл с таким именем уже существует, то можно воспользоваться подчеркиванием или цифрами(_PDF,PDF2). Конечно можно создать сигнатуру которая бы определяла сразу несколько видов файлов (например PDF и JPG), но лучше делать отдельные сигнатуры.

Также можно выставить приоритет сканирования. Хотя это и не обязательно. Для этого имя файла должно иметь вид:

Имя.[приоритет].sg

Приоритет может иметь значения от 0 до 9. 0 – самый высокий, 9 – самый низкий.

Например «MicroJoiner.1.sg» или «Microsoft Visual Studio.3.sg». Это иногда бывает полезно чтобы результат сканирования выводился в определенной последовательности.

Например, информация о компиляторе выводилась раньше информации о линкере.

Чтобы это произошло можно использовать для компиляторов приоритет 3, а для линкеров приоритет 5.

Вводим в диалог следующие данные:

File Name	PDF	Имя файла.
Author	Me	Можно оставить это поле пустым.
Type	Format	Можно использовать любую строку, но оставлять пустым нежелательно.
Name	PDF	Имя определяемой сигнатуры.
Comment	11.06.2014	Можно оставить это поле пустым.

Имя файла может совпадать с именем определяемой сигнатуры, а может и не совпадать.

Ещё можно выбирать используемый шаблон. Все шаблоны находятся в папке „\$DIEAPP/editor/templates” и их тоже можно при желании свободно редактировать или добавлять новые.

Если выбрать «классический» шаблон(«classic»), то появится файл «PDF.sg» следующего вида:

```
// DIE's signature file
// Author: Me
/*
11.06.2014
*/
function detect (bShowType,bShowVersion,bShowOptions)
{
    var sType="Format";
    var sName="PDF";
    var sVersion="-";
    var sOptions="-";
    var sResult="";
    var nDetected=0;
    // Start of user's code
    // End of user's code
    if(nDetected)
    {
        if(bShowType)
        {
            sResult+=sType+": ";
        }
        sResult+=sName;
        if(bShowVersion)
        {
            sResult+=" (" +sVersion+") ";
        }
        if(bShowOptions)
        {
            sResult+=" [" +sOptions+"] ";
        }
    }
    return sResult;
}
```

Ищем в поисковике (подойдет Google.com) информацию о структуре PDF файлов. К примеру это <http://resources.infosecinstitute.com/pdf-file-format-basic-structure/>

Итак в самом начале любого находится двойное слово 0x46445025, а по смещению 5 находится строка с номером версии длиной 3 символа.

Пишем следующий код:

```
// Start of user's code
if(Binary.getSize()>=8) // Если длина файла меньше 8 байт, то очевидно это не
PDF
{
    if(Binary.readDword(0)==0x46445025) // Читаем первое двойное слово и
сравниваем.
    {
        sVersion=Binary.getString(5,3); // Определяем версию
        nDetected=1; // Устанавливаем в 1, если проверки прошли успешно
    }
}
// End of user's code
```

Чтобы проверить сигнатуру нажмем на кнопку “Run”

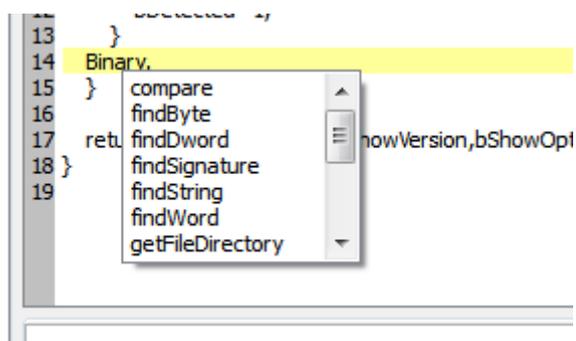
В окне результата появится строка:

Format: PDF(1.2)[-]

Так как „options“ будет всегда „-“, можно удалить следующие строки кода

```
var sOptions="-";
if(bShowOptions)
{
    sResult+="["+sOptions+"]";
}
```

Если в окне редактора набрать „<тип>.“ и нажать комбинацию клавиш «Ctrl+Space», то появятся все встроенные функции для данного типа (в данном случае «Binary»):



