

# Компьютерная безопасность

Захар Кириллов, MSc  
zahhar@gmail.com

Mainori Kõrgkool  
2010

# Цели и задачи

- Узнать о рисках и опасностях, сопутствующих передаче, обработке и хранению деликатных данных
- Получить представление о криптологии
- Научиться пользоваться техническими средствами для анализа угрозы безопасности и защиты компьютерных систем от несанкционированного доступа

# О чём речь?

- Криптография и криптология, исторические шифры, криптоалгоритмы и криптоанализ
- Защиты данных в сетях Ethernet и Wi-Fi
- Безопасность операционной системы и прикладного софта
- Безопасность и защита данных в интернете
- Корпоративные политики безопасности, их разработка, внедрение и контроль

# План работы

- 24 часа аудиторной работы: 6 встреч по 4ч
- 4 лекции + 2 практикума
  - Взлом Wi-Fi
  - Использование уязвимостей в прикладном софте (например, для получения контроля над удалённым компьютером)
- 2-3 домашние работы (+1 балл к экзамену)
- Экзамен (*“Профессор – лопух, но аппаратура при нём”*)

# Учебные материалы

- **Moodle** и **mainor.info** – конспект и полезные ссылки
- **Silence on the Wire**, Michal Zalewski
- **The Art of Software Security Assessment**, M. Dowd, J. McDonald, J. Schuh
- **Principles of information security**, Michael Whitman
- **Agressive network self-defence**, N. R. Wyler, B. Potter , C. Hurley
- Серия книг **Кевина Митника** (социальная инженерия)
- Серия книг **Stealing the Network**
- **Искусство войны** (древнекитайский трактат, V век до н.э.)

# Для самостоятельной работы

- У кого есть время и желание глубже разобраться в сути предмета, могут скачать, установить и поиграть с:
  - Дистр Линукса BackTrack 4 (LiveCD или USB)
  - Фреймворк Metasploit или w3af
  - Дистр Damn Vulnerable Linux
  - Приложение Damn Vulnerable Web App (DVWA) на PHP/mySQL (*устанавливайте в локальной “песочнице”, не ставьте на реальный веб-сервер!*)

# Глобальная проблема

- С каждым поколением кратно увеличивается объем приватной информации, которую мы хотим сохранить в секрете от посторонних (каждому есть что скрывать).
- Люди склонны потакать своему стремлению к удобству, простоте и скорости, а потому слабо заботятся о безопасности (обеспечение которой всегда усложняет и замедляет процессы + делает их дороже)
- Уровень преступности на почве безопасности растёт: множеству людей интересны наши секреты, потому что “Информация правит миром”.
- Последствия утечки или утраты чувствительной информации огромны, но размер ущерба сложно правильно оценить до того, как он будет причинён.

# Криптография

- Криптография = “*скрытопись*” (греч.)
- Наука о методах обеспечения конфиденциальности (невозможности прочтения информации посторонним) и аутентичности (целостности и подлинности авторства, а также невозможности отказа от авторства) информации.
- Суть криптографии – скрыть семантический смысл данных от посторонних лиц посредством изменения их формы (“зашифровать”)

# Криптоанализ

- Наука о методах получения исходного значения зашифрованной информации
- Оценка сильных и слабых сторон различных методов (алгоритмов) шифрования
- Поиск уязвимостей в криптографическом алгоритме или протоколе, “взлом” шифров

Криптология  
=  
Криптография + криптоанализ

*(термин появился в 1920-х годах)*

# Ещё ряд терминов

- **Открытый текст** (*исходный текст, plain text, avatekst*) – данные, хранимые или передаваемые в своём первоначальном виде (без шифрования)
- **Закрытый текст** (*шифровка, криптограмма, ciphertext, krüptogramm*) – результат работы криптосистемы над открытым текстом
- **Криптосистема** (*криптоалгоритм*) – совокупность обратимых преобразований открытого текста в зашифрованный

# Принцип работы криптосистемы 1/2

- Замена символов исходного текста на другие символы (*substitution*)
  - С использованием того же алфавита (A → C, B → D, C → E и т.д.)
  - С использованием символов других алфавитов или неалфавитных символов (“Пляшущие человечки” Конан Дойла, азбука Морзе, цветное кодирование и т.п.)

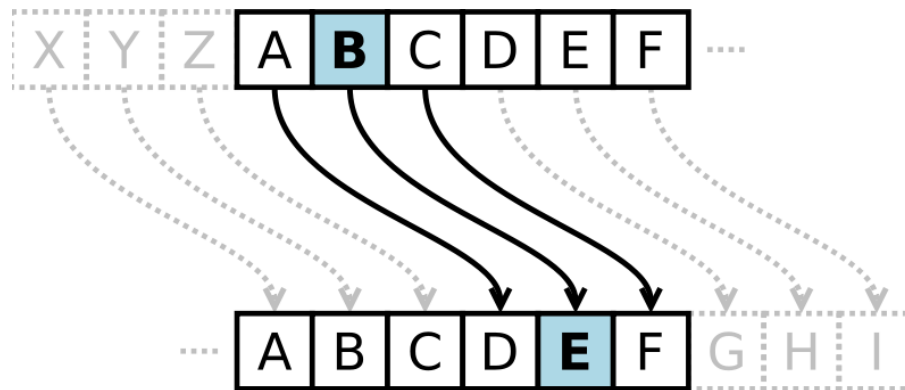


## Принцип работы криптосистемы 2/2

- Перестановка символов текста местами (*transposition*)
- ВСТРЕЧАЕМСЯ ЗАВТРА В 6 ВЕЧЕРА  
ВРАСАР6ЧА  
СЕЕЯВАВЕ  
ТЧМЗТВЕР
- ВРАСА Р6ЧАС ЕЕЯВА ВЕТЧМ ЗТВЕР

# Исторические криптоалгоритмы

- Шифр Цезаря



- Шифрование по книге (например по Библии)
- Магические квадраты
- Энигма

<http://enigmaco.de/enigma/enigma.html>

# Надёжность криптосистемы

- Для повышения сложности криптосистем в них одновременно применяется и принцип замены, и перестановки, причём в несколько “проходов” (16, 32, 64 и т.д.).
- При этом, криптосистема остаётся **ненадёжной (уязвимой)**, если её алгоритм чрезмерно сложен, а тайна шифрования сохраняется только до тех пор, пока сам алгоритм (или его нюансы) хранятся в секрете.

# Принцип Керкгоффа

- *“Система не должна требовать секретности, на случай, если она попадёт в руки врага”* или, проще (по Шеннону) – *“враг может знать систему”*
- Современные криптосистемы (в отличие от исторических) используют открытые алгоритмы (т.е. способ шифрования не держится в секрете и известен всем)
- Секретность шифровки обеспечивает **“ключ”** – параметр криптоалгоритма, от которого зависит конечный результат его работы.

# Шифрование при помощи ключа

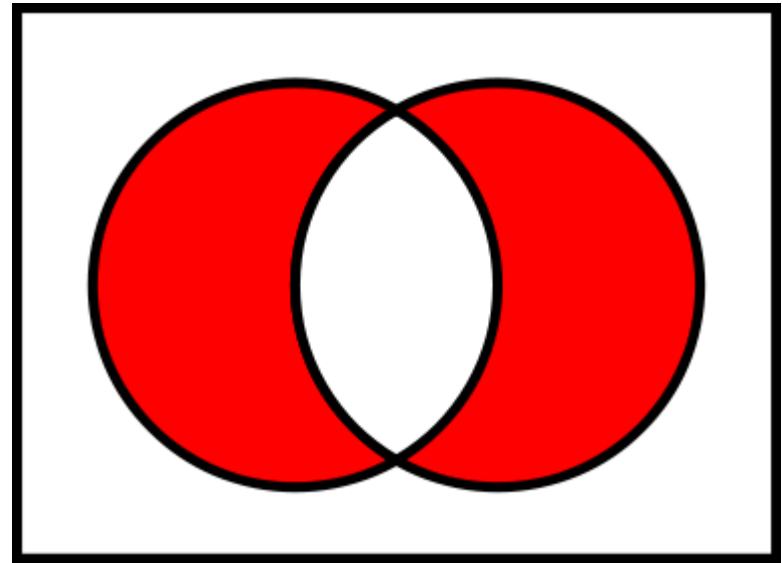
ВСТРЕЧАЕМСЯ ЗАВТРА В 6 ВЕЧЕРА  
КЛЮЧ КЛЮЧ КЛЮЧ КЛЮЧ КЛЮЧ КЛЮЧ

- Проводя логические или математические действия над каждой парой символов из открытого текста и ключа получаем новый символ шифровки:  
 $V+K = C1$ ,  $C+Л = C2$ ,  $T+Ю = C3$ ,  $P+Ч = C4$  и т.д.
- Зная ключ, можно провести обратную операцию:  
 $C1-K = V$ ,  $C2-Л = C$ ,  $C3-Ю = T$ ,  $C4-Ч = P$  и т.д.
- В компьютерных алгоритмах проще всего представить символы текста их двоичными кодами и производить над ними XOR (исключающее “или”)

# XOR

- XOR (eXclusive OR): “одно или другое, но не оба вместе”

p	q	XOR(p, q)
1	1	0
1	0	1
0	1	1
0	0	0



# Побитовое шифрование XOR

- Plaintext: A (ASCII-код 65 или 100 0001)
- Key: K (ASCII-код 75 или 100 1011)
- A XOR K =  
1000001  
1001011  
0001010 (=10 в десятичной системе)
- ASCII-коду 10 соотв. “line feed”, известная как LF или \n)
- Справедливо также обратное: '\n' XOR K = A

# Свойства ключа

- Выбор ключа оказывает не меньшее влияние на стойкость шифровки, чем выбор криптоалгоритма
  - Примитивный ключ + совершенный алгоритм или же сверхсложный ключ + простейший алгоритм примерно одинаково уязвимы
- Длинный ключ сложно безопасно передать другой стороне и сложно хранить в памяти человека
- В идеале каждый ключ должен применяться только 1 раз и быть равным длине текста, но непрактично (нужно безопасно передать ключ равный длине текста – проще передать текст)

# Методы взлома комп.систем

- **Физические**
  - похищение ключа, алгоритма, людей или оборудования, пытки и механический взлом
- **Технические** (*собственно криптоанализ*)
  - Попытки подобрать ключ и расшифровать криптограммы, изучая закономерности в данных, полученных не выдавая себя
- **Социальные**
  - Шантаж, подкуп, двойные агенты, угрозы

# Технические методы криптоанализа

- По криптограммам: чем больше число криптограмм, тем выше вероятность определения криптоалгоритма или ключа
- По расшифрованным частям криптограммы: часто в криптограммах встречаются похожие комбинации шифра, представляющие стандартные приветствия, дни недели, цифры, сокращения, названия мест, звания и т.п.
- По собственному открытому тексту и его криптограмме: криптоаналитик имеет доступ к криптосистеме и может получить результат её работы для своего текста (например, известен публичный ключ и алгоритм шифрования)
- По собственным криптограммам: криптоаналитик имеет доступ к криптосистеме и может получить результат её работы для сконструированной им криптограммы

