ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цель работы – разработать программу на .NET C#, позволяющую пользователю нажатием одной кнопки проанализировать ОС Windows на уязвимости безопасности и получить отчёт с советами по исправлению.

.NET C# - мощный язык программирования, позволяющий создавать как локальные клиентские приложения для ПК и мобильных устройств, так и программные серверы. В данной работе затронута возможность с удобством и быстротой создавать программы для компьютера с Windows, работающие с реестром и файловой системой этой ОС.

В нашем случае речь идёт об анализе системы на уязвимости безопасности и одной из важных задач является поиск настроек Windows – они хранятся в определённых областях реестра, к которым разрабатываемая программа должна получать доступ, читать значения ключей, проверять их на корректность и сообщать о результатах пользователю.

Функционал программы не ограничивается поиском настроек в реестре и включает в себя другие способы проверки безопасности ОС.

Исходный код программы доступен по ссылке: https://github.com/Hockwell/windows_SecurityAdvisor. Системные требования: Windows 10 (тестировалось на Windows 10 Home и Pro 1809, но частично должно работать и на др. версиях Windows), .NET Framework 4.7. Для работы программы не требуются права администратора.

Важно отметить следующие проблемы совместимости: от версии ОС к версии (от редакции к редакции, от билда Windows 10 к билду) Microsoft может изменять место хранения некоторых настроек и алгоритмы работы системы в целом. Следовательно, не просто создать программу, учитывающую все различия – требуется долгий период отладки на разных версиях, чего не проводилось для данной работы. Очевидно, разумнее самой ОС проводить самодиагностику, что реализуется в Windows 10 со временем. В связи с последним трендом сторонние программы диагностики теряют популярность.

ОСНОВНАЯ ЧАСТЬ

1 Структура программы

Использовался фреймворк WPF, однако лишь для одного – для создания простого окна запуска, которое показано на рисунке 1 (в программе 2 основных потока, один из которых управляет интерфейсом, поэтому интерфейс не зависает при работе программы и сообщает индикатором прогресса, что программа не закончила свою работу). Отчёт о работе сохраняется в txt - файл, пример которого показан на рисунке 2.