Если навести текстовый курсор на встроенную функции и нажать комбинацию клавиш «Alt», то появится прототип функции:

```

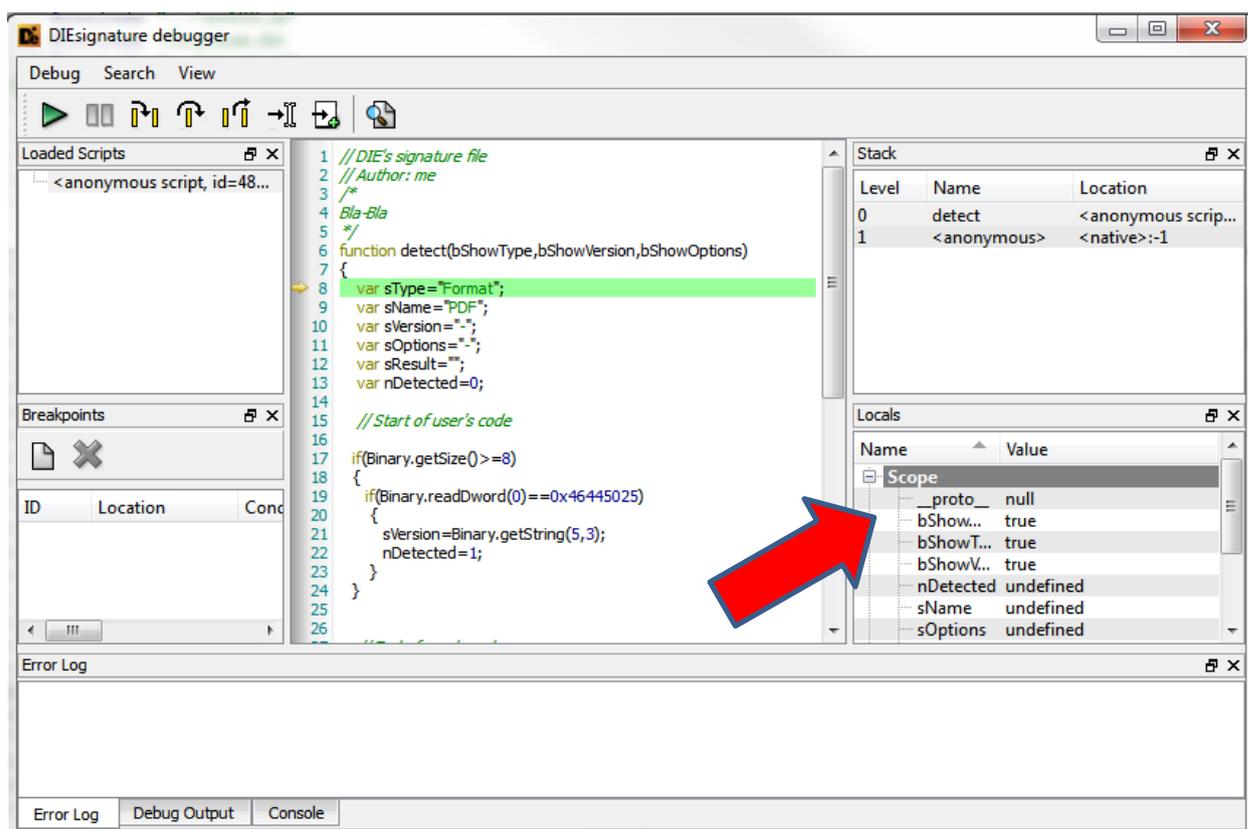
if(Binary.getSize()>=8)
{
  if(Binary.readDword(0)==0x46445025)
  {
    sVersion=Binary.getString(5,3);
    nDetected=1;
  }
}

```

QString getString(unsigned int nOffset,unsigned int nSize=50)

Сигнатуры также можно отлаживать. Для этого нужно нажать кнопку “Debug”.

Появится окно:



Отладчик имеет стандартное управление. Все используемые локальные переменные находятся в „Scope“.

Сигнатуры для PEID можно использовать и в DIE для типа „PE“.

К примеру имеется такая PEID-сигнатура:

```

[Stones PE Encryptor v1.13]
signature = 55 57 56 52 51 53 E8 ?? ?? ?? ?? 5D 8B D5 81
ep_only = true

```

Запускаем редактор сигнатур для PE-файлов. Для этого нужно открыть любой PE-файл и нажать кнопку «New»

Затем вводим следующие данные:

File Name	Stones PE Encruptor	Имя
Author	Me	Можно оставить это поле пустым.
Type	Protector	Здесь можно написать другой тип. Например Cryptor
Name	Stones PE Encruptor	Имя определяемой сигнатуры.
Comment	-	Можно оставить это поле пустым.

The image shows a 'New signature' dialog box with the following fields and values:

- File name: Stones PE Encruptor
- Author: Me
- Type: Protector
- Name: Stones PE Encruptor
- Comment: -
- Template: classic

Buttons: OK, Cancel

Появится новый файл с таким содержимым:

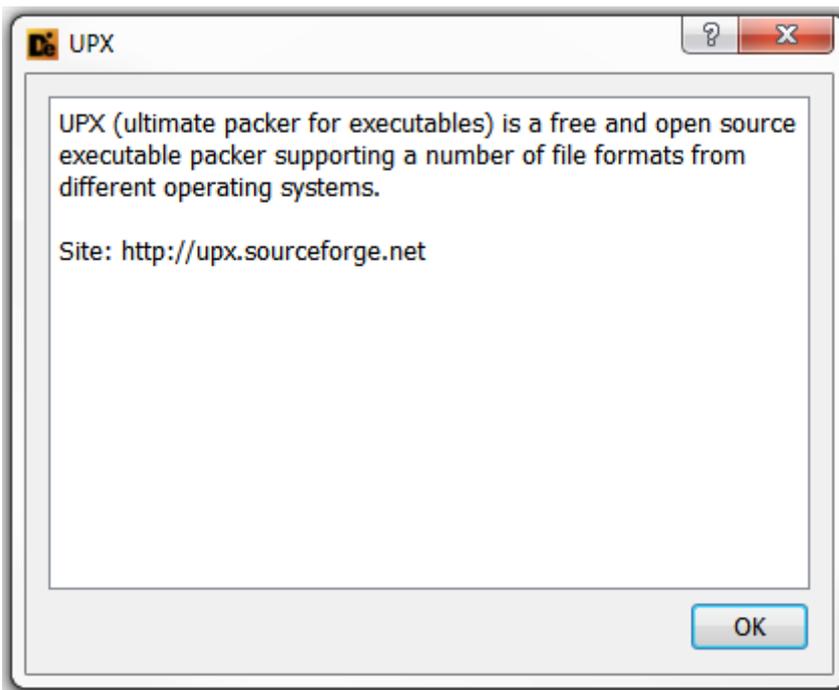
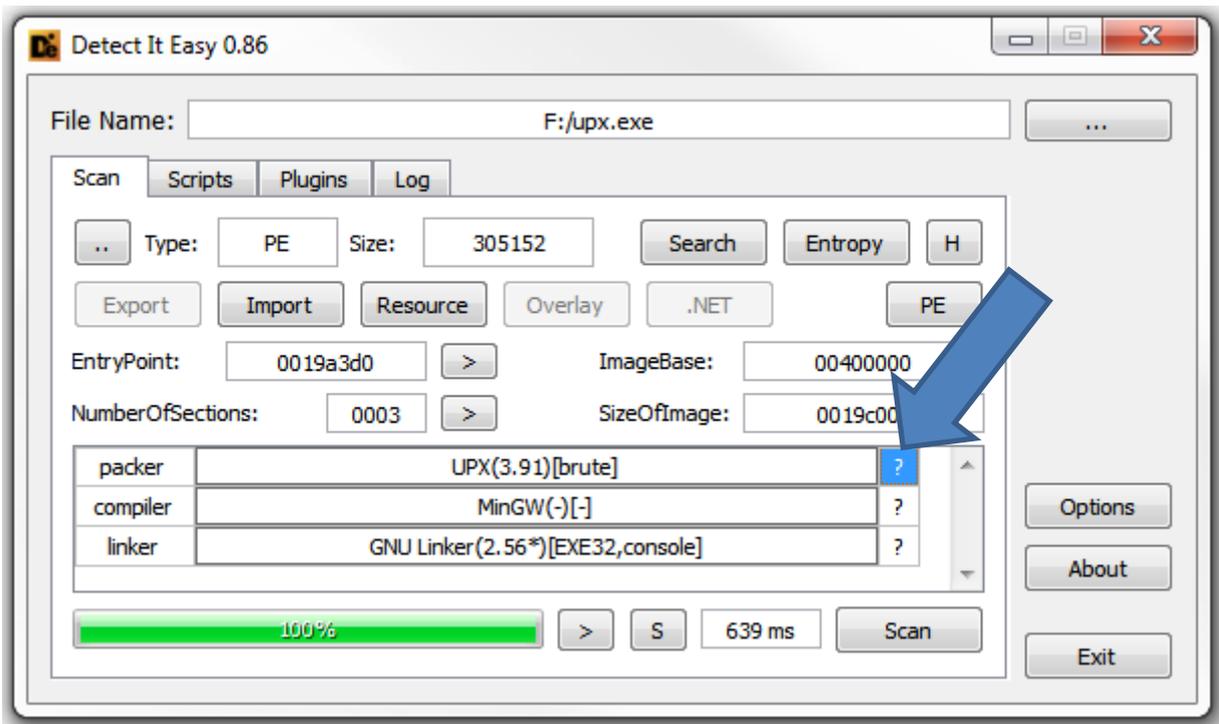
```
// DIE's signature file
// Author: Me
/*
-
*/
function detect (bShowType,bShowVersion,bShowOptions)
{
    var sType="Protector";
    var sName="Stones PE Encryptor";
    var sVersion="-";
    var sOptions="-";
    var sResult="";
    var nDetected=0;
    // Start of user's code

    // End of user's code
    if(nDetected)
    {
        if(bShowType)
        {
            sResult+=sType+": ";
        }
        sResult+=sName;
        if(bShowVersion)
        {
            sResult+=" (" +sVersion+" )";
        }
        if(bShowOptions)
        {
            sResult+=" [" +sOptions+"]";
        }
    }
    return sResult;
}
```