# Частотный анализ

- Все естественные языки имеют характерное частотное распределение букв. Сообщения, зашифрованные методами простой замены, сохраняют это частотное распределение и тем самым предоставляют возможность раскрытия шифра.
- Достаточно подсчитать, сколько раз тот или иной символ встречается в криптограмме, чтобы с большой вероятностью предположить, какому символу открытого текста он отвечает (иногда даже не зная языка шифровки)
- [http://en.wikipedia.org/wiki/Letter\\_frequency](http://en.wikipedia.org/wiki/Letter_frequency)

# Типы криптосистем

- Симметричные: для шифрования/дешифрования используется один и тот же ключ, которыми корреспонденты обмениваются по защищенному каналу
- Ассиметричные: для шифрования используется один ключ (публичный), а для дешифрования – другой (приватный). Приватный ключ не может быть получен на основе открытого.
- Хэши (дайджесты): открытое сообщение произвольной длинный “сворачивается” в строку фикс.длины (*одностороннее шифрование без ключа*)

# Симметричные криптосистемы

- Один ключ для шифрации/дешифрации
- Ключ выбирается до начала обмена сообщениями
- Ключ держится в секрете обоими сторонами
- Стойкость обеспечивается длиной ключа, частотой смены ключа, сложностью криптоалгоритма
- Важен “лавинный эффект”: минимальное различие в исходных текстах (1 символ) должно давать непохожие шифры

# Блочные и потоковые шифры

- Блочный шифр: открытый текст разбивается на блоки, с каждым из которых происходит ряд преобразований при помощи ключа; зашифрованные блоки собираются в закрытый текст.
- Потоковый шифр: каждый символ открытого текста шифруется отдельно; на результат шифрования влияет не только ключ, но и позиция символа в тексте
  - Высокая скорость (в реальном времени)
  - Применение: мобильная связь (A3, A5, A8), беспроводные сети (RC4 в WEP для Wi-Fi)
  - Более “молодые”, развивающиеся алгоритмы

# Режимы шифрования блоками

- **Electronic Codebook (ECB – электронная кодовая книга)**



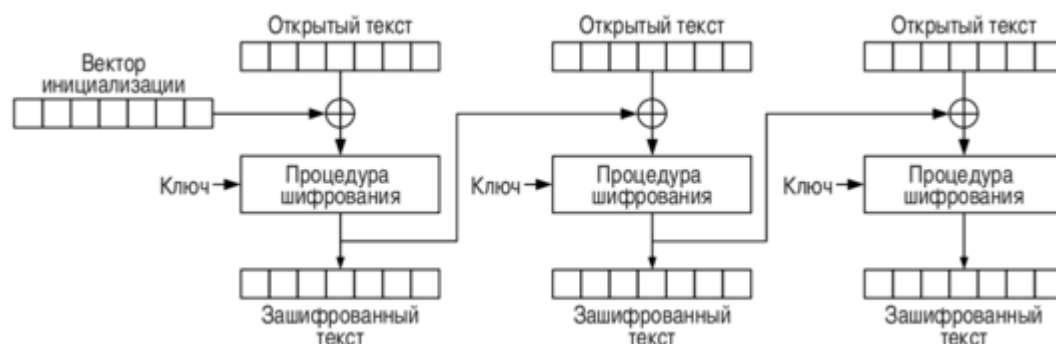
- Открытый текст делится на блоки равной длины, каждый из которых шифруется при помощи общего для всех блоков ключа отдельно от других блоков

# Плюсы-минусы ЕСВ

- + Процесс кодирования хорошо распараллеливается
- - Одинаковые блоки открытого текста преобразуются в одинаковый шифр – позволяет создавать “словарь” известных блоков
- - Алгоритм сам по себе не может определить свою целостность: уязвим к вставке лишних блоков, их замене или удалению из шифра

# Режимы шифрования блоками

- **Cipher Block Chaining (CBC – сцепление блоков шифра)**



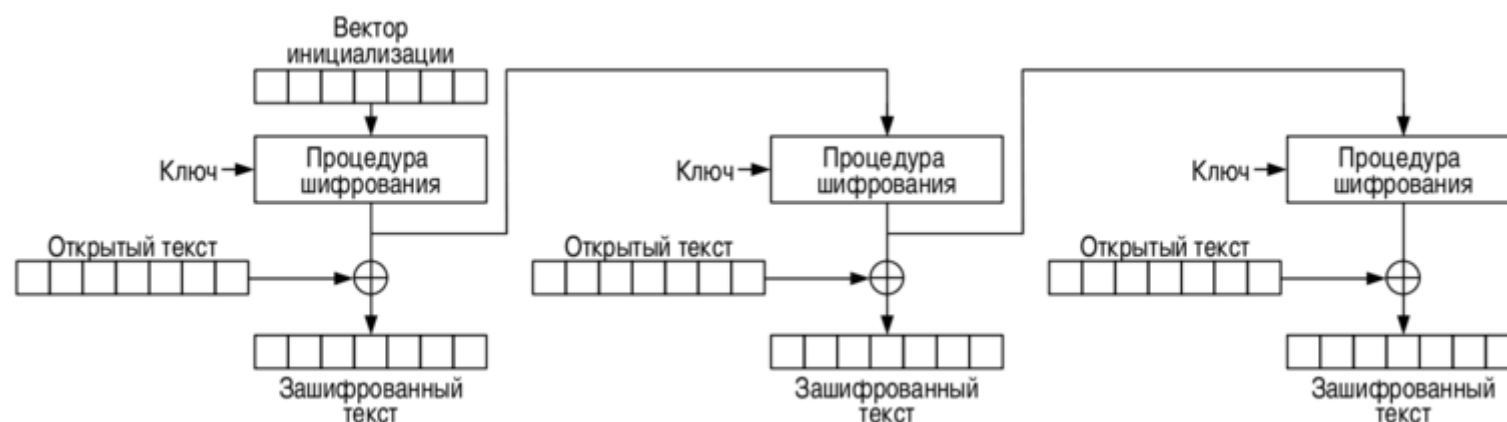
- Результат шифрации предыдущего блока используется для шифрации следующего
- Для первого блока используется случайно выбранное начальное значение, для следующих – XOR пред.блока

# Плюсы-минусы СВС

- + Одинаковые блоки открытого текста каждый раз превращаются в разные шифровки
- - Распараллелить можно только процесс дешифрации; но шифрование ведётся последовательно (= сравнительно медленно)
- - потеря или ошибка в одном из блоков фактически препятствует расшифровке последующих

# Режимы шифрования блоками

- **Output Feedback (OFB – обратная связь по ВЫВОДУ)**



- При помощи ключа и иниц.вектора генерируются зашифрованный ключевой блок, являющийся основой для последующих ключевых блоков
- XOR делается между каждым ключевым блоком и блоком открытого текста

# Плюсы-минусы OFB

- + Неплохо распараллеливается при кодировании/декодировании: последовательно нужно только обработать ключблоки (можно сделать это заранее), а XOR можно производить полностью независимо
- + Повреждённый блок закрытого текста не повлияет на расшифровку остальных блоков

# Достоинства симметричных криптосистем

- Очень высокая скорость (особенно с применением параллельных процессов)
- Простота реализации
- Криптостойкость пропорциональна длине ключа
- Хорошая изученность теоретической части

# Недостатки симметричных криптосистем

- Сложность управления ключами растёт квадратично с увеличением числа корреспондентов:
  - Для 10 человек нужно 45 ключей
  - Для 100 – 4950
  - Для 1000 – 499 500
- Ключ невозможно использовать для подтверждения авторства, т.к. Его знают обе стороны

# Распространённые алгоритмы

- DES / TripleDES – например, в кредитных картах
- AES – в беспроводных сетях
- IDEA – смарт-карты, VoIP, мультимедиа

# Ассиметричные криптосистемы

- Предпосылки:
  - Проблема безопасной передачи ключа
  - Проблема идентификации сторон
  - Появление мощных вычислительных ресурсов (проверка сверхбольших чисел на их взаимную простоту и проч.)
- Первый алгоритм (RSA) – 1973 год (рассекречен в 1977)

# Математика Диффи-Хелмана

1) Alice и Bob выбирают простое число  $p=23$  и случайное число  $g=5$  (всегда так, чтобы  $p>g$ . На самом деле  $p$  – сверхбольшое простое число). Эти числа не являются тайной.

2) Alice выбирает **тайное** целое  $a=6$  и отправляет Bob'у  $A = g^a \bmod p$

$$A = 5^6 \bmod 23 = 8.$$

3) Bob выбирает **тайное** целое  $b=15$  и отправляет Alice  $B = g^b \bmod p$

$$B = 5^{15} \bmod 23 = 19.$$

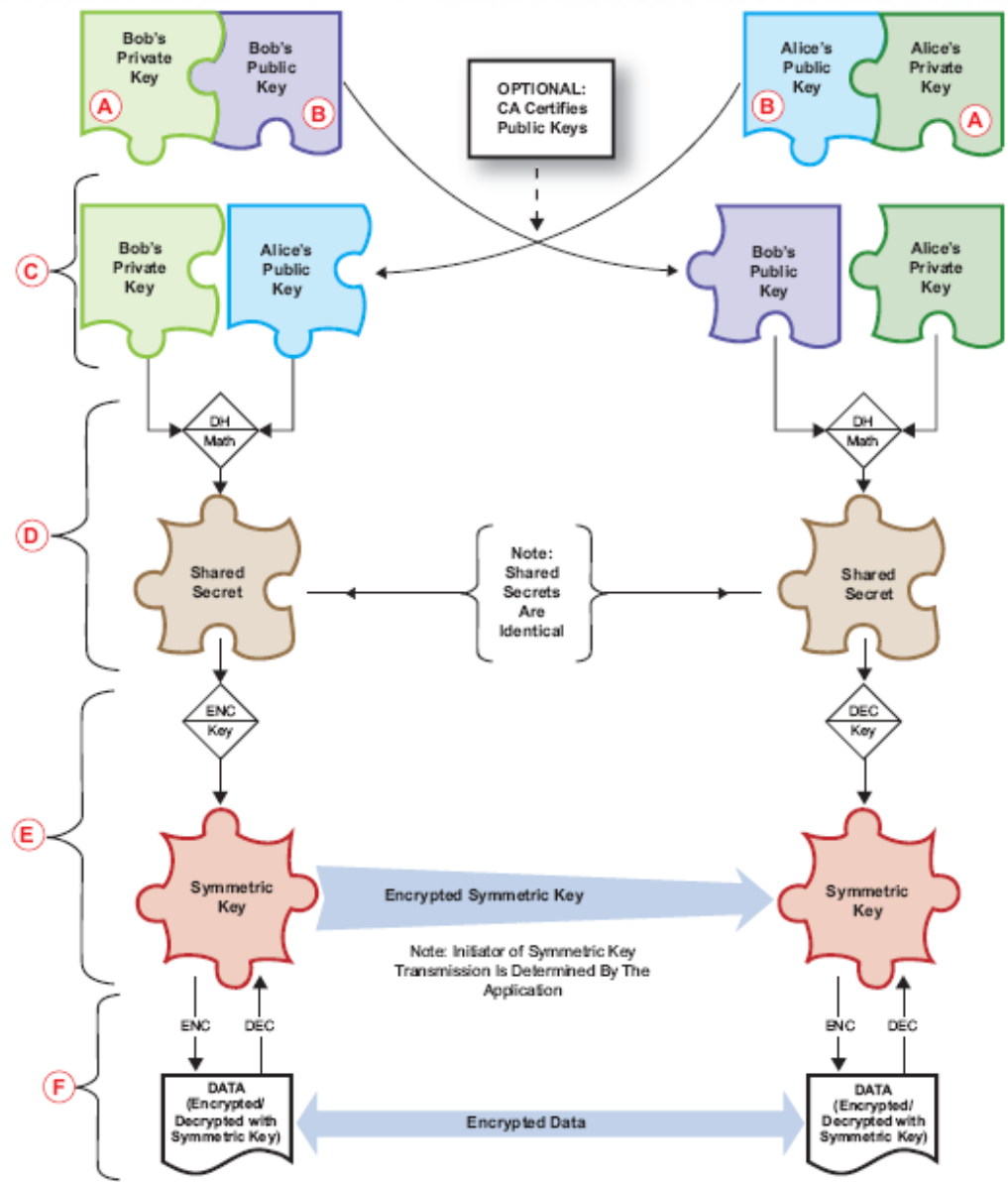
4) Alice высчитывает ключ:  $s = B^a \bmod p$