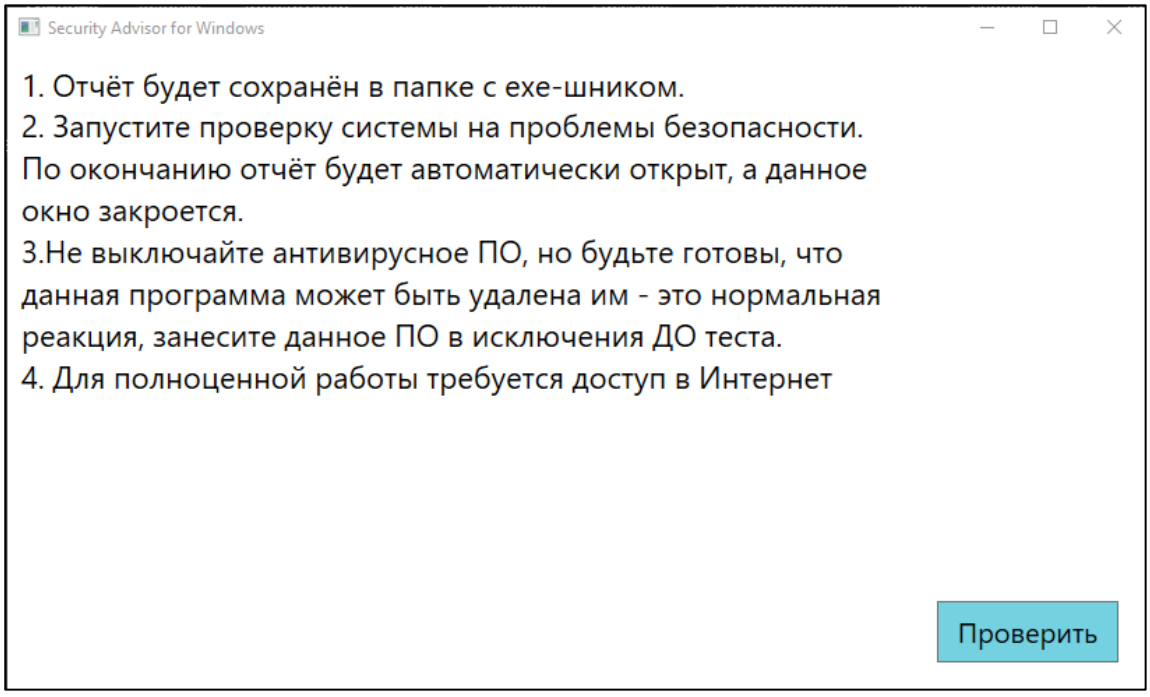


Рисунок 1 – Интерфейс программы

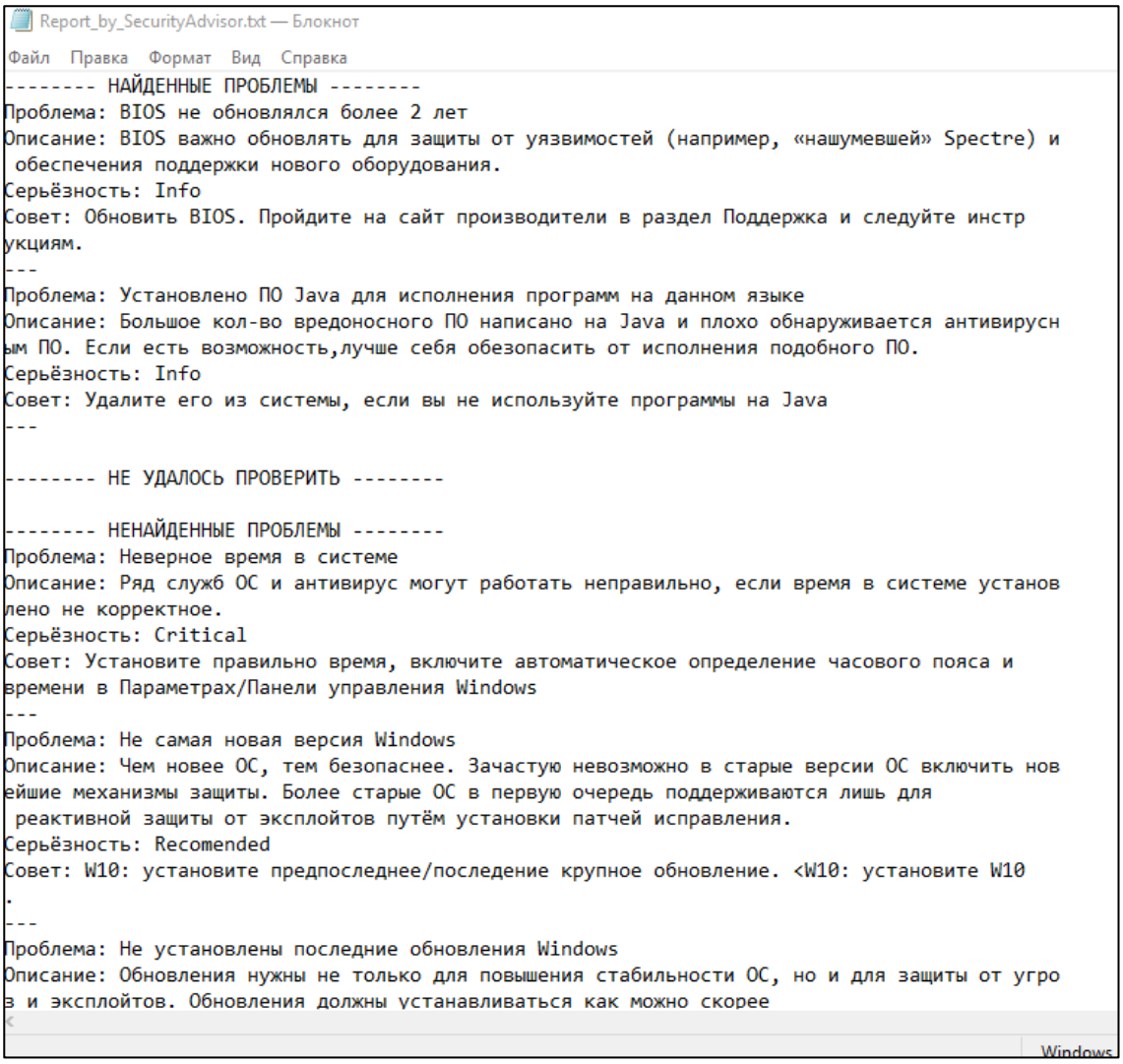


Рисунок 2 – Отчёт программы о проблемах безопасности

1.1 Особенности

Архитектура модульная и удобна тем, что если понадобится добавить на обнаружение новый фактор безопасности, то для этого нужно лишь создать класс-наследник от DetectionTechnique.cs (или его «абстрактных» наследников), в котором обязательно итог работы должен быть отражён в поле Status (ошибка, проблема найдена, не найдена) и добавить в Singleton-классе DB.cs в список problems очередную запись с следующими стандартными полями (например, если речь идёт об обнаружении работоспособности антивируса):

- Name = "Антивирус не защищает ОС",
- AdviceForUser = "Проверьте установлен ли антивирус, включён ли он",

- Raiting = ProblemRaiting.Critical,
- Description = "Антивирус обязательно должен быть установлен для предотвращения повреждения и кражи данных",
- Detection = new AntivirusDT().

Корректный вывод в отчёт и обработка будут выполнены автоматически.

1.2 Архитектура

На рисунке 1 показана диаграмма классов, которые непосредственно анализируют систему на факторы безопасности.

На рисунке 2 показаны классы, которые обеспечивали работу «техник детектирования» – так называются классы-наследники класса DetectionTechnique.

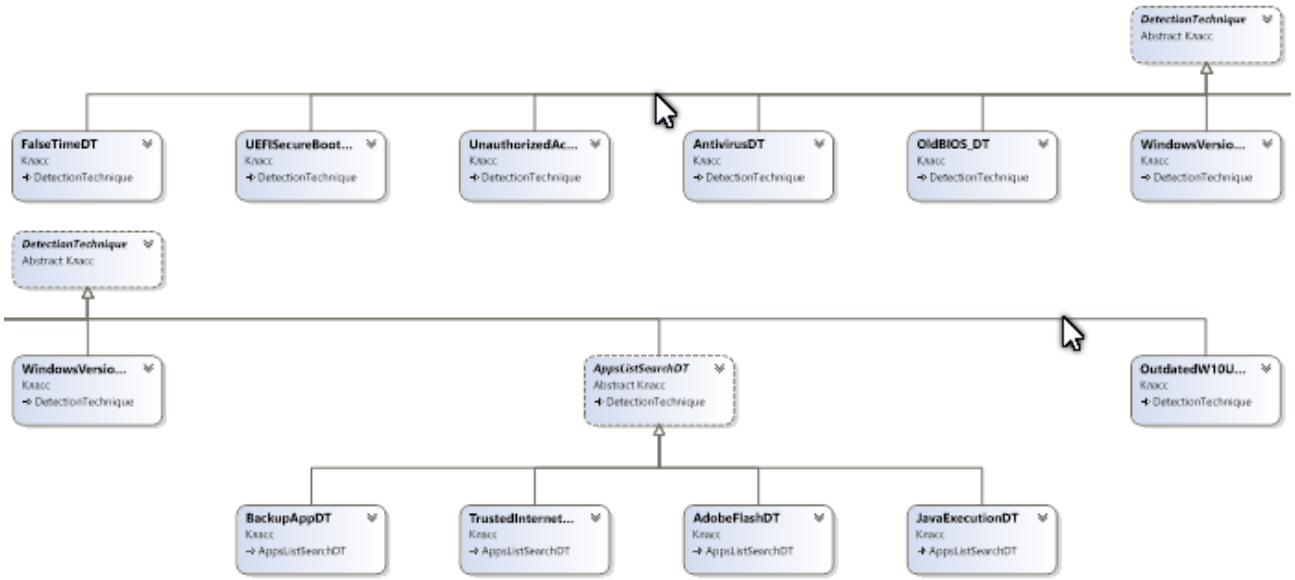


Рисунок 3 – Классы наследники от класса DetectionTechnique (сверху левая часть диаграммы, снизу правая)