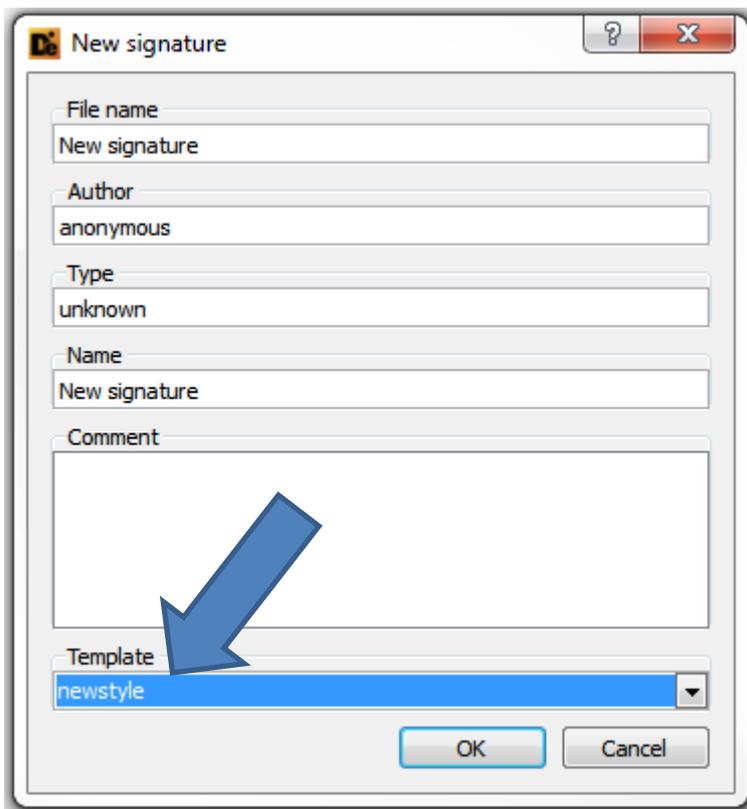
Пишем следующий код:

```
// Start of user's code
if(PE.comparePE("\55 57 56 52 51 53 E8 ?? ?? ?? ?? 5D 8B D5 81")) // Наша peid
сигнатура
{
    sVersion=" 1.13"; // Версия
    sOptions="-"; // Опции неизвестны.
    nDetected=1; // Устанавливаем в 1, если проверки прошли успешно
}
// End of user's code
```

Если есть немного дополнительной информации, то можно поместить её в html файл папки «info» программы. Если нажать на кнопку со знаком вопроса в обычной версии, то появится окно с этой дополнительной информацией. Если конечно соответствующий файл существует. Имя html файла должно соответствовать имени определяемой сигнатуры. Регистр букв тоже должен учитываться.



Как уже говорилось можно использовать собственные шаблоны для скриптов. Jason Hood(<http://adoxa.altervista.org>) написал новую систему шаблонов (“newstyle”):