$$19^6 \bmod 23 = 2.$$

5) Bob высчитывает  $= A^b \bmod p$

$$8^{15} \bmod 23 = 2.$$

**РЕЗУЛЬТАТ:** оба корреспондента получили одинаковый ключ, который не участвовал в процедуре обмена и может быть использован как для симметричного шифрования (**красные значения** – тайные)



# Преимущества АСК

- Ключи можно передавать по незащищённому каналу
- У каждого корреспондента свой секретный ключ, что обеспечивает возможность т.н. “цифровой подписи”
- Срок жизни пары ключей в ассиметричной системе гораздо больше, чем симметричного ключа
- Для больших групп корреспондентов число ассиметричных ключей меньше, чем симметричных

## Недостатки АКС

- Уязвимы к атаке Man in the Middle (перехват и подмена ключей)
- Сравнительно медленные (иногда в 10-1000 раз в сравнении с симметричными)
- Опираются на сложную математическую теорию – сложно внести изменения в алгоритм, более сложные реализации
- Для получения криптоскойкости, равной симметричной криптосистеме, нужно использовать более длинные ключи (64 → 512, 128 → 2304 и т.д.)

# Область применения АКС

- Защита небольших объёмов информации
- Обмен симметричными ключами
- Аутентификация пользователей / сайтов
  
- Наиболее известные алгоритмы: RSA, DSA, El-Gamal (*последний сравнительно прост в реализации*)

# Хеши

- Хеш-функция (hash, digest, fingerprint, *räsi*) – однонаправленный алгоритм для преобразования сообщения произвольной в строку фиксированной длины (без *использования ключа*)
- Два разных хеша гарантированно принадлежат различным открытым текстам!
- Поскольку длина хеша всегда константна (*даже если открытый текст меньше!*), то могут найтись два открытых текста, хеш которых будет равен (**коллизия**)

# Криптографически нестойкие хеши

- Контрольные суммы (деление сообщения на блоки и суммирование их) – как в сетевых протоколах (TCP/IP)
- Возникновение коллизий маловероятно, но ВОЗМОЖНО
- Применяется для быстрого поиска одинаковых/измененных файлов, контроля целостности передачи и проч.
- Примеры алгоритмов: CRC32, Adler32

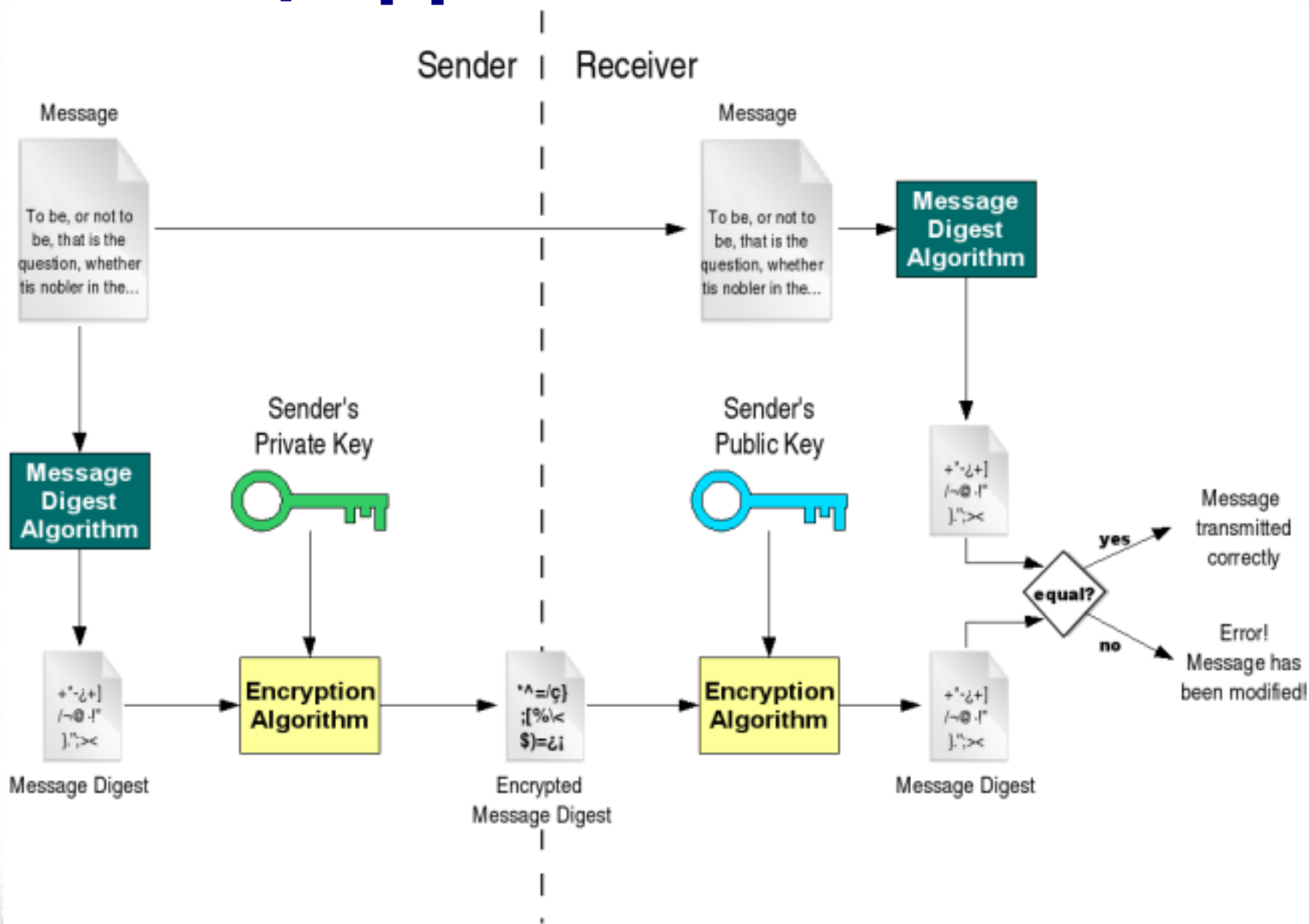
# Криптографически стойкие хеши

- Необратимость: для данного значения хеш-функции  $h$  должно быть неосуществимо в разумный срок найти блок данных  $H$ , для которого  $F(H) = h$ .
- Стойкость к коллизиям:
  - для сообщения  $H1$  должно быть вычислительно неосуществимо подобрать другое сообщение  $H2$ , для которого  $F(H1) = F(H2)$  – то есть с таким же хешем
  - должно быть неосуществимо в разумный срок подобрать пару любых сообщений  $H1$  и  $H2$ , имеющих одинаковый хеш
- “Лавинный эффект”: минимальное различие в исходных текстах (на 1 символ) должно давать очень разные хэши

# Применение хешей

- Контроль целостности на серьезном уровне
- Цифровая подпись: вместо шифрования всего документа достаточно зашифровать только его хеш, а документ передать в открытом виде. Целостность документа можно проверить, подсчитав его хеш и сравнив с расшифрованным значением
- Хранение чувствительной информации (паролей)

# Цифровая подпись



# Популярные хеш-функции

- MD5 (128 бит, придуман в 1991 году, скомпрометирован теоретически в 1996 году) – ненадежен с 2008 года
- SHA-1 (160 бит, придуман в 1996 году) – теоретически уязвим, но является стандартом
- SHA-2 (512 бит) – набирает популярность
- SHA-3 – в разработке

# Другие хеш-функции

Algorithm	Output size (bits)	Internal state size	Block size	Length size	Word size	Collision attacks (complexity)
<b>HAVAL</b>	256/224/192/160/128	256	1024	64	32	Yes
<b>MD2</b>	128	384	128	No	32	Almost
<b>MD4</b>	128	128	512	64	32	Yes ( $2^8$ ) <sup>[10]</sup>
<b>MD5</b>	128	128	512	64	32	Yes ( $2^{32}$ ) <sup>[12]</sup>
<b>PANAMA</b>	256	8736	256	No	32	Yes
<b>RadioGatún</b>	Arbitrarily long	58 words	3 words	No	1-64	With flaws ( $2^{352}$ or $2^{704}$ ) <sup>[13]</sup>
<b>RIPEMD</b>	128	128	512	64	32	Yes
<b>RIPEMD-128/256</b>	128/256	128/256	512	64	32	No
<b>RIPEMD-160/320</b>	160/320	160/320	512	64	32	No
<b>SHA-0</b>	160	160	512	64	32	Yes ( $2^{39}$ ) <sup>[14]</sup>
<b>SHA-1</b>	160	160	512	64	40	With flaws ( $2^{52}$ ) <sup>[15]</sup>
<b>SHA-256/224</b>	256/224	256	512	64	32	No
<b>SHA-512/384</b>	512/384	512	1024	128	64	No
<b>Tiger(2)-192/160/128</b>	192/160/128	192	512	64	64	"Pseudo-near collision" <sup>[16]</sup>
<b>WHIRLPOOL</b>	512	512	512	256	8	No

# Взлом хеш-функции

- Поиск коллизий - малореален, невозможен для всех хешей и текстов
- *Bruteforce* (подбор методом последовательного перебора или перебора по словарю) малореален вследствие временных ограничений)
- *Rainbow Tables* (“Радужные таблицы”) - поисковые таблицы, где для увеличения скорости поиска в жертву приносится объём потребляемой памяти и точность поиска

# Радужные таблицы

- “Настроены” на конкретную хеш-функцию (таблицы для MD5 не подойдут для взлома SHA-1)
- Не содержат всех возможных паролей, поэтому при использовании происходит поиск+вычисление пароля по специальному алгоритму
- Например, для поиска с 75% вероятностью 8-символьного текста, хешированного MD5, нужны таблицы размером 600ГБ. Их генерация на современном компьютере занимает не менее 1 года, а поиск – около 10-15 минут.

