Тема 20

Проблемы обеспечения безопасности приложений

Уязвимости приложений

Переполнение буфера

«Гонки»

Использование привилегий серверных компонент

Манипуляции с данными на клиентской стороне

Переполнение буфера - наиболее распространённая атака уровня приложений

Цель

Получение контроля над объектом атаки

Механизм реализации

Запуск кода на атакуемом узле

Местонахождение атакующего

В разных сегментах с объектом атаки

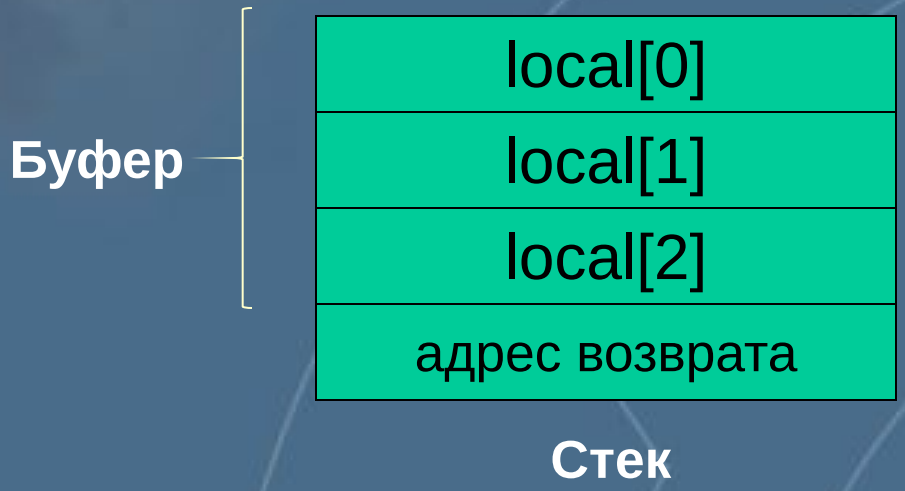
Используемые уязвимости

Ошибки реализации

Степень риска ***Высокая***

Переполнение стека

```
f_vulner()
{
char local[3]
...
...
}
```

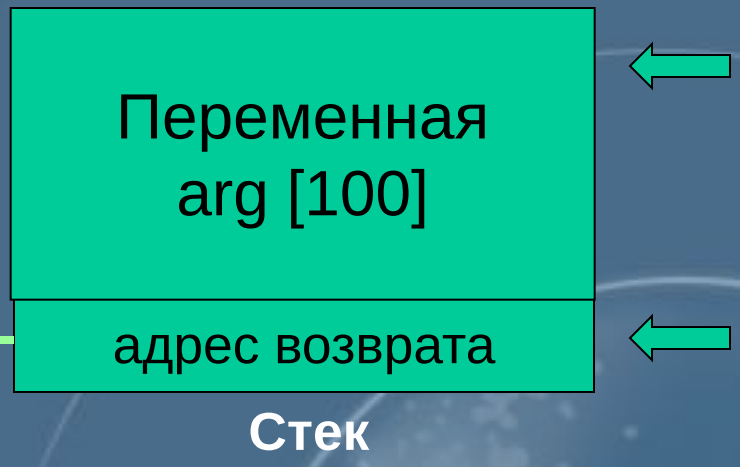


«Переполнение стека»

```
int f_vulner (char arg)
{
    char local[100]
    //обработка
    return 0
}
void main()
{
    char arg[200]
    gets (arg)
      .
      .
    f_vulner (arg)
    printf(arg)
    return 0
}
```

strcpy(local, arg)

Обычный ход выполнения программы



«Переполнение стека»

```
int f_vulner (char arg)
{
    char local[100]
    //обработка
    return 0
}
void main()
{
    char arg[200]
    gets (arg)
      .
      .
    f_vulner (arg)
    printf(arg)
    return 0
}
```

strcpy(local, arg)
Ошибка !

Переполнение стека

«Переполнение стека»

Вызов функций ядра (программное прерывание INT 0x80)

Вызов функций из модулей DLL

Использование функции «WinExec»

Использование переполнения стека

Причины переполнения буфера

Отсутствие необходимых проверок на корректность аргументов

Отсутствие средств вычисления длины буфера при работе с указателями

Последствия переполнения буфера

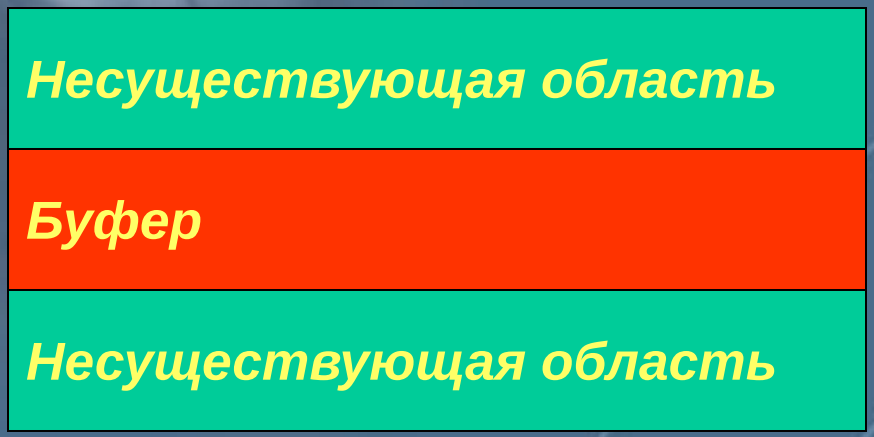
Чтение ячеек памяти, не принадлежащих массиву