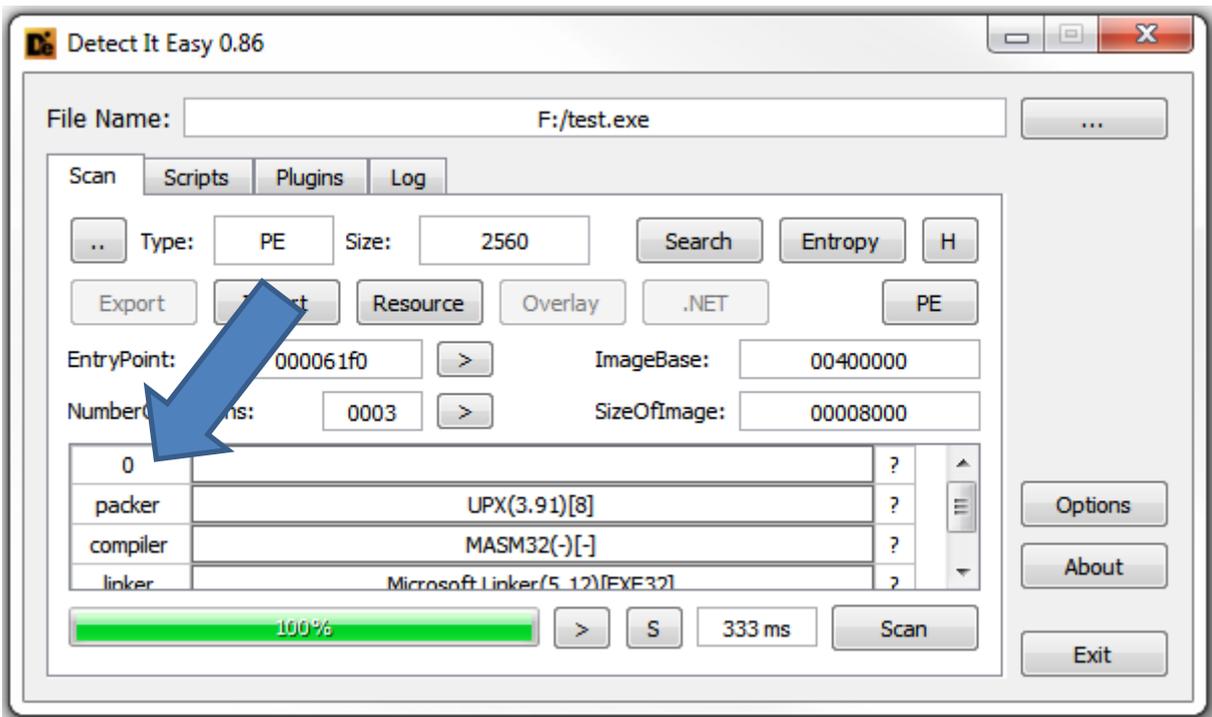
В этой системе шаблонов наш пример можно записать как:

```
// DIE's signature file
// Author: Me
/*
-
*/
init("Protector ", " Stones PE Encryptor ");

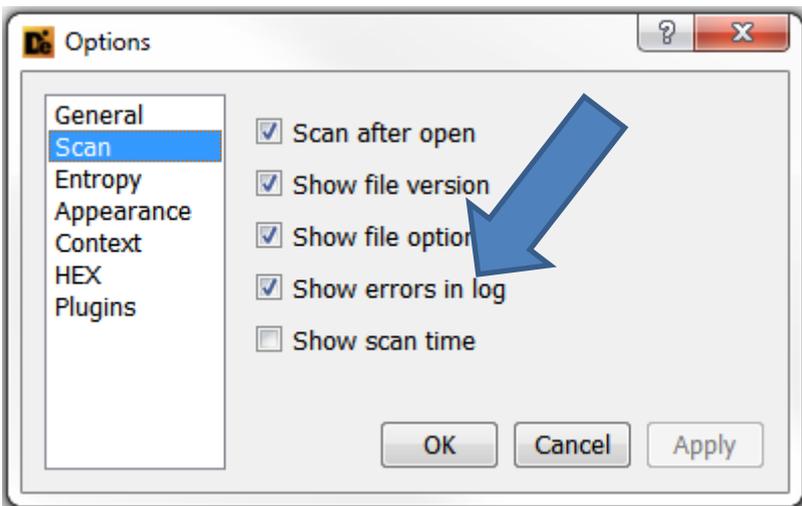
function detect (bShowType, bShowVersion, bShowOptions)
{
    // Start of user's code
    if(PE.comparePE("\55 57 56 52 51 53 E8 ?? ?? ?? ?? 5D 8B D5 81")) // Наша
peid сигнатура
    {
        sVersion=" 1.13"; // Версия
        sOptions="-"; // Опции неизвестны
        bDetected =1; // Устанавливаем в 1, если проверки прошли
успешно
    }
    // End of user's code
    return result (bShowType, bShowVersion, bShowOptions);
}
```

Теперь немного об ошибках в скрипте.

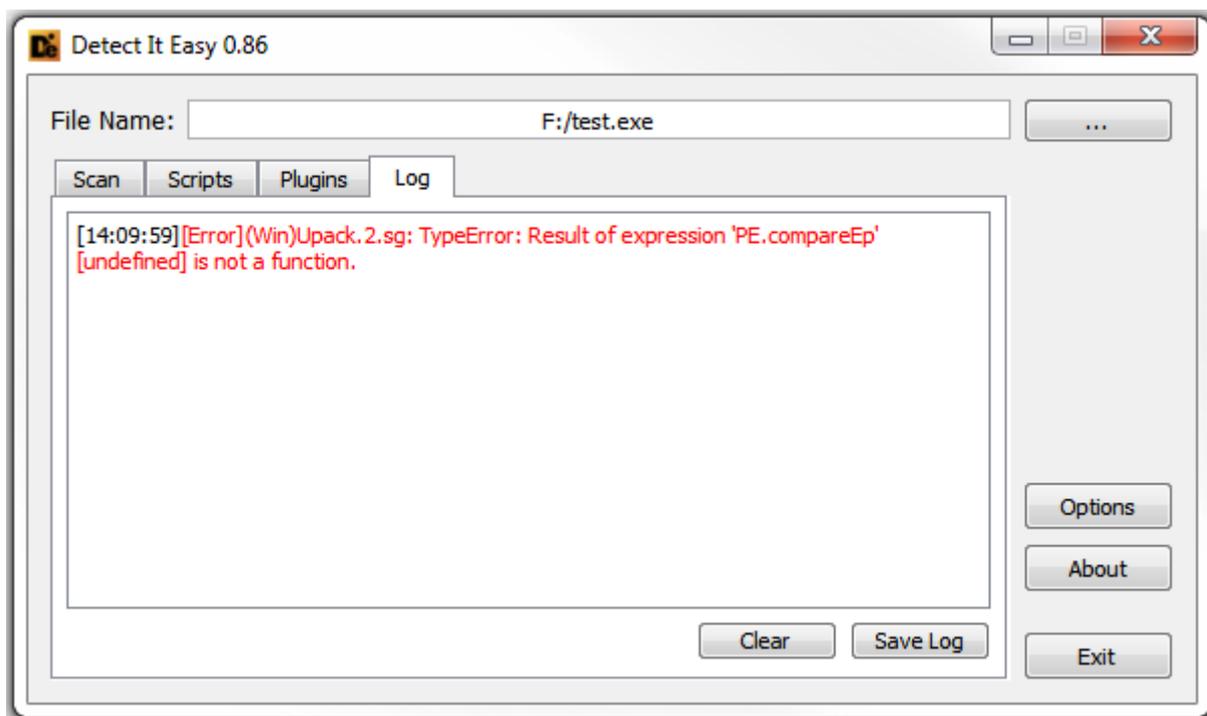
Если во время сканирования файла появится такой результат:



Это означает, что где-то произошла ошибка. Для того чтобы узнать точное место, идем в настройки и выставляем показ ошибок:

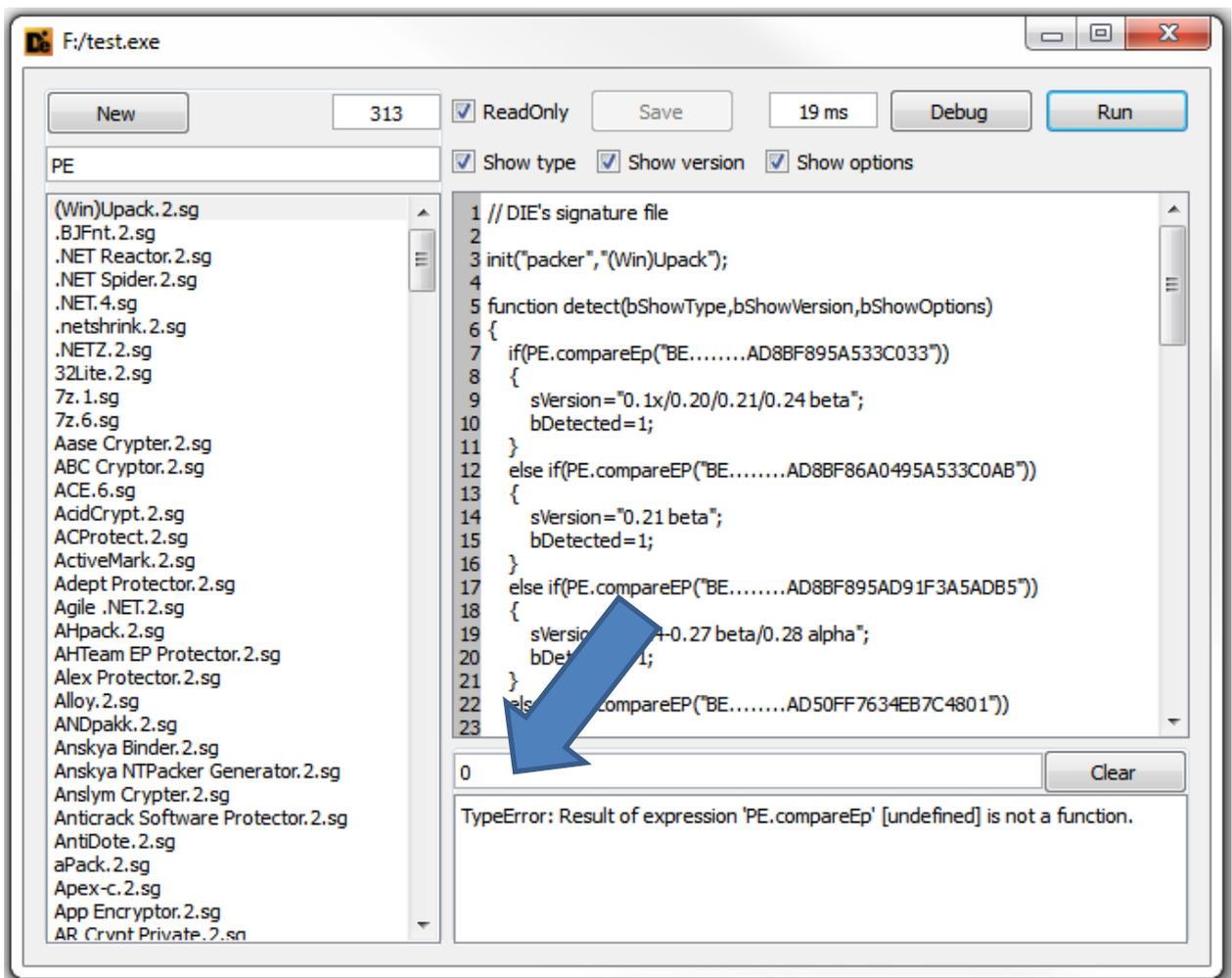


Ещё раз запускаем сканирование файла и затем переходим в окно лога:

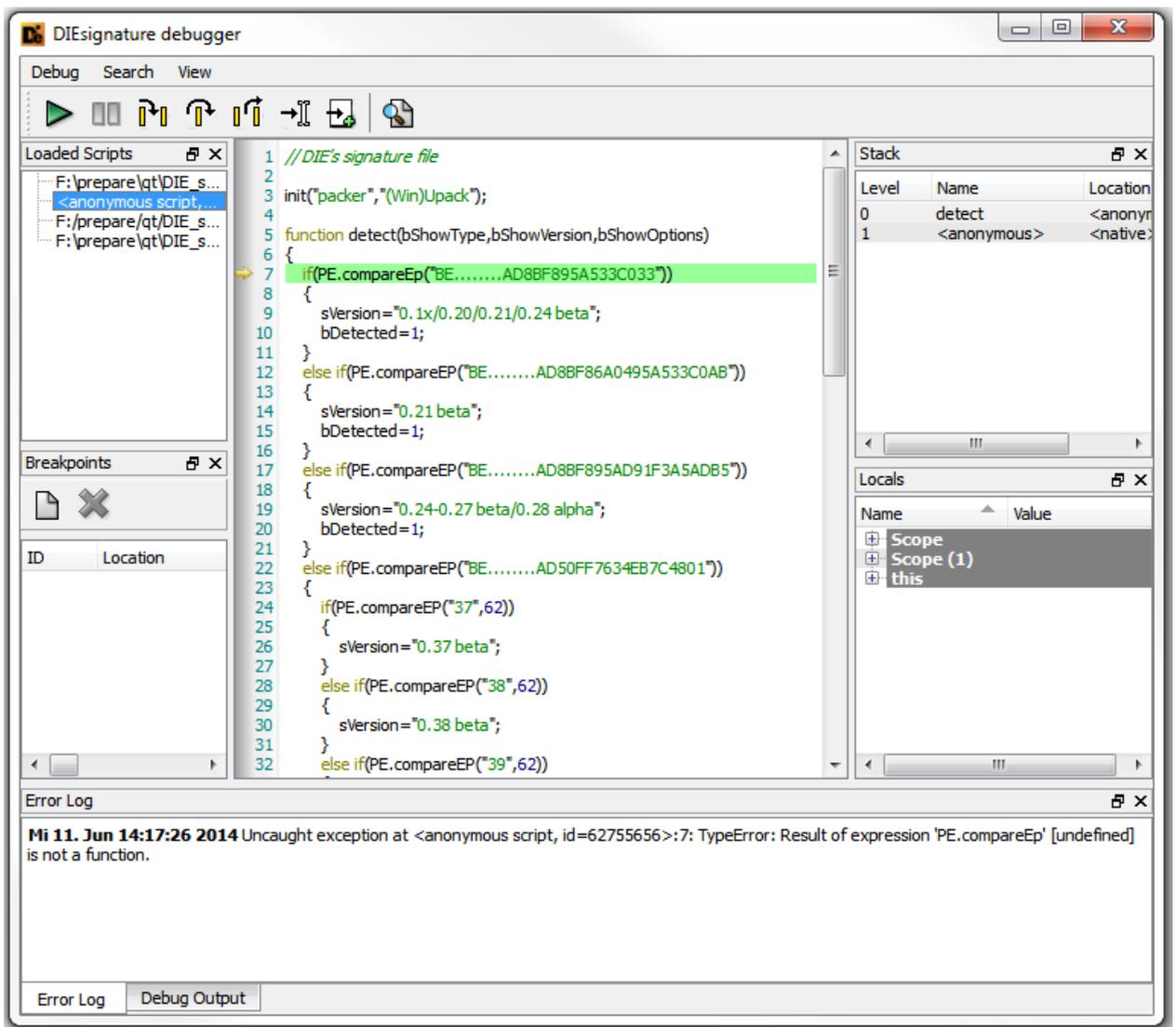


Это значит что ошибка произошла где-то в файле “(Win)Upack.2.sg”

Открываем этот файл в редакторе сигнатур И нажимаем кнопку “Run”