# Защита хешей

- Используйте salt (“соль”) - уникальное значение, добавляемое к открытому тексту
- $\text{hash}(\text{opentext} + \text{salt}) = \text{unique hash}$
- Salt – любой случайный текст
- Salt – уникальна для каждого хеша
- Не используйте в качестве соли короткие значения, которые могут быть легко вычислены (ID записи, дата регистрации)
- Для проверки такого хеша нужно хранить salt

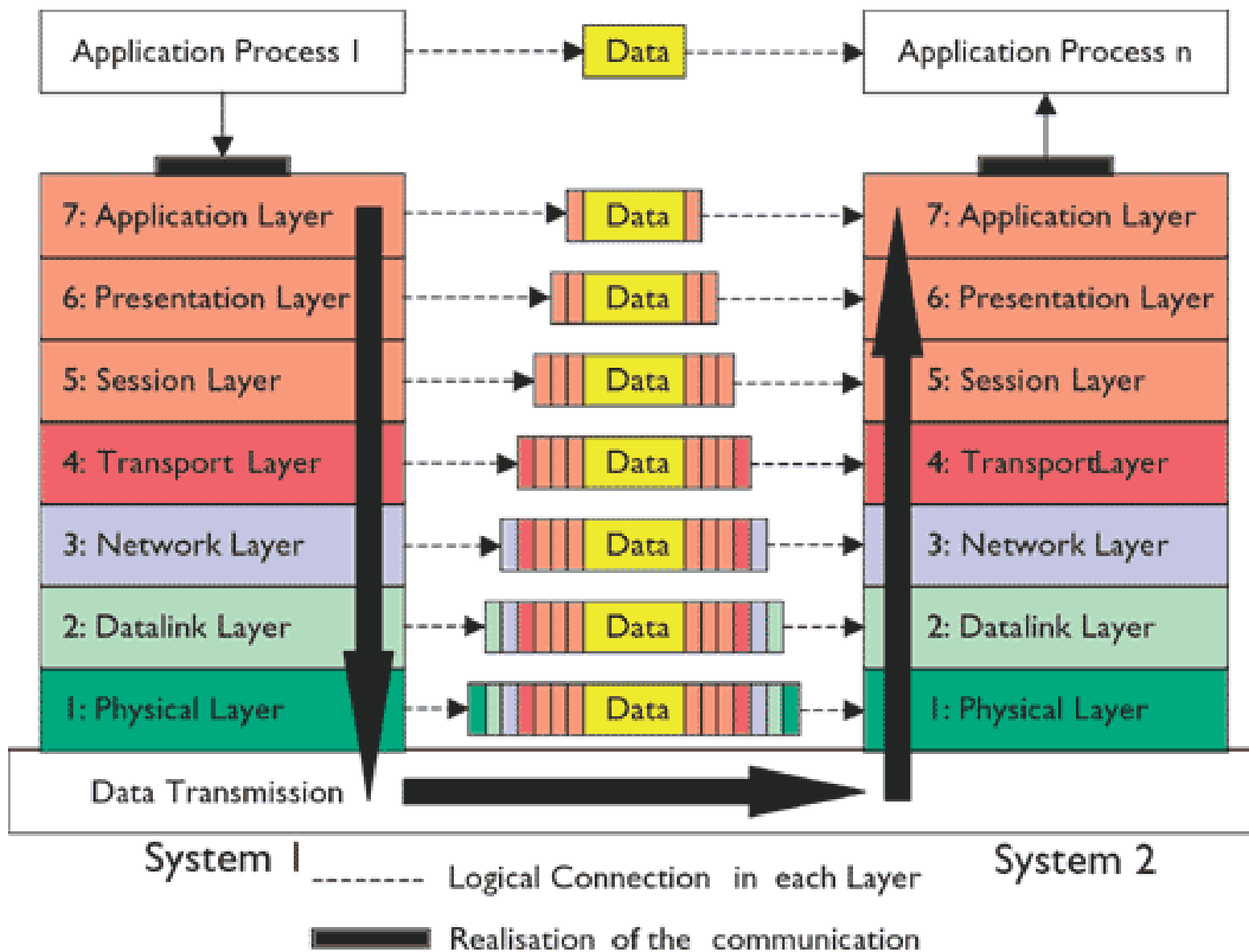
# Использование “соли”

- $\text{hash}(\text{opentext} + \text{salt}) = \text{unique hash}$
- $\text{md5}(\text{"password"}) =$   
 $\text{"5f4dcc3b5aa765d61d8327deb882cf99"}$
- “Соль” = current timestamp (текущее дата-время до миллисекунд)
- $\text{md5}(\text{"password20100310173027137"}) =$   
 $\text{"b3dedf134692365df1e03efaa8ed4f83"}$
- Два пользователя с одинаковым паролем скорее всего получат разные хеши!
- Такого пароля скорее всего не будет в радужных таблицах

# Безопасность сетей Ethernet

- Говорим “сети Ethernet” – подразумеваем локальные сети с коммутатором (switch) или концентратором (hub)
- Wi-Fi – “надстройка” над сетью Ethernet, общий порт в сетевом коммутаторе (т.е. концентратор для всех wifi-соединений)

# Снова модель OSI



# Почему Ethernet уязвим?

- Уровни OSI – изолированы. Каждый уровень работает так, как будто не знает ничего о существовании других уровней
- Если один из уровней будет “взломан”, то другие ничего не узнают и не смогут определить аномалию
- “Интеллект” (пароли, софт и т.д.) появляются на верхних уровнях, нижние же сравнительно малозащищены, а потому уязвимы для различных атак
- Ethernet настолько широко распространён, что заменить неудачную реализацию невозможно

# Атаки физического уровня

- Для осуществления атаки требуется физический доступ к устройству
- У компьютера есть около 10 физических портов (не только Ethernet)
- Порты есть не только у компьютера (switch, hub, router – почти любое оборудование)
- К порту можно подсоединить почти любое устройство, которое удастся незаметно пронести (и вынести) из помещения
- Физический уровень не ограничивается только портами!

# Что и для чего подключают?

- Клавиатуру, мышь, монитор – подбор пароля, просмотр и копирование данных
- Аппаратный keylogger (запись ввода с клавиатуры) – продаётся интернете, 10\$
- Генератор помех (*prankster*)
- Сетевой интерфейс можно переподключить к своей сети/компьютеру или использовать второй сетевой интерфейс
- *Pod slurping* (iPod или внешний накопитель с программой для автопоиска и копирования данных)

# Другие атаки физического уровня

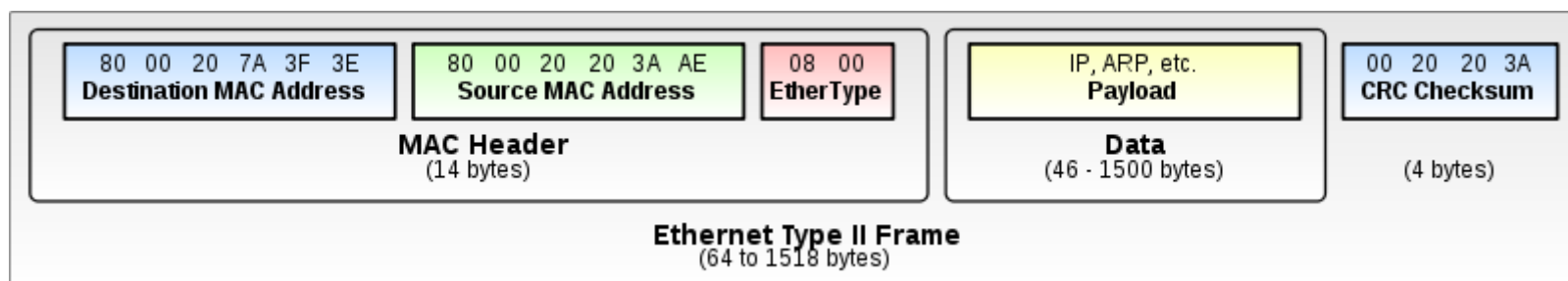
- Электроснабжение: отключение или скачки
  - Диверсия
  - Вывод оборудования из строя
  - Прикрытия для других физических атак
- Environmental Control (температура, влажность)
  - перегрев оборудования для его перезагрузки или выхода из строя
- Отключение устройств резервного копирования и других критических компонент

# Защита на физическом уровне

- Физическая защита: замки, двери, охрана
- Политики доступа в помещения, обращения к данным, использованием периферией
- Для серверов и сетевого оборудования: все неиспользуемые порты должны быть выключены!
  - #int fa0/1 shutdown
  - USB и Firewire тоже можно отключить программно
- Оборудование в стойках должно быть заперто на замок
- Видеонаблюдение

# Канальный уровень

- Физическая адресация (MAC), контроль ошибок (LLC), фреймы, драйвер сетевого адаптера



- MAC-адрес: 48-битный идентификатор
  - Первые 24 бита: код производителя
  - Следующие 24 бита: код интерфейса
  - Может быть изменен программно
    - `ifconfig ethN hw ether <mac-address>`
  - При *broadcast* передаются 0xF