Модификация ячеек памяти

- *Системные данные (адрес возврата и т. д.)*
- *Другие переменные*
- *Исполняемый код*
- *Несуществующая (свободная область)*

Предотвращение ошибок переполнения

Использование механизма структурных исключений



Использование языков программирования, делающих невозможным переполнение буфера

Использование «Heap» для указателей

Отказ от индикатора завершения

Методы защиты

Установка пакетов исправления

Исправление исходного кода с перекомпиляцией

Тестирование программ специальными утилитами

«Гонки»

Гонки можно определить как некорректное поведение программы, вызванное неожидаемой последовательностью зависящих друг от друга событий. Другими словами, разработчик программы ошибочно полагает, что одно событие всегда должно предшествовать другому, но реально может возникнуть ситуация, когда это будет не так.

Гонками также называются ситуации, при которых два или более процессов (потоков) обрабатывают разделяемые данные (файлы или переменные), и конечный результат зависит от соотношения скоростей процессов (потоков).

В общем случае, процесс (поток) не выполняется атомарно. Его выполнение может быть прервано, и другой процесс может выполнить свои действия между двумя любыми операции прерванного процесса. В защищённом приложении любая пара операций должна работать корректно, даже если между ними выполняется произвольный код других процессов (потоков).

«Гонки»

Проблемы, причиной которых являются гонки, можно поделить на две категории:

Вмешательство со стороны недоверенного процесса. В этом случае критичными являются либо последовательность выполнения операций, либо «атомарность» (неделимость) двух операций.

Вмешательство со стороны доверенного процесса (с точки зрения защищённого приложения). Такие ситуации называют взаимными блокировками, а также дедлоками (deadlocks), клинчами (clinch) или тупиками.

«Гонки» - пример уязвимости

Name

CAN-2003-0127 (under review)

Description

The kernel module loader in Linux kernel 2.2.x before 2.2.25, and 2.4.x before 2.4.21, allows local users to gain
root privileges by using ptrace to attach to a child process
that is spawned by the kernel.