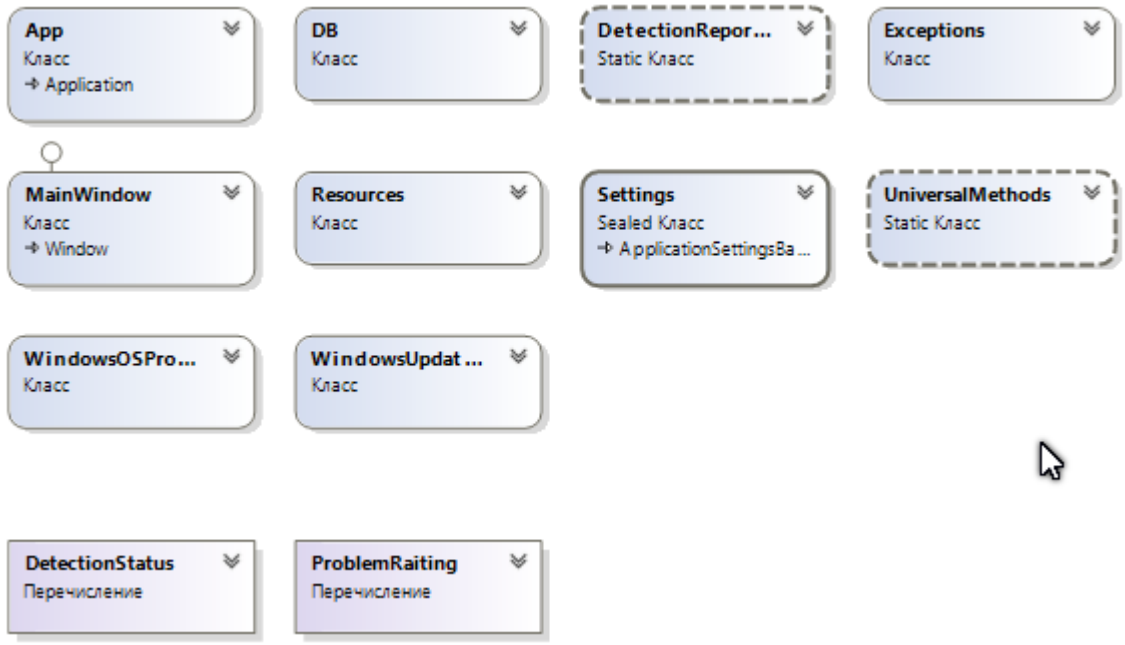


Рисунок 4 – Классы, не связанные непосредственно с обнаружением факторов безопасности

1.3 Обработка исключительных ситуаций

Особенно стоит отметить меры, принятые для поддержания стабильности и информативности программы в исключительных ситуациях. Пользователь получит информацию только о тех исключительных ситуациях, которые останавливает дальнейшую работу программы. При этом могут возникать и многие другие ситуации, которые будут без уведомлений обработаны программой и затронут ровно тот процесс работы, к которому относятся. Так, например, если доступа в Интернет не будет, программа не будет использовать техники, зависящие от Интернета и в отчёте они будут отмечены как не найденные по ошибке. В другом случае какая-то техника может обратиться в реестр и не найдёт там нужное ей значение или ветку, в таком случае это затронет только данную технику, другие техники продолжат свою работу.

При модернизации программа будет сообщать более подробную, чем обычно (если не заниматься педантичной обработкой исключений), информацию о том, что серьёзно нарушает её работоспособность.

Поскольку в программе используется парсер модели DOM сайта, то было важно в том числе попытаться сообщить разработчику, что парсер больше не актуален, ибо не верно функционирует – определить это можно не только по исключениям, которые возникают в стандартных случаях (например, парсер пытается в целочисленную переменную записать строку), но и по анализу адекватности той информации, которые он распознает (например, если он обнаружил на сайте сводки обновлений версию Windows 10 под номером 1807 как самую последнюю, но в анализируемой системе установлена версия 1809, то это, очевидно, неверная работа парсинга, либо опечатка на сайте).

2 Обнаруживаемые проблемы

В данном разделе перечисляются отслеживаемые факторы безопасности, а также обоснование зачем каждый из них обнаруживать.

2.1 Антивирус не установлен или не функционирует

Серьёзность: критично.

Помещаем тестовый стандартизированный (AMTSO) файл, который должен обнаруживаться всеми антивирусами мира (международное соглашение) – «eicar.txt» (содержит ключевую строку символов, на которую реагирует сигнатурный детектор) и проверяем его наличие в системе в течение определённого времени.

Если файл пропадает, значит он был обнаружен и удалён антивирусом.

Если файл остаётся на месте, возможны следующие варианты, о которых будет рассказано пользователю (а он, в свою очередь, должен проверить эти случаи):

- антивирус не работоспособен,
- антивирус не установлен.

2.2 Отсутствует система резервного копирования

Серьёзность: рекомендуется исправить.

Программа запрашивает список установленных программ и ищет известные ей программы резервного копирования. Так же, она проверяет работоспособность служб, отвечающих в Windows за резервное копирование (в Windows 7 – «Резервное копирование», в более поздних будет проверена ещё и функция «История файлов»). Если она не находит их, предупреждает пользователя.

Программа анализирует список установленного ПО, который хранится в реестре в разделе: HKEY_LOCAL_MACHINE SOFTWARE\ Microsoft\Windows\CurrentVersion\Uninstall. Алгоритм поиска построен таким образом, чтобы имена исполняемых файлов и версия не влияли на качество поиска – поиск по ключевым словам.

2.3 Не установлено последнее обновление для текущей версии Windows 10

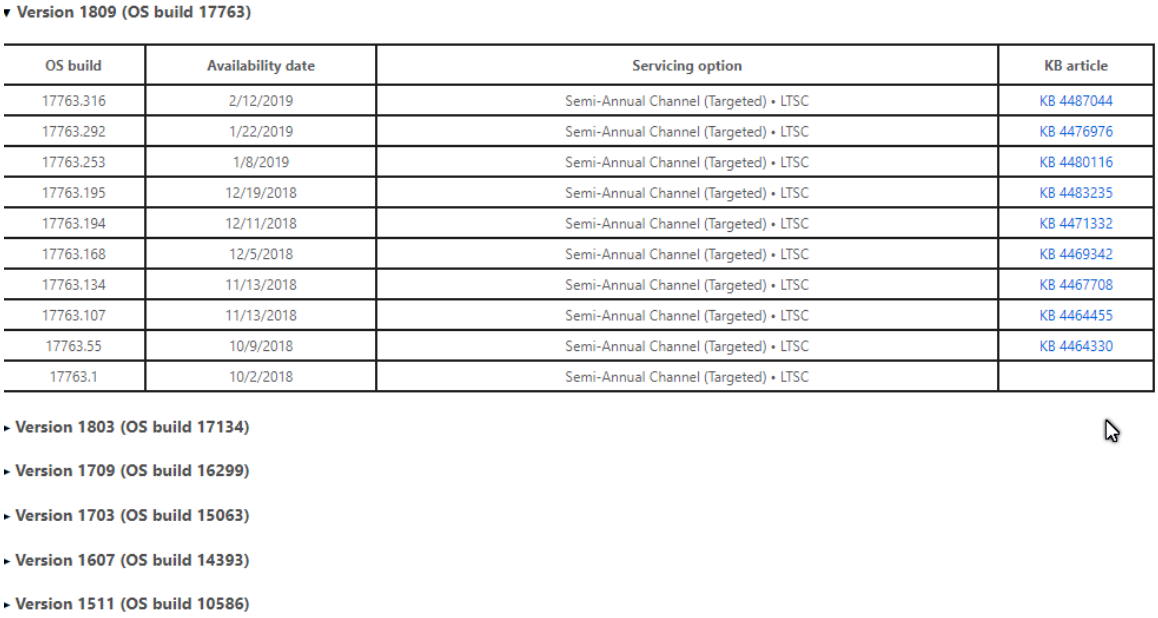
Серьёзность: критично.