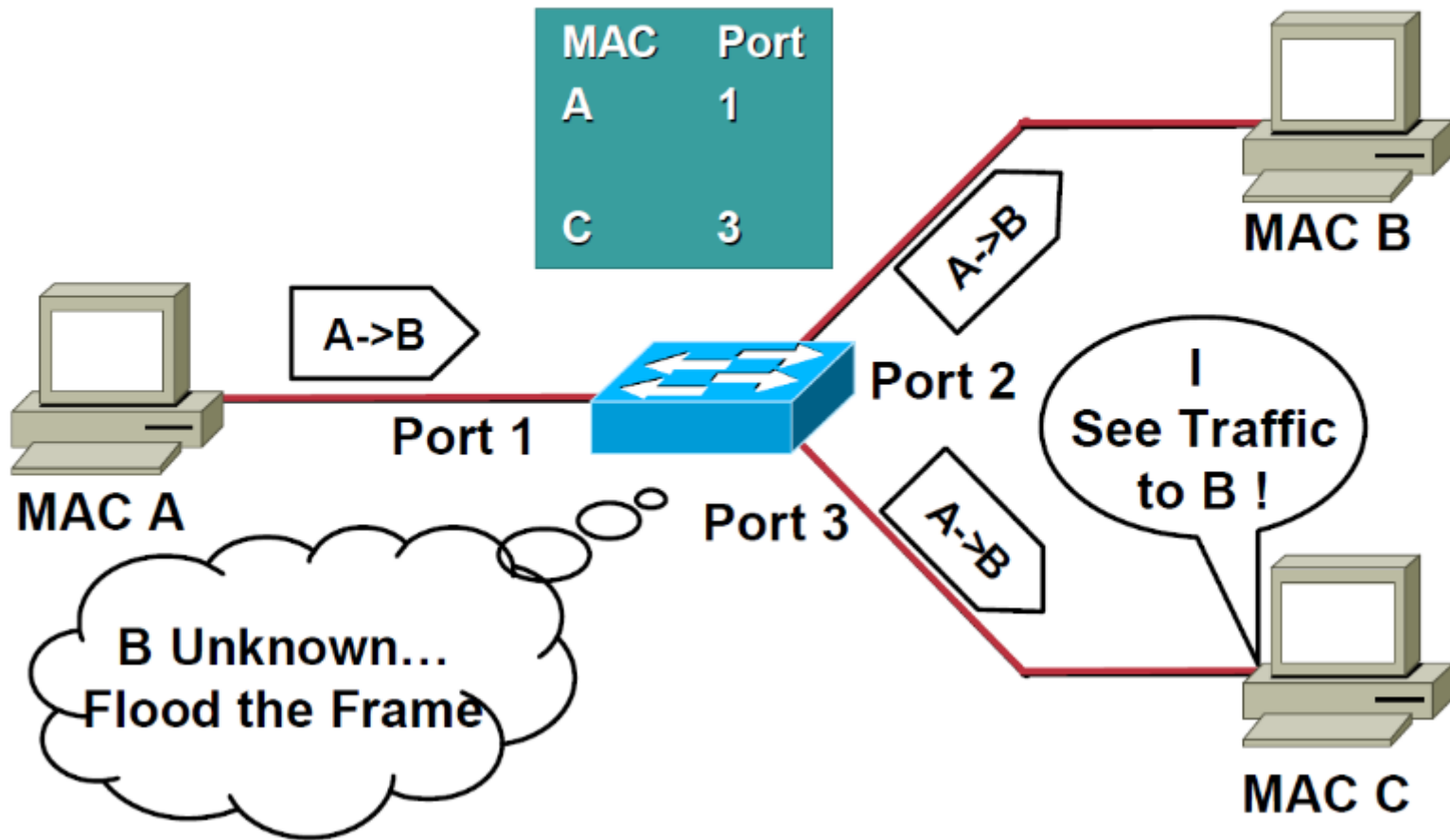
# Атака против коммутатора CAM-overflow

- Цель: перехватить трафик от хоста А к хосту В через хост С, каждый из которых подключён к своему порту коммутатора
- Коммутатор хранит соответствия между своим физическим портом и подключенным к нему MAC-адресом сетевого интерфейса в CAM-таблице
- CAM-table (Content Addressable Memory) имеет фиксированную длину (несколько тысяч-десятков тысяч записей)
- Впервые проведена в 1999 году