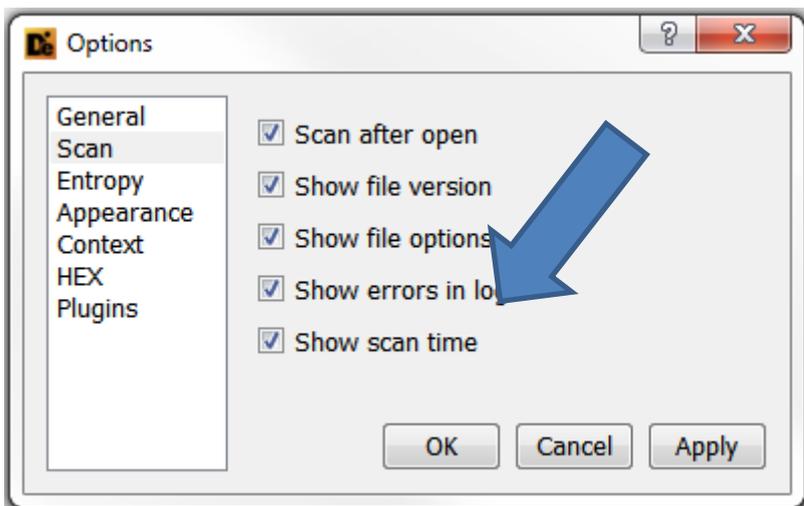


Видно, что проблема в этом скрипте. Для того чтобы точно найти в какой строке ошибка, нажимаем кнопку "Debug" - Откроется отладчик. Нажимаем клавишу "F5" на клавиатуре, отладчик запустится и мы остановимся на проблемном участке:

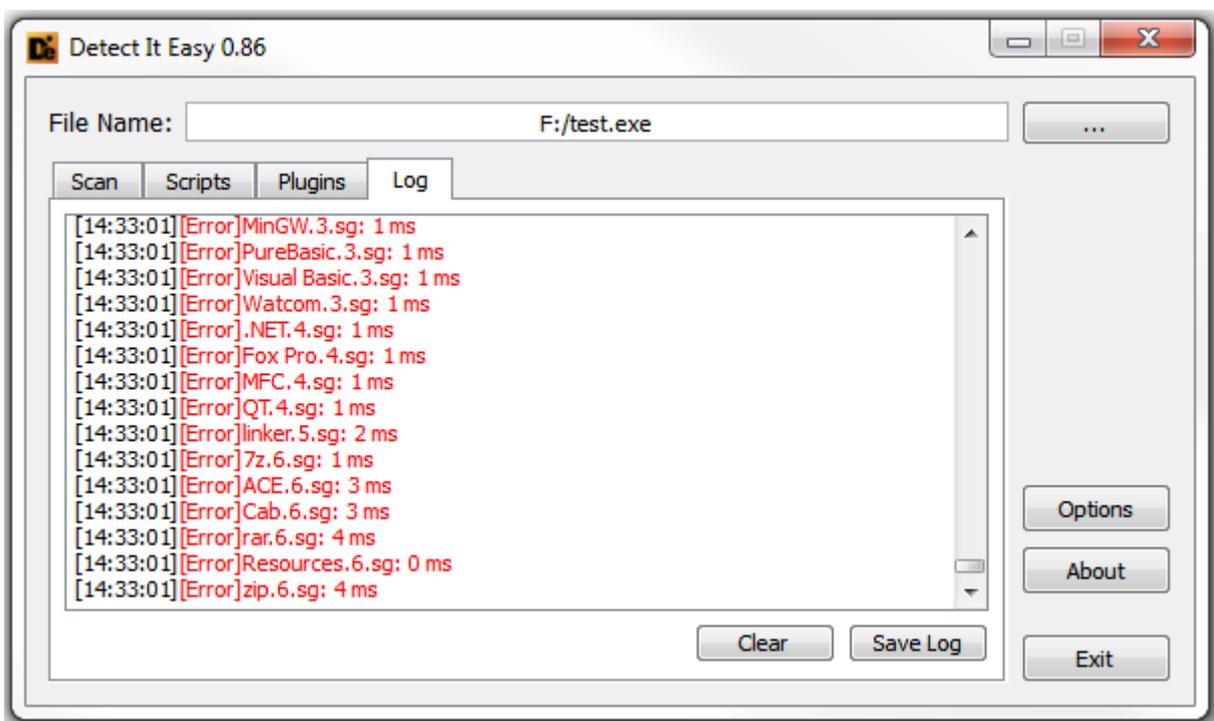


Ошибка в строке 7. Заменяем `compareEp` на `compareEP` и сообщение об ошибке исчезает. Язык скриптов DIE основан на JavaScript и поэтому имена переменных и функций чувствительны к регистру, это означает что `compareEp` отличается от `compareEP`. В этом отличие например от Pascal, где регистр не имеет значения.

В некоторых случаях время сканирование файла имеет значение, поэтому иногда полезно оптимизировать код сигнатур для наилучшего быстродействия. Чтобы узнать скорость срабатывания отдельных сигнатур для определенного файла нужно в настройках выставить показ времени сканирования и показ ошибок:



Теперь в лог будет записываться время сканирования каждой сигнатуры по отдельности.



DIE в настоящее время еще не достиг финальной версии (1.0) поэтому активно улучшается, если есть новые идеи по улучшению функционала, то пишите на horsicq@gmail.com