Проблема заключается в том, что обновления безопасности нужно устанавливать как можно скорее и любое промедление может оказаться фатальным. Последний нашумевший случай – заражение ransomware WannaCry миллионов компьютеров организаций и обычных пользователей и это при том, что Microsoft заранее выпустила обновления безопасности, не позволявшее данному зловредному ПО проникнуть в систему по самому популярному вектору атаки – через уязвимую сетевую службу SMB. По последней статистике (2019) лишь 3% эксплойтов, используемых злоумышленниками, блокируются установкой таких обновлений, остальные либо в статусе 0day, либо известны, но обновления безопасности ещё не были выпущены. Однако это не причина игнорировать риск, поскольку при низкой вероятности возникновения проблемы из-за этого может быть крайне велик ущерб.

Напрямую в реестре не хранятся данные об дате обновлений и статусе службы, ровно, как и Power Shell-скриптом достать эту информацию нельзя на практике, хотя можно в теории. Для Windows 10 был избран более сложный подход: есть официальный сайт, фиксирующий обновления W10, где указаны

актуальные билды Windows 10. Билд – это версия ОС как проекта, малейшее изменение и билд меняется. Сложность в том, что сайт не имеет публичного API и добывать информацию нужно путём разбора непосредственно dom-а страницы, что программа и делает. В реестре программа должна найти два ключа по пути HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion – UBR (дробную часть билда – суффикс билда, она меняется при каждом обновлении Windows 10) и CurrentBuildNumber (целая часть билда – это альтернативная версия Windows 10, меняющаяся только при крупных обновлениях).

Таким образом, программа сравнивает основной номер билда и суффикс с последним билдом на сайте. Стоит заметить, что на сайте есть несколько таблиц выпуска обновлений, в каждой из которых написаны билды. Каждая таблица отвечает за свою версию Windows 10 (версия определяется «крупным» обновлением). Пример такой таблицы показан на рисунке 4. Очевидно, что достаточно не просто определить какой билд программа должна запомнить – запоминается самый верхний билд (таблицы отсортированы в порядке убывания актуальности билда) той таблицы, которая отвечает за текущую версию ОС Windows 10.



▾ **Version 1809 (OS build 17763)**

OS build	Availability date	Servicing option	KB article
17763.316	2/12/2019	Semi-Annual Channel (Targeted) • LTSC	KB 4487044
17763.292	1/22/2019	Semi-Annual Channel (Targeted) • LTSC	KB 4476976
17763.253	1/8/2019	Semi-Annual Channel (Targeted) • LTSC	KB 4480116
17763.195	12/19/2018	Semi-Annual Channel (Targeted) • LTSC	KB 4483235
17763.194	12/11/2018	Semi-Annual Channel (Targeted) • LTSC	KB 4471332
17763.168	12/5/2018	Semi-Annual Channel (Targeted) • LTSC	KB 4469342
17763.134	11/13/2018	Semi-Annual Channel (Targeted) • LTSC	KB 4467708
17763.107	11/13/2018	Semi-Annual Channel (Targeted) • LTSC	KB 4464455
17763.55	10/9/2018	Semi-Annual Channel (Targeted) • LTSC	KB 4464330
17763.1	10/2/2018	Semi-Annual Channel (Targeted) • LTSC	

▸ **Version 1803 (OS build 17134)**

▸ **Version 1709 (OS build 16299)**

▸ **Version 1703 (OS build 15063)**

▸ **Version 1607 (OS build 14393)**

▸ **Version 1511 (OS build 10586)**

Рисунок 5 – Таблицы билдов Windows 10

2.4 Не самая безопасная версия ОС

Серьёзность: рекомендуется исправить.

Чем новее ОС – тем безопаснее. Зачастую невозможно в старые версии ОС включить новейшие механизмы защиты. Более старые ОС в первую очередь поддерживаются лишь для реактивной защиты от эксплойтов путём установки патчей исправления.

Программа определяет основную версию ОС и если она <10, то сообщает об этом. Для версий Windows 10 логика другая: допускается, что пользователь использует последнее, либо предпоследнее крупное обновление (версию). Программа запрашивает данные со специального официального сайта, где нет публичного API, но данные можно распознать с html-страницы, используя dom-парсинг, как в случае с техникой 2.3. Пример таблицы, которая парсится, показан на рисунке 5:

Windows 10 current versions by servicing option

Last Updated: 2/19/2019

Version	Servicing option	Availability date	OS build	Latest revision date	
1809	Semi-Annual Channel (Targeted)	11/13/2018	17763.316	2/12/2019	*Microsoft recommends*
1809	Long-Term Servicing Channel (LTSC)	11/13/2018	17763.316	2/12/2019	
1803	Semi-Annual Channel	7/10/2018	17134.619	2/19/2019	
1803	Semi-Annual Channel (Targeted)	4/30/2018	17134.619	2/19/2019	
1709	Semi-Annual Channel	1/18/2018	16299.1004	2/19/2019	
1709	Semi-Annual Channel (Targeted)	10/17/2017	16299.1004	2/19/2019	
1703	Semi-Annual Channel	7/11/2017	15063.1659	2/19/2019	
1703	Current Branch (CB)	4/11/2017	15063.1659	2/19/2019	
1607	Current Branch for Business (CBB)	11/29/2016	14393.2828	2/19/2019	
1607	Current Branch (CB)	8/2/2016	14393.2828	2/19/2019	
1607	Long-Term Servicing Branch (LTSB)	8/2/2016	14393.2828	2/19/2019	
1507 (RTM)	Long-Term Servicing Branch (LTSB)	7/29/2015	10240.18132	2/12/2019	

Рисунок 6 – Таблица версий Windows 10 на Microsoft

Как можно заметить, версии могут повторяться, в то время как будут отличаться «ветки» обновления. Программа запоминает из этой таблицы первые 2 разных версии, поскольку в таблице версии отсортированы в порядке убывания актуальности. Если брать за основу таблицу на рисунке 5, то программа запоминает как наиболее безопасные версии 1809 и 1803. Если на компьютере

пользователя будет система, которая будет меньше по номеру обоих версий, то проблема будет считаться обнаруженной. Если версия в системе равна одной из двух наиболее безопасных, то проблема считается не найденной. Однако, если версия в системе больше обоих наиболее актуальных версий с сайта, либо находится между ними по значению, то это исключительная ситуация, говорящая либо о неактуальности сайта, либо о таком изменении его структуры, что парсер не справляется с нововведениями.

Поиск данных о версии ОС на текущем устройстве идёт с использованием ветки HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion. В ней есть такие интересующие нас ключи, как ReleaseId и ProductName.

Актуальность сайта сводок обновлений Microsoft проверяется по полю «Last Updated» в нём. Если сайт не обновлялся больше полугода, то он считается не актуальным и техника 2.3 перестаёт работать совсем, а техника 2.4 будет детектировать лишь наличие более старых версий Windows, нежели 10.

2.5 Не установлен безопасный браузер

Серьёзность: рекомендуется исправить.

Один из следующих браузеров должен быть в системе для надёжной защиты системы: Google Chrome, Opera, Chromium, Edge, FireFox, Яндекс браузер. Данные браузеры постоянно обновляются и имеют слаженный оперативный процесс поиска и исправления уязвимостей.