# CAM Overflow

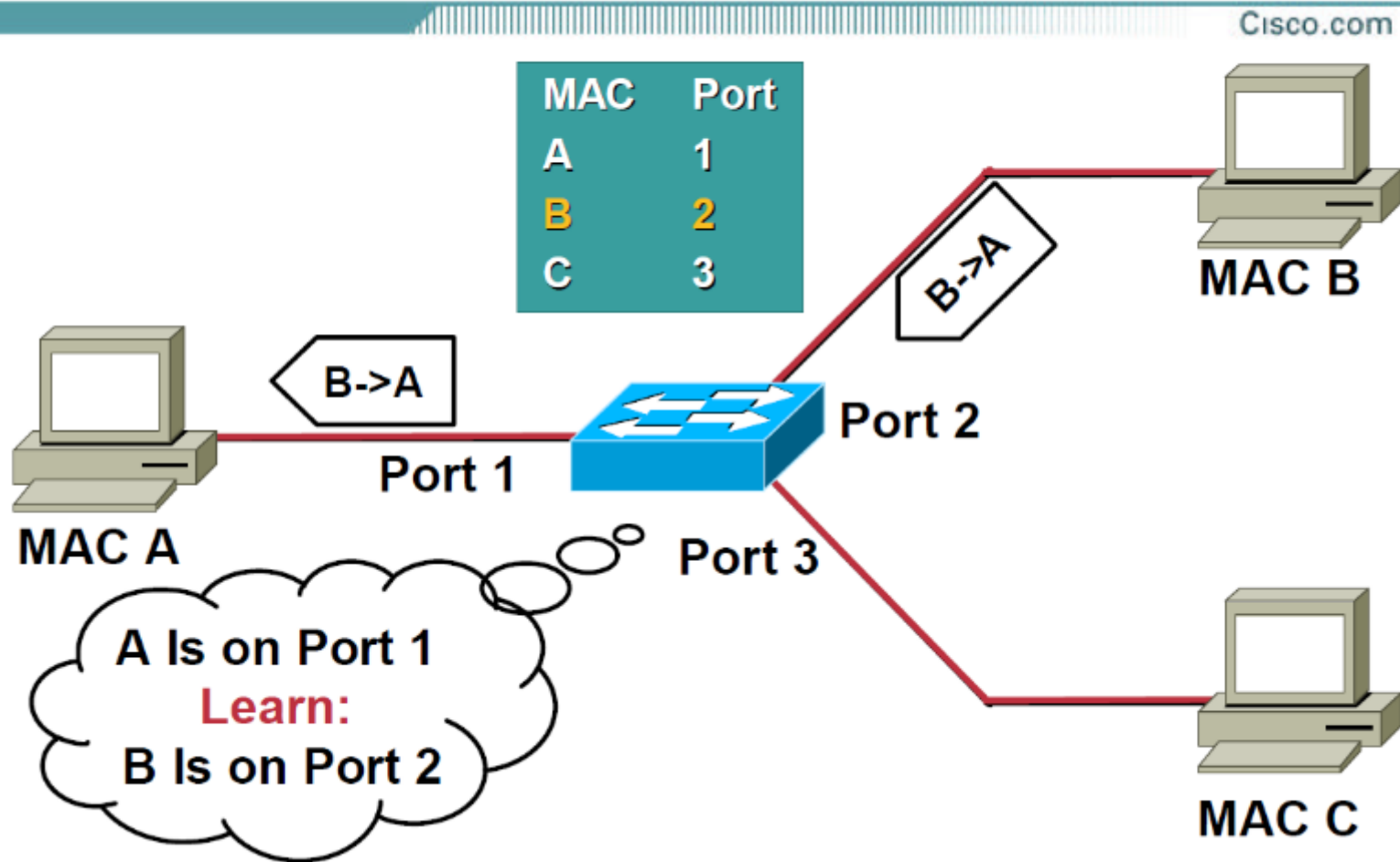
## Normal CAM Behaviour 1/3

Cisco.com



# CAM Overflow

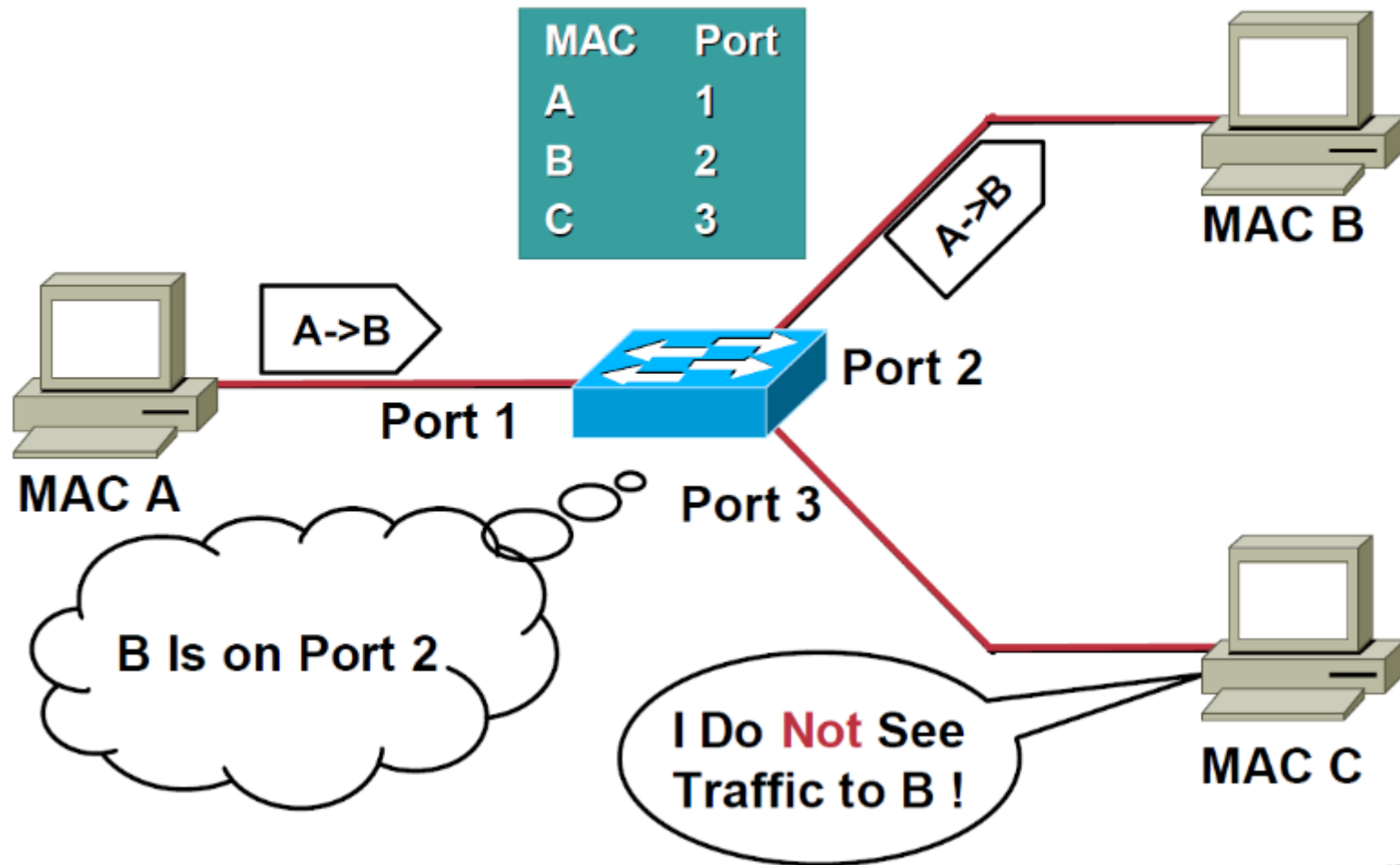
## Normal CAM Behaviour 2/3



# CAM Overflow

## Normal CAM Behaviour 3/3

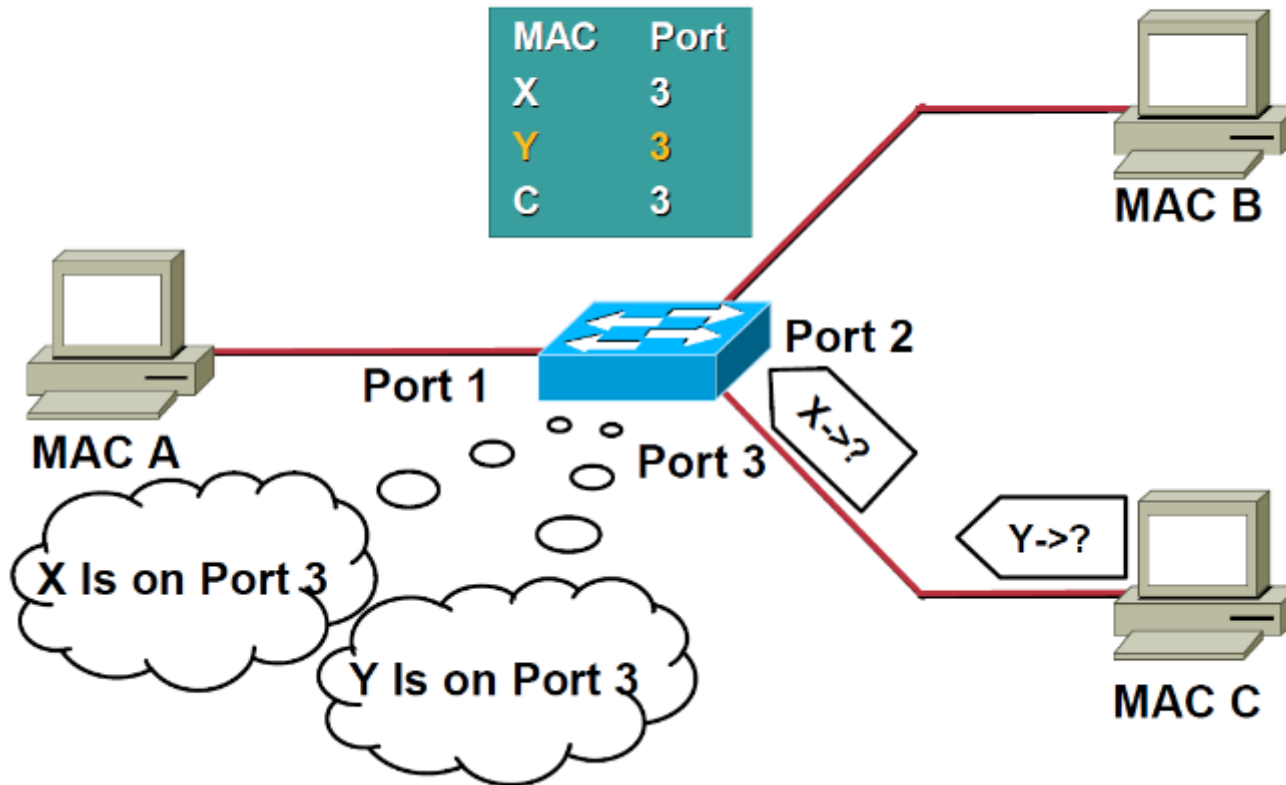
Cisco.com



# CAM Overflow

## CAM Overflow 2/3

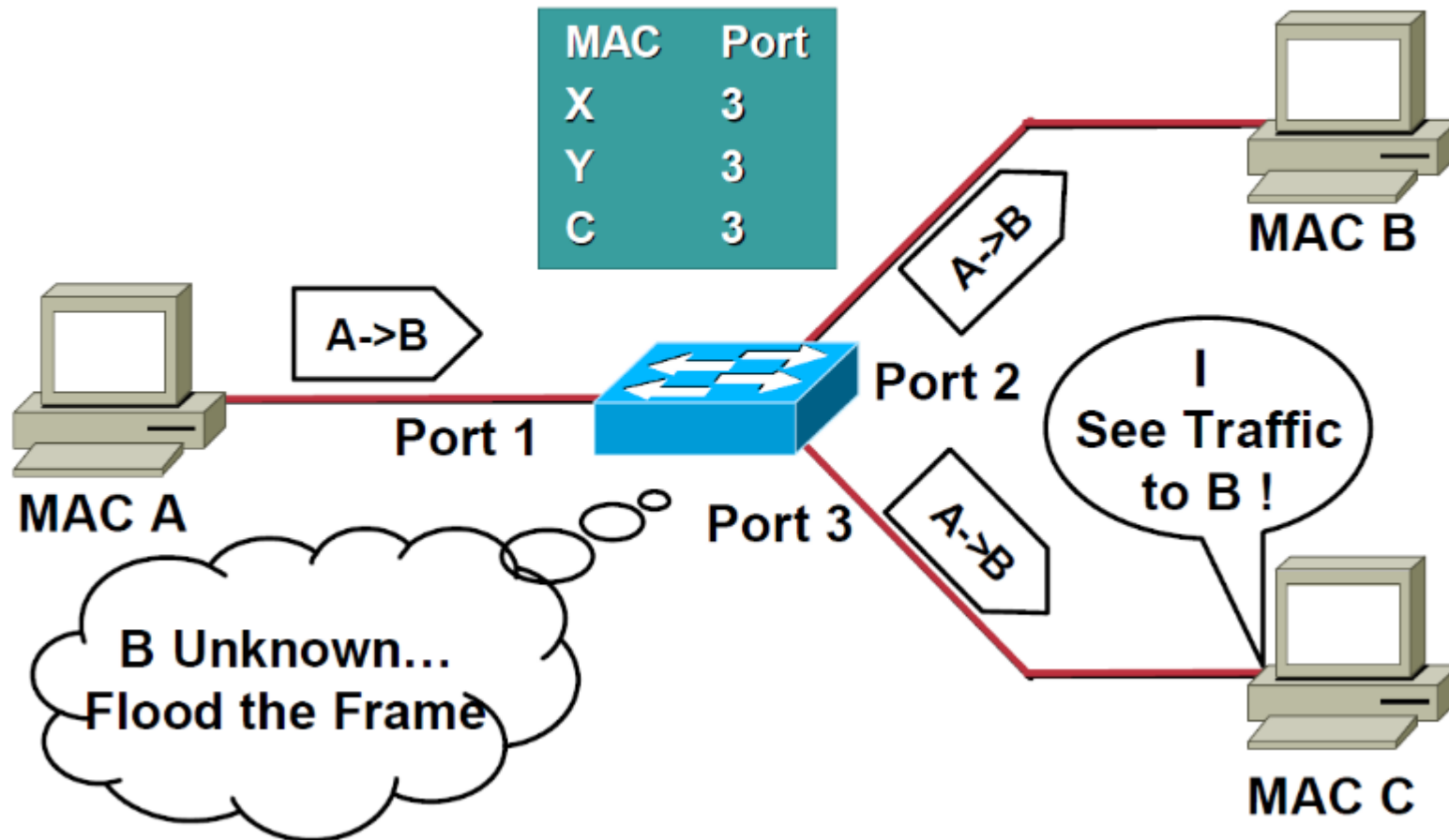
Cisco.com



# CAM Overflow

## CAM Overflow 3/3

Cisco.com



# Реализация атаки

- Утилита dsniff для Linux – генерирует 8000 CAM-записей в секунду
- Через минуту CAM-таблица свича переполнена и трафик, не находящий нужной записи уходит в broadcast (на интерфейс злоумышленника)
- Трафик между хостами, чьи MAC-адреса уже были внесены в таблицу не будет нарушен

# Защита от CAM Overflow

- Статические CAM-таблицы (неудобно в больших сетях и если устройства часто меняются)
- Ограничение числа MAC-адресов на порту
  - #int fa0/1
  - #port security max-mac-count 2
- Отключение подозрительных портов (или другое действие)
  - #port security action shutdown

# ARP: Address Resolution Protocol

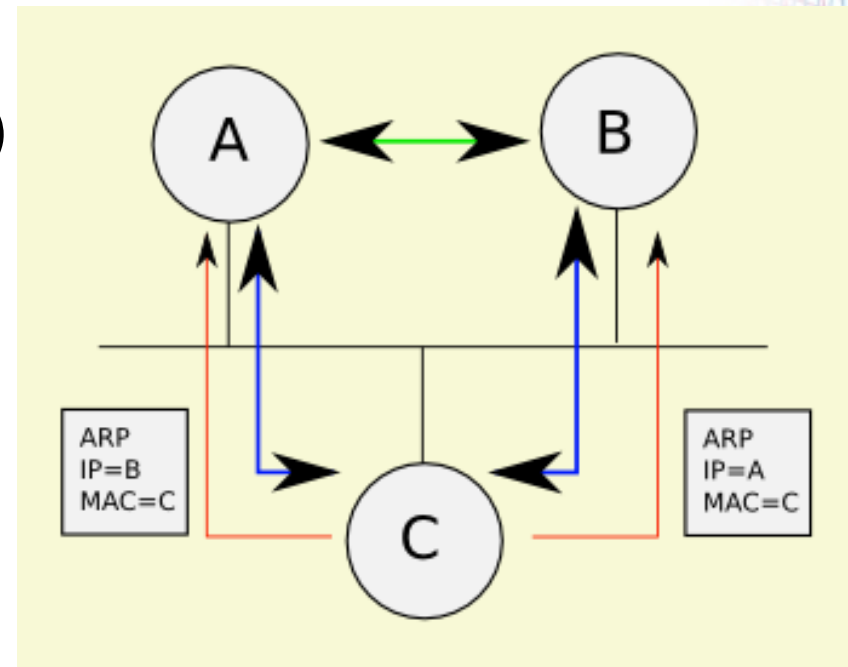
- Отвечает за соответствие между физической (по MAC-адресу) и логической (по IP-адресу) адресацией
- ARP-cache (ARP-таблица): таблица соответствия MACaddr → IPaddr
- Если в ARP-таблице нет MAC-адреса нужно хоста, то рассылается broadcast, на который отвечает только тот хост, которому адресован пакет по IP-адресу
- gARP (Gratuitous ARP) – оповещение хостом других хостов о смене им IP- или MAC-адреса (тоже через broadcast)
- Протокол работает “на честном слове”

# ARP Spoofing & ARP Poisoning

Суть атаки: путём подмены записей (*poisoning* - “отравление”) в ARP-таблице хостов можно:

Вызвать DoS (*Denial-of-Service* – “отказ в обслуживании”), если подменить MAC-адреса хоста (рутера) несуществующим;

Подслушать (*spoofing*) траффик – перенаправив его на хост атакующего (атака типа *man-in-the-middle*).