Программа анализирует список установленного ПО.

2.6 Неверное время в системе

Серьёзность: критично.

Ряд служб ОС и антивирус могут работать неправильно, если время в системе установлено не корректное.

Программа запросит в Интернете с соответствующего независимые данные о текущем времени. Полученные данные выглядят примерно так:

mycallback({"$id":"1","currentDateTime":"2019-02-20T17:46+01:00","utcOffset":"01:00:00","isDayLightSavingsTime":false,"dayOfTheWeek":"Wednesday","timeZoneName":"Central Europe Standard Time","currentFileTime":131951583653623718,"ordinalDate":"2019-51","serviceResponse":null}). Часовой пояс не нужен для этого, достаточно будет отслеживать разницу с тем временем, что есть в системе и тем временем, что получено из Интернета, между ними не должно быть более 12 часов (Земля поделена на 24 часовых пояса). Если в системе время не верное, то для остальных операций используется полученное из Интернета время.

2.7 Установлено ПО Java для запуска программ на данном языке

Серьёзность: для информации.

Большое кол-во вредоносного ПО написано на Java и плохо обнаруживается антивирусным ПО. Если есть возможность, лучше себя обезопасить от исполнения подобного ПО.

Программа анализирует список установленного ПО.

2.8 Не работает защита системных файлов от НСД

Серьёзность: критично.

ОС должна контролировать, как и куда программы получают доступ, в частности критические области ОС должны быть как можно лучше изолированы от НСД посторонних программ. Наличие такой проблемы говорит о том, что нарушена целостность ОС, возможно вследствие пиратской модификации - пиратское ПО нарушает целостность лицензионного аналога и опасно внедрением шпионского ПО.

Программа пытается разместить в папке System32 вложенную папку. При получении отказа сообщает об отсутствии проблемы.

2.9 Технология UEFI secure boot не включена

Серьёзность: рекомендуется исправить.

Программа проверяет в ветке HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Control\SecureBoot\State ключ UEFISecureBootEnabled = 1 – безопасное значение.

Технология осуществляет контроль подписей образа загружаемой ОС и драйверов. Если образ будет модифицирован вредоносным ПО (буткит, руткит), то он не загрузится, поскольку вредоносный драйвер будет заблокирован. Предотвращается загрузка как неподписанных данных, так и данных из чёрного списка.

2.10 BIOS не обновлялся более 2 лет

Серьёзность: для информации.

Назначить более серьёзную критичность мешает тот факт, что производители парой банально не выпускают обновления BIOS по истечении короткого срока поддержки (2-3 года).

Программа проверяет в ветке HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ SystemInformation ключ BIOSReleaseDate.

BIOS важно обновлять для защиты от уязвимостей (например, «нашумевшей» Spectre) и обеспечения поддержки нового оборудования.

2.11 Установлен AdobeFlash

Серьёзность: для информации.

Данное ПО позволяет воспроизводить веб-медиа-контент, однако часто используется злоумышленниками для совершения зловредных действий в системе минуя браузер. Менее 4% сайтов сегодня используют Flash, а браузеры отказываются воспроизводить соответствующий контент с помощью Flash по умолчанию во избежание проблем. На всякий случай лучше удалить данное ПО из системы, тем более, что оно не имеет смысла, однако может стать причиной атаки.

Программа анализирует список установленного ПО.

ЗАКЛЮЧЕНИЕ

Цели и задачи курсовой работы выполнены. Была создана программа, способная проанализировать ОС Windows на проблемы безопасности и небезопасные настройки.

Были рассмотрены такие программные способы анализа ОС Windows, как чтение ключей реестра, размещение тестовых файлов и размещение каталогов в защищённых системных областях.

Весь основной исходный код программы указан в Приложении А.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Полное руководство по платформе .NET 4.7 // METANIT.RU: онлайн-справочник. URL: https://metanit.com/sharp/ (дата обращения: 20.12.2018).

2 В. Ф. Шаньгин. Защита информации в компьютерных системах и сетях. – ДМК Пресс, 2012. – 593 с.

3 Джозеф Албахари, Бен Албахари. Справочник по C# 7.0. Полное описание языка. – Вильямс, 2016. – 1024 с.

4 Pavel Yosifovich, Mark E. Russinovich. Windows Internals: Part 1: System Architecture, Processes, Threads, Memory Management, and More. – Microsoft Press, 2017. – 800 pp.

ПРИЛОЖЕНИЕ А

Исходный код программы

Листинг 1 – MainWindows.xaml.cs

```
using SecurityAdvisor.Infrastructure.FileExport;
using SecurityAdvisor.Infrastructure.Generic;
using SecurityAdvisor.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows;

namespace SecurityAdvisor
{
    public partial class MainWindow : Window
    {
        private delegate void DelegateForAsync();

        private DB db;
        private List<WindowsOSProblem> problems;

        public MainWindow()
        {
            InitializeComponent();
        }

        private void Check_Btn_Click(object sender, RoutedEventArgs e)
        {
            CleanupExcessFiles();

            Check_Btn.IsEnabled = false;
            progressBar.Visibility = Visibility.Visible;

            DelegateForAsync asyncJob = new DelegateForAsync(MainJob);
            asyncJob.BeginInvoke(null, null);


        }

        private void CleanupExcessFiles()
        {
            File.Delete(Exceptions.ERRORS_LOG_FILE_NAME);
        }

        private void MainJob()
        {
            try
            {
                LoadProblemsList();
                DetectProblems();
                DetectionReportTXTExport.SaveDetectionReport(problems);
                UpdateGUI();
            }
            catch (Exception e)
            {
                MessageBox.Show(e.Message, "Возникла ошибка", MessageBoxButton.OK,
MessageBoxImage.Warning);
                Exceptions.PrintInfoAboutException(e);
                CloseApp();
            }
        }
```

Продолжение листинга 1

```
private void LoadProblemsList()
        {
            db = DB.Load();
            problems = db.GetProblemsList();
        }

        private void DetectProblems()
        {
            if (!(problems[0].Detection is Infrastructure.Detection.FalseTimeDT))
                throw new Exception("FalseTimeDT должно содержаться в БД на 0 месте - его
инфа нужна остальным техникам и вспомогательным операциям");

            DetectProblem(0);

            try
            {
                db.GetLocalOSBuildAndVersionFromRegistry();
                db.GetLastOSBuildAndActualVersionFromInternet();
            }
            catch (Exception e)
            {
                Exceptions.PrintInfoAboutException(e);
            }

            for (int i = 1; i < problems.Count; i++)
            {
                DetectProblem(i);
            }
        }

        private void DetectProblem(int problemIndex)
        {
            try
            {
                problems[problemIndex].Detection.Execute();
            }
            catch (Exception e)
            {
                Exceptions.PrintInfoAboutException(e);
                problems[problemIndex].Detection.Status = DetectionStatus.Error;
            }
        }

        private void UpdateGUI()
        {
            Action action = () => { progressBar.Visibility = Visibility.Hidden;
Check_Btn.IsEnabled = true; CloseApp(); };
            Dispatcher.BeginInvoke(action);
        }

        private void CloseApp()
        {
            Environment.Exit(1);
        }
    }
}
```

Листинг 2 – MainWindows.xaml.cs

```
using SecurityAdvisor.Infrastructure;
using SecurityAdvisor.Infrastructure.Generic;

namespace SecurityAdvisor.Model
{
    public class WindowsOSProblem
    {
        public string Name { get; set; }
        public string Description { get; set; }
        public ProblemRaiting Raiting { get; set; }
        public DetectionTechnique Detection { get; set; }
        public string AdviceForUser { get; set; }
    }
}
```

Листинг 3 – WindowsUpdatesSiteProvider.cs

```
using CsQuery;
using System;
using System.IO;
using System.Net;
using System.Text.RegularExpressions;
using static SecurityAdvisor.Infrastructure.Generic.UniversalMethods;

namespace SecurityAdvisor.Infrastructure.Generic
{
    class WindowsUpdatesSiteProvider
    {
        private const string URI =
@"https://winreleaseinfoprod.blob.core.windows.net/winreleaseinfoprod/en-US.html";

        public CQ DOM { get; set; }
        private DB db = DB.Load();

        public bool GetSiteDOMAndCheckSiteRelevance()
        {
            try
            {
                GetPageInHTML();
                CheckSiteRelevance();
                return true;
            }
            catch (Exception e)
            {
                Exceptions.PrintInfoAboutException(e);
                return false;
            }
        }
```

Продолжение листинга 3

```
private bool GetPageInHTML()
        {
            string html;

            WebRequest request = WebRequest.Create(URI);
            request.Proxy = null;

            using (WebResponse response = request.GetResponse())
            using (Stream responseStream = response.GetResponseStream())
            using (StreamReader reader = new StreamReader(responseStream))
            {
                html = reader.ReadToEnd();
            }
            DOM = CQ.Create(html);

            return true;
        }

        private void CheckSiteRelevance()
        {
            int halfYear_hours = (365 / 2) * 24;
            string div = DOM["div"].Text();

            Match lastUpdateDateMatch = Regex.Match(div, @"\d{1,2}/\d{2}/\d{4}");
            if (!lastUpdateDateMatch.Success)
                throw new Exception("Регулярное выражение не обнаружило дату на сайте");
            string[] MM_dd_yyyy = lastUpdateDateMatch.Value.Split('/');
            DateTime lastUpdateDate = new DateTime(year: int.Parse(MM_dd_yyyy[2]), month:
int.Parse(MM_dd_yyyy[0]), day: int.Parse(MM_dd_yyyy[1]));

            DB db = DB.Load();
            if (db.IsActualTimeNotDetermined())
                throw new Exception("Время не было определено средствами FalseTimeDT");

            double timeDifference_hours = CalcTimesDifferenceInHours(lastUpdateDate,
db.ActualTime);
            if (timeDifference_hours >= halfYear_hours)
                throw new Exception("Сайт для проверки обновлённости ОС не актуален");
        }

    }
}
```

Листинг 4 – UniversalMethods.cs

```
using Microsoft.Win32;
using System;

namespace SecurityAdvisor.Infrastructure.Generic
{
    static class UniversalMethods
    {
        public static double CalcTimesDifferenceInHours(DateTime time1, DateTime time2)
        {
            return new TimeSpan(Math.Abs(time1.Ticks - time2.Ticks)).TotalHours;
        }

        public static object GetKeyValueFromRegistry(string path, string key)
        {
            object value;
            using (RegistryKey branch = Registry.LocalMachine.OpenSubKey(path))
            {
                value = branch.GetValue(key);
            }
            return value;
        }
    }
}
```

Листинг 5 – AdobeFlashDT.cs

```
using SecurityAdvisor.Infrastructure.Generic;
using System.Collections.Generic;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class AdobeFlashDT : AppsListSearchDT
    {
        protected override List<string[]> KeywordGroupsForSearch => new List<string[]>()
        {
            new string[] {"Flash", "Adobe"}
        };

        public override void Execute()
        {
            Status = IsAtLeastOneAppInstalledAlready() ? DetectionStatus.Found :
DetectionStatus.NotFound;
        }
    }
}
```

Листинг 6 – AntivirusDT.cs

```
using SecurityAdvisor.Infrastructure.Generic;
using System;
using System.IO;
using System.Text;
using System.Threading;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class AntivirusDT : DetectionTechnique
    {
        #region Consts
        private const string EICAR_STRING = @"X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-
ANTIVIRUS-TEST-FILE!$H+H*";
        private const string TEST_FILE_NAME = "eicar.txt";
        #endregion

        public override void Execute()
        {
            try
            {
                PlaceTestFile();
                CheckFileAvailability();
            }
            catch (UnauthorizedAccessException)
            {
                Status = DetectionStatus.NotFound;
            }
        }

        private void PlaceTestFile()
        {
            File.WriteAllText(TEST_FILE_NAME, EICAR_STRING, Encoding.Default);
        }

        private void CheckFileAvailability()
        {
            int loop = 0;
            int loopMax = 3;

            do
            {
                Thread.Sleep(3000);
                loop ++;
            }
            while (IsUndetectedByAntivirus() && loop < loopMax);

            Status = (loop == loopMax) ? DetectionStatus.Found : DetectionStatus.NotFound;

            bool IsUndetectedByAntivirus() => File.Exists(TEST_FILE_NAME) && new
FileInfo(TEST_FILE_NAME).Length != 0;
        }
    }
}
```

Листинг 7 – AppsListSearchDT.cs

```
using Microsoft.Win32;
using SecurityAdvisor.Infrastructure.Generic;
using System;
using System.Collections.Generic;

namespace SecurityAdvisor.Infrastructure.Detection
{
    abstract class AppsListSearchDT : DetectionTechnique
    {
        private const string APP_NAME_BY_ERROR = "-- Error --";
        protected abstract List<string[]> KeywordGroupsForSearch { get; }

        public override void Execute()
        {
            Status = IsAtLeastOneAppInstalledAlready() ? DetectionStatus.NotFound :
DetectionStatus.Found;
        }

        public static List<string> GetInstalledAppNamesList()
        {
            List<string> appsNames = new List<string>();

            string uninstallKey = @"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall";
            using (RegistryKey rk = Registry.LocalMachine.OpenSubKey(uninstallKey))
            {
                foreach (string skName in rk.GetSubKeyNames())
                {
                    using (RegistryKey sk = rk.OpenSubKey(skName))
                    {
                        try
                        {
                            string displayName = sk.GetValue("DisplayName") as string;
                            appsNames.Add(displayName);
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine(ex.StackTrace);
                            appsNames.Add(APP_NAME_BY_ERROR);
                        }
                    }
                }
            }

            return appsNames;
        }

        protected bool IsAtLeastOneAppInstalledAlready()
        {
            List<string> installedAppsNames = DB.Load().GetInstalledProgramsList();

            if (installedAppsNames == null)
            {
                throw new Exception("installedAppsNames is null");
            }

            int keywordsCounter = 0;

            foreach (string appName in installedAppsNames)
            {
                if (IsBadAppName())
                    continue;
```