Атака на А и В

↔ ARP-spoofing

Обмен данными между А и В

↔ До ARP-spoofing'a

↔ После ARP-spoofing'a

# Как может навредить ARP-spoofing?

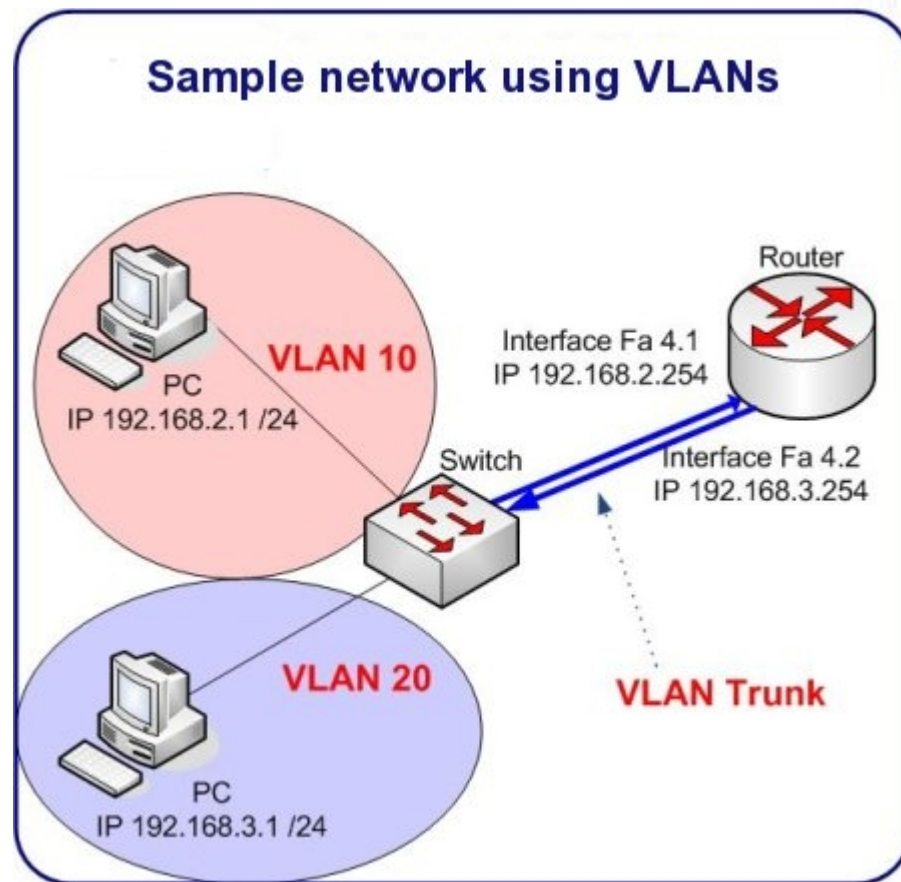
- Перехват всех данных, передаваемых по незащищённым протоколам верхних уровней (HTTP, POP3, SMTP, SNMP, FTP) – паролей, хешей, ключей и проч.
- Подмена сертификатов и DNS-записей (перенаправление с сайта банка на ложный сайт)
- Отказ от обслуживания сети или её сегмента

# Инструментай для ARP-атак

- Cain & Abel – для Windows
- Dsniff – для Linux
- Ettercap – Win/Linux
- Arp-sk (winarp-sk) – Win/Linux

# Защита от ARP-атак: VLAN

- VLAN (Virtual LAN) – создают отдельный broadcast domain, в идеале – каждому клиенту свой VLAN
- Необходим свитч, поддерживающий достаточное число VLANов (или каскад таких свитчей)
- Обмен данными между VLANами будет возможен только через рутер



# Защита от ARP-атак: шифрование

- Зашифровать весь трафик (IPSec, PPPoE, PPTPd)
- Нечего прослушивать – нет смысла использовать ARP-spoofing
- Остаётся риск DoS-атак (*не подслушаем, так сломаем!*)
- Нужен дополнительный софт на клиенте
- Увеличивается нагрузка на CPU

# Защита от ARP-атак: статические настройки

- Использовать статические ARP-таблицы (не всегда помогает, т.к. gARP может их обновлять) – зафиксировать IP/MAC для рутера или вообще для всех хостов
- Ф-ция *ACL packet filtering* на умных коммутаторах анализирует соответствие
- Привязать известный MAC-адрес к определенному порту коммутатора
- Отключить ARP вообще (все изменения вносятся руками, неизвестные хосты не получают доступа – слишком радикально)

# Мониторинг ARP-атак

- Суть: определить хост (порт) виновника атаки и отключить его от сети
- Утилита `arpwatch` для Linux
- Утилита `Xarp` для Windows
- Средствами IDS/IPS (Intrusion Detection System / Intrusion Prevention System) – уведомление о подозрительной активности например по SNMP и принятие решений (например, о восстановлении ARP-таблицы по умолчанию)

# Советы сисадмину

- Использовать “умное” оборудование, IDS/IPS
- Неиспользуемые порты объединить в отдельный VLAN и/или выключить
- Использовать шифрование трафика для незащищённых протоколов (IPSec, PPPoE)
- В критических местах (рутеры, свичи, сервера) использовать статическую адресацию MAC/IP
- Предпочитать протоколы, требующие шифрации (HTTP → HTTPS, FTP → FTPS, POP3 → POP3S)
- Производить настройки через SSH (а не Telnet), а мониторинг – через SNMPv3
- Иметь backup всех настроек и план на случай чрезвычайной ситуации

# Атаки сетевого уровня (L3)

- L3: определение лучшего пути передачи данных, маршрутизация
- Все протоколы L3 так или иначе уязвимы (ICPM, IGMP, RIP, BGP, IPX)
  - Авторизация в виде plaintext или md5
  - Более умные → шире круг возможных атак
  - Использование broadcast и multicast
- Например, можно добавить в сеть рутер с более высоким приоритетом (=получить преимущество при пересылке траффика)
- Зацикливание траффика (looping), проведение штормов (flooding), перенаправление (routing table injection)

# Защита на сетевом уровне

- Как правило – прерогатива сисадминов ISP и крупных организаций
- Для сисадмина малой организации: firewall и/или IDS/IPS
- Безопасность упирается в:
  - правильный выбор оборудования,
  - планирование сети,
  - мониторинг сети,
  - выбор надёжных партнёров (ISP)

# Безопасность на уровнях 4-7

- Начиная с L4 (транспортный) обмен данными идёт на логическом уровне, а не на физическом
- Полагаются на нижние уровни
- Реализуются программно (содержат ошибки в коде, уязвимы через обновления, backdoor, etc)
- Достаточно сложные протоколы, их “прослушивание” простыми средствами затруднено
- Цель атак – получить контроль над устройством (а не только над передаваемыми им данными!)
- Невозможно обнаружить вторжение сетевым оборудованием (свичом, рутером)

# Ср-ва обеспечения безопасности на верхних уровнях OSI

- Firewall
- VPN
- Anti-virus
- IDS / IPS
- Мониторинг
- Политика безопасности

# Особенности Wi-Fi сетей

- Сети Wi-Fi (семейство протоколов IEEE802.11) – “надстройка” над Ethernet
- Разница – в L1 и L2 уровнях по OSI
- Большинство “точек доступа” (Wi-Fi access point) работает в режиме “хаба”, а потому все клиенты могут видеть трафик друг друга (если он не зашифрован индивидуальным ключом)
- Беспроводная сеть – рай для злоумышленников

# Уровни авторизации в Wi-Fi

- Open (no authorisation) – любой может подключиться к сети (аналогично физическому подключению кабеля к порту свитча или хаба Ethernet)
- WEP – потоковый шифр (RC4) 40-бит – взломан в 2002 году, рекомендуется прекратить использование после 30.06.2010)
- WPA (TKIP) – 128-битный ключ (RC4) – доказана уязвимость в 2008/2009 гг (с ограничениями)
- WPA2 (CCMP) – блочный шифр (AES), 128-битный ключ. Пока уязвим только bruteforce

# Атаки на Wi-Fi сети

- WEP: возможен взлом любого ключа за сравнительно небольшое время, причём неважно наличие активных клиентов
- WPA и WPA2: пока на практике применяется только “handshake dump” с атакой по словарю, но взлом TKIP возможен и без словаря!
- DoS атаки сравнительно просты (глушение сигнала, отключение клиентов и проч.)

# Элементарная защита Wi-Fi

- Физический уровень:
  - Ограничить мощность передатчика
  - Ограничить время работы передатчика
  - Использовать направленные антенны
- “Ламерский уровень” (от школьников):
  - Запретить вещание ESSID
  - “Белый список” MAC-адресов клиентов

# Защита частных Wi-Fi сетей

- Используйте WPA2-PSK с шифрацией AES,
- Выбирайте длинный безопасный ключ (не менее 8 символов, спецсимволы), которого нет в словарях
- Периодически меняйте ключ, если он сравнительно простой
- Обновите firmware своей точки доступа или используйте open-source дистрибутив DD-WRT (Open WRT и им подобные)

# Защита корпоративных Wi-Fi сетей

- Создайте 2 сети: одну для работников (и доверенных лиц), другую – для всех желающих
- В сети для работников используйте персональные сертификаты (802.1x, Radius)
- Вынесите каждую из сетей в отдельный VLAN
- Настройте фильтрацию траффика и ограничение скорости в публичной сети
- Используйте мониторинг и политики для контроля корпоративной сети

# Honeypot aka Evil Twin Wi-Fi

- А что если мы установим бесплатную открытую точку доступа, над которой имеем полный контроль?
- А что, если это сделает злоумышленник?
- Мораль: не подключайтесь к неизвестным публичным точкам доступа, а если пользуетесь ими, то предпочитайте только защищённые протоколы (HTTPS, POP3S, FTPS и другие)

# Защита рабочей станции

- Kensington lock для ноутбука
- Антивирус (хотя бы Windows Security Essentials)
- Firewall (хотя бы Windows Firewall)
- Запретить “Общий доступ” (Shared access) к информации на диске компьютера
- Обновлять весь критический софт, имеющий доступ к интернету (от браузера, до FTP-клиента и Skype) и опсистему
- Отключить автозапуск приложений с подключенных внешних устройств