Продолжение листинга 7

```
                foreach (string[] keywordsGroup in KeywordGroupsForSearch)
                {
                    keywordsCounter = 0;
                    for (int i = 0; i < keywordsGroup.Length; i++)
                    {
                        if (appName.IndexOf(keywordsGroup[i],
StringComparison.CurrentCultureIgnoreCase) == -1)
                            break;
                        else
                            keywordsCounter++;
                    }
                    if (IsAllKeywordsGroupContainedInAppNameString())
                        return true;

                    bool IsAllKeywordsGroupContainedInAppNameString() => keywordsCounter
== keywordsGroup.Length;
                }

                bool IsBadAppName() => appName == null || appName == APP_NAME_BY_ERROR;
            }
            return false;
        }
    }
}
```

Листинг 8 –WindowsVersionTooOldDT.cs

```
using SecurityAdvisor.Infrastructure.Generic;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class WindowsVersionTooOldDT : DetectionTechnique
    {
        public const string W10_VERSION_REGEX_PATTERN = @"\s\d{4}\s";
        public const int W10_UPDATES_BRANCHES_AMOUNT = 3;
        public const int MOST_SECURE_W10_VERSIONS_AMOUNT = 2;
        private static readonly float[] SUPPORTED_OLD_OS_VERSIONS = new float[] { 7, 8,
8.1f };
        private static readonly List<int> MOST_SECURE_W10_VERSION_NULL_VALUE = null;
        private List<int> MostSecureW10Versions { get; set; } =
MOST_SECURE_W10_VERSION_NULL_VALUE;
        private DB db;

        public override void Execute()
        {
            db = DB.Load();

            if (db.LocalOSVersion == DB.LOCAL_OS_VERSION_NULL_VALUE)
                throw new Exception("Билд системы не был получен");
```

Продолжение листинга 8

```
if (IsNotW10OnComputer())
            {
                Status = DetectionStatus.Found;
                return;
            }

            if (db.WindowsUpdatesSite_DOM == DB.WINDOWS_UPDATES_SITE_DOM_NULL_VALUE)
                throw new Exception(Exceptions.MSG_DOM_NOT_INIT_IN_DB);

            ParseW10ActualVersions();

            if (IsActualW10OnComputer())
            {
                Status = DetectionStatus.NotFound;
            }
            else if (IsNotActualW10OnComputer())
            {
                Status = DetectionStatus.Found;
            }
            else
            {
                throw new Exception(Exceptions.MSG_PARSING_FAILED_OR_BAD_SITE);
            }

            bool IsActualW10OnComputer() =>
MostSecureW10Versions.Contains((int)db.LocalOSVersion);
            bool IsNotActualW10OnComputer() => db.LocalOSVersion <
MostSecureW10Versions.Min();
        }

        public static bool IsNotW10OnComputer() =>
SUPPORTED_OLD_OS_VERSIONS.Contains(DB.Load().LocalOSVersion);

        private void ParseW10ActualVersions()
        {
            string versionsTable = db.WindowsUpdatesSite_DOM["div > table >
tbody"].Eq(0).Text();
            Match w10VersionMatch = Regex.Match(versionsTable, W10_VERSION_REGEX_PATTERN);
            if (!w10VersionMatch.Success)
                throw new Exception(Exceptions.MSG_W10_VERSION_NOT_PARSED);
            MostSecureW10Versions = new List<int>(MOST_SECURE_W10_VERSIONS_AMOUNT);

            int lastVersion = int.Parse(w10VersionMatch.Value);
            MostSecureW10Versions.Add(lastVersion);

            FindAndParsePenultimateVersion(w10VersionMatch, lastVersion);
        }

        private void FindAndParsePenultimateVersion(Match w10VersionMatch, int
lastVersion)
        {
            for (int i = 0; i < (MOST_SECURE_W10_VERSIONS_AMOUNT - 1) *
W10_UPDATES_BRANCHES_AMOUNT; i++)
            {
                w10VersionMatch = w10VersionMatch.NextMatch();
                if (!w10VersionMatch.Success)
                    throw new Exception(Exceptions.MSG_W10_VERSION_NOT_PARSED);
                int penultimateVersion = int.Parse(w10VersionMatch.Value);
                if (lastVersion < penultimateVersion)
                {
                    throw new Exception("Не верно определены версии W10.");
                }
```

Листинг 9 –BackupAppDT.cs

```
using System.Collections.Generic;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class BackupAppDT : AppsListSearchDT
    {
        protected override List<string[]> KeywordGroupsForSearch => new List<string[]>()
        {
            new string[] {"True", "Image" },
            new string[] {"Kaspersky", "Total", "Security" },
            new string[] {"Backup"},
            new string[] {"Macrium", "Reflect" }
        };
    }
}
```

Листинг 10 – DetectionTechnique.cs

```
using SecurityAdvisor.Infrastructure.Generic;

namespace SecurityAdvisor.Infrastructure
{
    public abstract class DetectionTechnique
    {
        public DetectionStatus Status { get; set; } = DetectionStatus.Error;

        public abstract void Execute();
    }
}
```

Листинг 11 – FalseTimeDT.cs

```
using SecurityAdvisor.Infrastructure.Generic;
using System;
using System.Net;
using System.Text.RegularExpressions;
using static SecurityAdvisor.Infrastructure.Generic.UniversalMethods;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class FalseTimeDT : DetectionTechnique
    {
        private const int HOURS_AMOUNT_FOR_TIMES_DIFFERENCE = 12;
        private const string URI_WITH_TIME =
@"http://worldclockapi.com/api/jsonp/cet/now?callback=mycallback";

        public override void Execute()
        {
            DB db = DB.Load();
            DateTime timeInOS = DateTime.Now;
            DateTime onlineTime;

            try
            {
                onlineTime = GetTimeFromInternet();
            }
            catch (WebException e)
            {
```

Продолжение листинга 11

```
                db.ActualTime = timeInOS;
                Status = DetectionStatus.Error;
                Exceptions.PrintInfoAboutException(e);
                return;
            }

            double timesDeltaInDays = CalcTimesDifferenceInHours(timeInOS, onlineTime);
            if (IsDifferentTimes())
            {
                Status = DetectionStatus.Found;
                db.ActualTime = onlineTime;
            }
            else
            {
                Status = DetectionStatus.NotFound;
                db.ActualTime = timeInOS;
            }

            bool IsDifferentTimes() => timesDeltaInDays >
HOURS_AMOUNT_FOR_TIMES_DIFFERENCE;

        }

        private DateTime GetTimeFromInternet()
        {
            WebClient web = new WebClient { Proxy = null };

            string json = web.DownloadString(URI_WITH_TIME);
            Match onlineTimeMatch = Regex.Match(json, @"\d{4}-\d{2}-\d{2}T\d{2}:\d{2}");
            if (!onlineTimeMatch.Success)
                throw new Exception("Регулярное выражение не обнаружило дату и время в
json-e");
            string onlineTime = onlineTimeMatch.Value.Replace('T',' ');
            DateTime onlineTime_DT = DateTime.ParseExact(onlineTime, "yyyy-MM-dd HH:mm",
null);

            return onlineTime_DT;

        }

    }
}
```