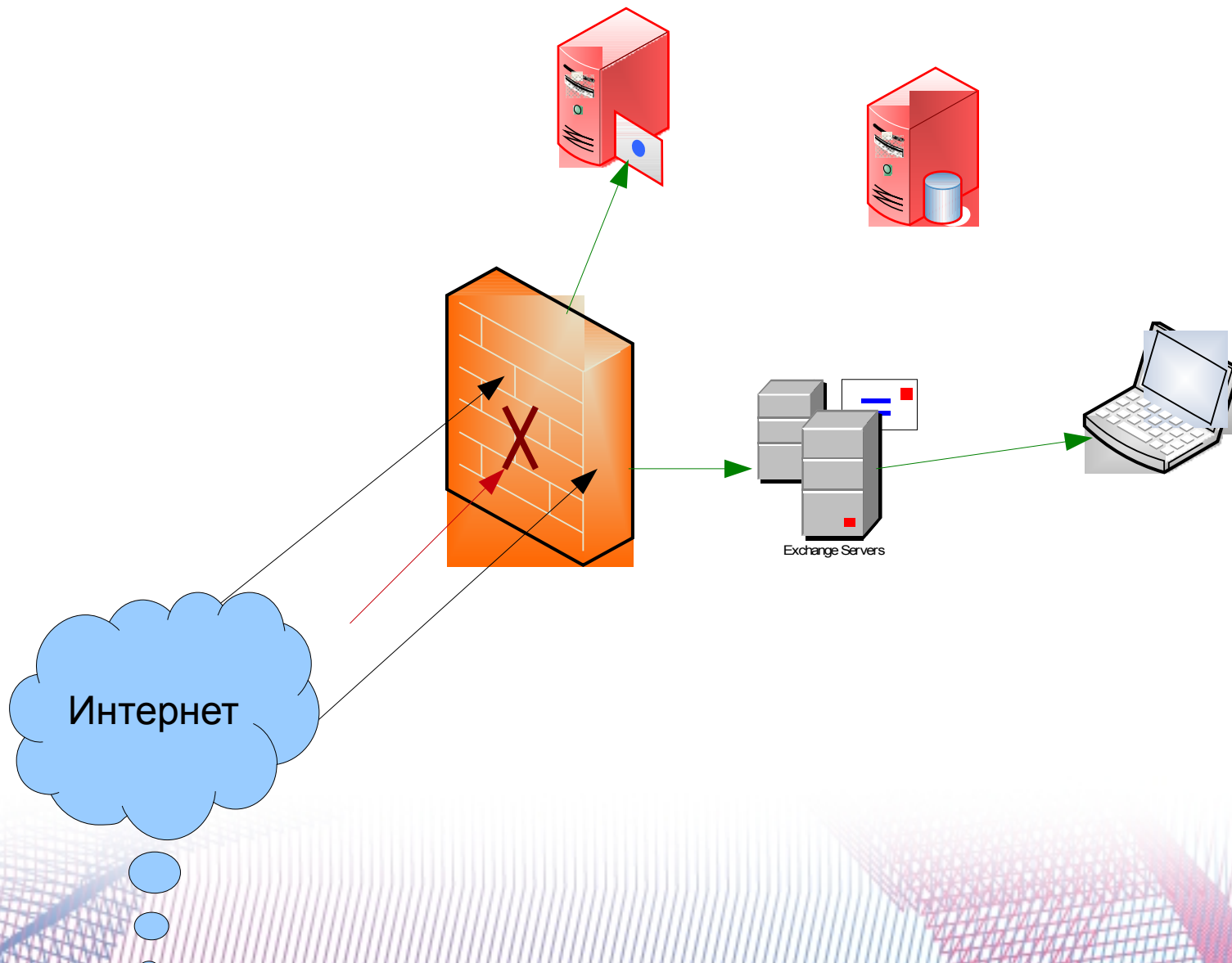
# Антивирусы

- Цена (может ли бесплатное быть хорошим?) – удачных примеров мало (ClamAV)
- Размер базы данных известных вирусов
- Эвристические алгоритмы (“интеллект”)
- Частота обновлений и скорость реакции
- Дополнительные возможности (контроль изменений регистра и т.п.)

# Firewall aka брэндмауэр

- Межсетевой экран (*firewall, tulemüür*) – программно-аппаратный комплекс для контроля и фильтрации траффика на основе заданных правил
  - Аппаратные (отдельное “железо”, на бытовом уровне часто объединены с рутером)
  - Программные (“железную” часть берет на себя рабочая станция; на бытовом уровне часто объединены с антивирусом)

# Суть фаервола



# Правила фаервола

from	port	to	port	Service	rule
192.168.1.x	any	Internet	any	HTTP, FTP	Access
Internet	any	192.168.1.10	80	HTTP	Access
215.45.1.114	22	192.168.1.10	22	SSH	Access
any	any	any	any	any	Deny

# Функции аппаратного фаервола

- Разделить локальную сеть на несколько “зон” с различными правилами для каждой из них
- Перенаправлять соединения на различные “порты” серверов и рабочих станций в локальной сети
- Создавать и поддерживать VPN-туннели
- Опционально: анализировать проходящий трафик
  - антивирус, антиспам, подсчёт и приоритезация трафика

# Функции программного фаервола

- Делать рабочую станцию “невидимой” даже для пинга в локальной сети
- Блокировать входящий/исходящий трафик
- Ограничивать доступ к веб-сайтам
- Опционально: антивирус, антиспам, и т.д

# Data forensics

- Данные на магнитных носителях (HDD, USB flash, etc) можно восстановить до тех пор, пока в ячейки памяти не будет записана новая информация
  - Empty Recycle Bin не удаляет данные :)
  - Quick Format тоже не удаляет
  - Уничтожение MBR и FAT фактически не удаляет данные, а только усложняет процесс их восстановления
- Списанные и сломанные носители информации – ценная находка для злоумышленника
  - Используйте размагничивание
  - Или физические уничтожение носителей

# **Защита (веб)- программного обеспечения**

# Немного статистики

- 83% веб-сайтов содержат те или иные уязвимости
- Из них 78% содержат относительно критические уязвимости
- С вероятностью до 20% уязвимые веб-сайты могут быть заражены вредоносным кодом автоматически

(Источник: отчёт [ptsecurity.ru](http://ptsecurity.ru) за 2008 год)

# Угрозы ПО

- Path Traversal (Directory Traversal) – “обратный путь”
- Слабая аутентификация и авторизация
- Утечка информации (например, через логи или сообщения об ошибках)
- XSS (Cross-Site Scripting) – межсайтовое исполнение скриптов
- SQL-Injection (внедрение произвольных SQL-запросов)

# SQL-injections

- Суть атаки: изменить передаваемые веб-приложению параметры так, чтобы изменить выполняемый на их основе SQL-запрос

```
$sql = "SELECT * FROM users WHERE  
username = ' " . $_GET['username'] . " '  
AND  
password = ' " . $_GET['password'] . " '";
```

- В общем случае атака не зависит от СУБД, ЯП, типа HTTP-запроса (POST/GET) или типа передаваемой переменной (строка, число, дата, логическая переменная)
- Атака может быть выполнена посредством cookies, серверных переменных и т.п. автоматически

# Где можно выполнить инъекцию?

- В любой запрос: `SELECT`, `INSERT`, `UPDATE`, `DELETE`
- В любой участок запроса: список выбираемых полей, список таблиц, условия объединения таблиц (`JOIN`), условия выборки данных (`WHERE`), упорядочивание (`ORDER BY`), группировка (`GROUP BY`), ограничение выборки (`LIMIT`)
- Конечный вид инъекции зависит от типа и версии используемой СУБД

# Как выполнить инъекцию?

- `http://example.com/?name=abc`
- `$sql = "... WHERE name = $GET['name'] ...";`
  - `?name=abc'`
  - `?name=abc"`
  - `?name="--`
  - `?name="--`
  - `?name=abc OR 1=1--`
  - `?name=abc' OR '1' = '1`
  - `?name=abc ORDER BY RAND()`

# Что мы должны увидеть?

- Хороший сценарий (безопасный сайт):
  - Культурное сообщение об ошибке (404) или другой явный признак, что переданные параметры обрабатываются и игнорируются (отсутствие вывода вообще)
  - Ничего не изменилось (но это ещё не факт, что инъекция не сработала!)
- Плохой сценарий (уязвимый сайт)
  - Сообщение об ошибке со стороны СУБД
  - Изменился вывод страницы в соответствии с переданными параметрами
  - Часть страницы с данными исчезла из вывода

# Примеры

- ?name=abc возвращал 1 запись, а
  - ?name=abc+OR+1=1--  
вернул несколько записей
- ?name=abc возвращал несколько записей, а
  - ?name=abc+ORDER+BY+RAND()--  
вернул их же переупорядоченными в случайном порядке

# Что можно натворить?

- Получить доступ к данным других таблиц:
  - `SELECT * FROM news WHERE id = $_GET['id']`
  - `?id=-1+UNION+SELECT+*+FROM+admin`
  - `SELECT * FROM news WHERE id = -1 UNION SELECT * FROM admin`
  - Получаем все записи из `admin` и ни одной записи из `news`
- Получить доступ к данным в обход условия `WHERE` - отрезать его, отметив как комментарий
- Выполнить несколько запросов одновременно, разделив их “;” (разделитель зависит от СУБД)
  - `?id=1;DROP+DATABASE+name`

# Что можно натворить?

- Получить доступ к структуре СУБД, исследую таблицу `information_schema`
- Получить доступ к файловой системе посредством ф-ций `load_file()`, `load data`
  - `union select load_file('/etc/passwd')`
- Выполнить произвольную команду на стороне сервера посредством ф-ций `EXEC`, `shell()` и других

# Защита от инъекций

- Любые пользовательские данные ненадёжны, пока мы не докажем обратное
- Пользовательский ввод нужно “экранировать” – `addslashes()`, `magic_quotes` и т.д.
- Все данные нужно фильтровать по “области допустимых значений”
  - Проверить на тип
  - Целочисленные – по диапазону
  - Строковые – по длине
- Избегать подстановки данных в запрос (заменить на `if`, `switch`)
- Использовать `prepared statements`, хранимые процедуры и проч.

# XSS: Cross-Site Scripting

- Межсайтовый скриптинг: встраивание в код сайта пользовательских скриптов
- Отправить форму  
`<script>alert('Hacker!')</script>` (или передать параметром)
- Получить доступ к cookies
- Модифицировать содержимое сайта (DOM)
- Перенаправлять пользователей
- DoS

# Защита от XSS

- Жёсткая фильтрация пользовательского ввода, вырезание всех запрещенных тегов `strip_tags()`, а также атрибутов у разрешенных (`onclick`)
- Преобразование `<>` “” в `&quot; &gt; &lt;` – `html_special_chars()`
- Запретить протокол `javascript:` в ссылках
- Запретить `eval()` в любом его проявлении

# Общие рекомендации по защите (веб-) приложений

- Регулярное создание резервных копий, как можно чаще! (каждые в 4 часа)
- Ведение логов всех критических операций: кто, что, где и когда изменял
- Разработка при помощи системы контроля версий git/svn
- Ограничение прав системного пользователя на выполнения операций с СУБД (в идеале – только SELECT и INSERT)

# Пользователи и пароли

- Шифрование паролей пользователей (salt)
- Требования к сложности паролей для всех пользователей
- Требовать от пользователя смены пароля при первом входе в систему по заранее сгенерированному паролю
- Периодическая смена паролей (через каждые  $Y$  дней) с историей паролей (предыдущие пароли нельзя переиспользовать в течение  $X$  дней)

# Сессии и cookies

- Менять идентификатор сессии при каждой авторизации, связывать идентификатор сессии с IP-адресом и строкой USER\_AGENT, не разрешать продолжения сессии после продолжительного периода бездействия пользователя
- Не использовать cookies для хранения сколь-либо важной информации (парли или их хеши)