Листинг 12 – OutdatedW10UpdatesDT.cs

```
using SecurityAdvisor.Infrastructure.Generic;
using System;
using System.Text.RegularExpressions;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class OutdatedW10UpdatesDT : DetectionTechnique
    {
        public const int W10_ACTUAL_VERSIONS_AMOUNT = 4;
        private const float LAST_W10_BUILD_NULL_VALUE = -2;
        private float LastW10Build { get; set; } = LAST_W10_BUILD_NULL_VALUE;
        private DB db;
```

Продолжение листинга 12

```
public override void Execute()
        {
            db = DB.Load();

            if (db.WindowsUpdatesSite_DOM == DB.WINDOWS_UPDATES_SITE_DOM_NULL_VALUE)
                throw new Exception(Exceptions.MSG_DOM_NOT_INIT_IN_DB);
            if (db.LocalOSBuild == DB.LOCAL_OS_BUILD_NULL_VALUE)
                throw new Exception("Билд системы не был получен");
            if (WindowsVersionTooOldDT.IsNotW10OnComputer())
                throw new Exception("Данная система не является Windows 10 и проверить
актуальность обновлений невозможно");

            double lastBuild = ParseW10LastBuild();
            if (lastBuild == db.LocalOSBuild)
                Status = DetectionStatus.NotFound;
            else if (lastBuild > db.LocalOSBuild)
                Status = DetectionStatus.Found;
            else
                throw new Exception(Exceptions.MSG_PARSING_FAILED_OR_BAD_SITE);
        }
        private double ParseW10LastBuild()
        {
            int h4Counter = 0;
            bool isW10VersionFound = false;
            for (;h4Counter < W10_ACTUAL_VERSIONS_AMOUNT; h4Counter ++)
            {
                string h4Content = db.WindowsUpdatesSite_DOM["div > h4 >
a"].Eq(h4Counter).Text();
                Match W10VersionMatch = Regex.Match(h4Content,
WindowsVersionTooOldDT.W10_VERSION_REGEX_PATTERN);
                if (isW10VersionFound = int.Parse(W10VersionMatch.Value) ==
db.LocalOSVersion)
                    break;
            }
            if (!isW10VersionFound)
                throw new Exception("Парсер не нашёл на сайте таблицу с билдами текущей
версии W10");

            int buildsTableIndex = h4Counter + 1; //см. h4Counter
            string buildsTable = db.WindowsUpdatesSite_DOM["div > table >
tbody"].Eq(buildsTableIndex).Text();
            Match lastBuild = Regex.Match(buildsTable, @"\d{5,7}.\d{3,5}\s");

            return double.Parse(lastBuild.Value.Replace('.', ','));
        }
    }
}
```

Листинг 13 – OldBIOS_DT.cs

```
using System.Globalization;
using static SecurityAdvisor.Infrastructure.Generic.UniversalMethods;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class OldBIOS_DT : DetectionTechnique
    {
        private const string PATH = @"SYSTEM\CurrentControlSet\Control\SystemInformation";
        private const string KEY = "BIOSReleaseDate";
        private const int TWO_YEARS = 365*2*24;

        public override void Execute()
        {
            DB db = DB.Load();

            if (db.IsActualTimeNotDetermined())
                throw new Exception("Для работы данной техники нужно запустить FalseTimeDT
раньше, а не позже!");

            string biosDate = (string) GetKeyValueFromRegistry(PATH, KEY);
            DateTime biosDate_DT = DateTime.ParseExact(biosDate, "MM/dd/yyyy",
CultureInfo.InvariantCulture);

            if (IsTrueDateDifference())
                Status = DetectionStatus.NotFound;
            else
                Status = DetectionStatus.Found;

            bool IsTrueDateDifference() => CalcTimesDifferenceInHours(db.ActualTime,
biosDate_DT) <= TWO_YEARS;
        }
    }
}
```

Листинг 14 – JavaExecutionDT.cs

```
using SecurityAdvisor.Infrastructure.Generic;
using System.Collections.Generic;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class JavaExecutionDT : AppsListSearchDT
    {
        protected override List<string[]> KeywordGroupsForSearch => new List<string[]>()
        {
            new string[] {"Java"}
        };

        public override void Execute()
        {
            Status = IsAtLeastOneAppInstalledAlready() ? DetectionStatus.Found :
DetectionStatus.NotFound;
        }
    }
}
```

Листинг 15 – TrustedInternetBrowserDT.cs

```
using SecurityAdvisor.Infrastructure.Generic;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class TrustedInternetBrowsersDT : AppsListSearchDT
    {
        protected override List<string[]> KeywordGroupsForSearch => new List<string[]>()
        {
            new string[] {"Chrome" },
            new string[] {"Opera" },
            new string[] {"Firefox" },
            new string[] {"Chromium" },
            new string[] {"Edge"},
            new string[] {"Yandex", "Browser"}
        };
    }
}
```

Листинг 16 – UEFISecureBootDT.cs

```
using static SecurityAdvisor.Infrastructure.Generic.UniversalMethods;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class UEFISecureBootDT : DetectionTechnique
    {
        private const string PATH = @"SYSTEM\CurrentControlSet\Control\SecureBoot\State";
        private const string KEY = "UEFISecureBootEnabled";
        private const int SECURE_VALUE = 1;

        public override void Execute()
        {
            int enabled = (int) GetKeyValueFromRegistry(PATH, KEY);

            if (IsSecureValue())
            {
                Status = Generic.DetectionStatus.NotFound;
            }
            else
            {
                Status = Generic.DetectionStatus.Found;
            }

            bool IsSecureValue() => enabled == SECURE_VALUE;
        }
    }
}
```

Листинг 17 – UnauthorizedAccessToOS_DT.cs

```
using System;
using System.IO;

namespace SecurityAdvisor.Infrastructure.Detection
{
    class UnauthorizedAccessToOS_DT : DetectionTechnique
    {
        private const string PATH_TO_BAD_FOLDER = @"C:\Windows\System32\bad";

        public override void Execute()
        {
            Status = CheckSystemFilesProtection() ? Generic.DetectionStatus.NotFound :
Generic.DetectionStatus.Found;
        }

        private bool CheckSystemFilesProtection()
        {
            try
            {
                Directory.CreateDirectory(PATH_TO_BAD_FOLDER);
                return false;
            }
            catch (UnauthorizedAccessException)
            {
                return true;
            }
        }
    }
}
```