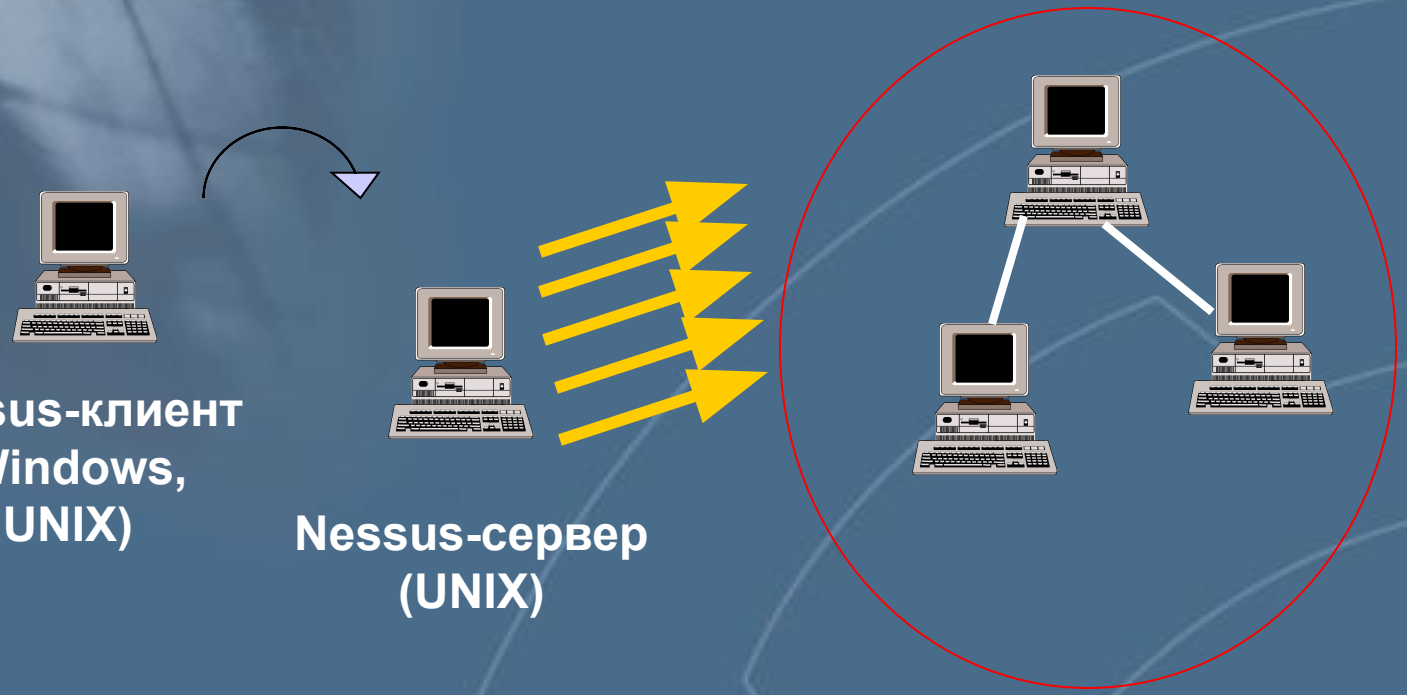
Практическая работа 25

Уязвимость загрузчика модулей ядра в ОС Linux

Тема 21

Сетевой сканер Nessus

Сетевой сканер Nessus

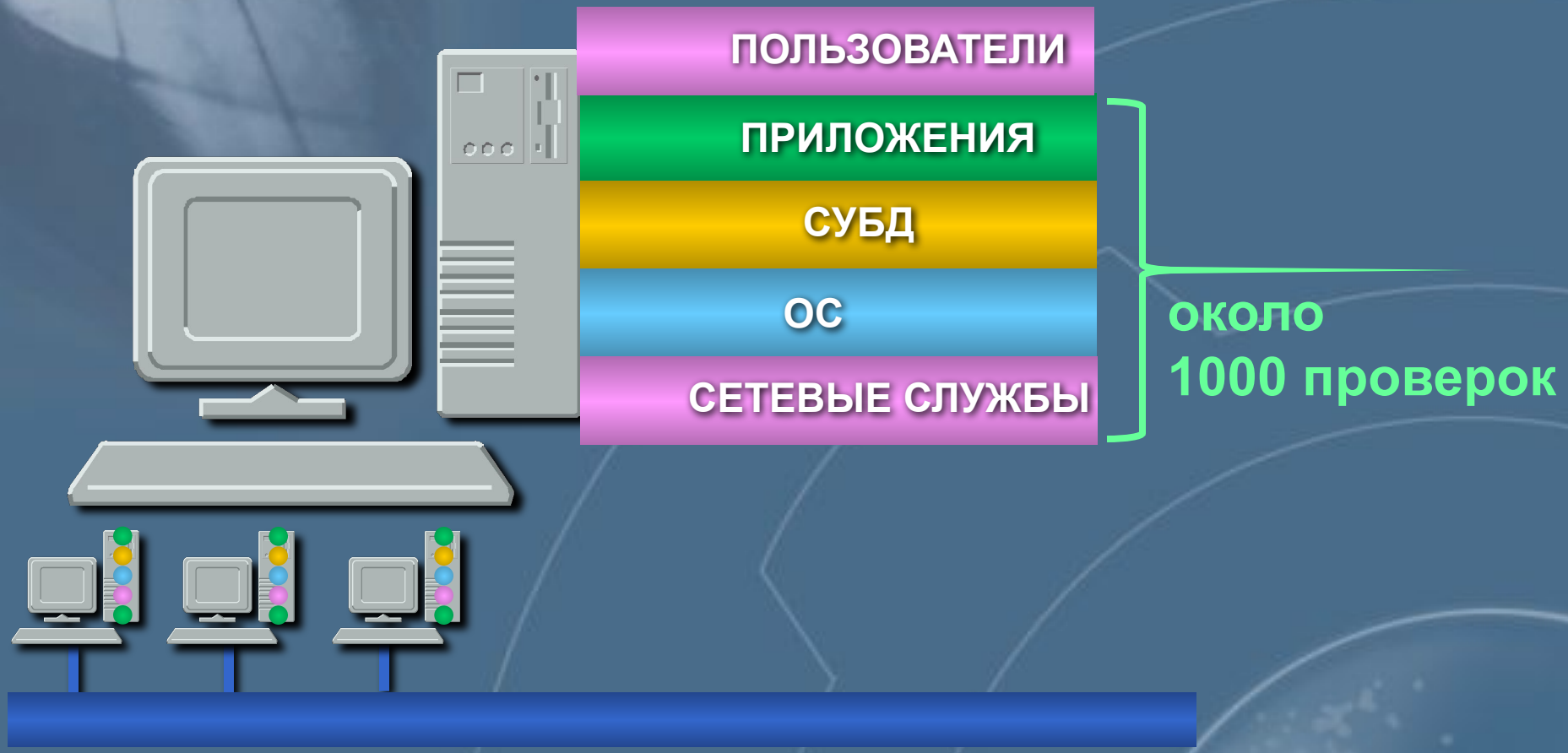


Nessus-клиент (Windows, UNIX)

Nessus-сервер (UNIX)

Сканируемая сеть

Сетевой сканер Nessus

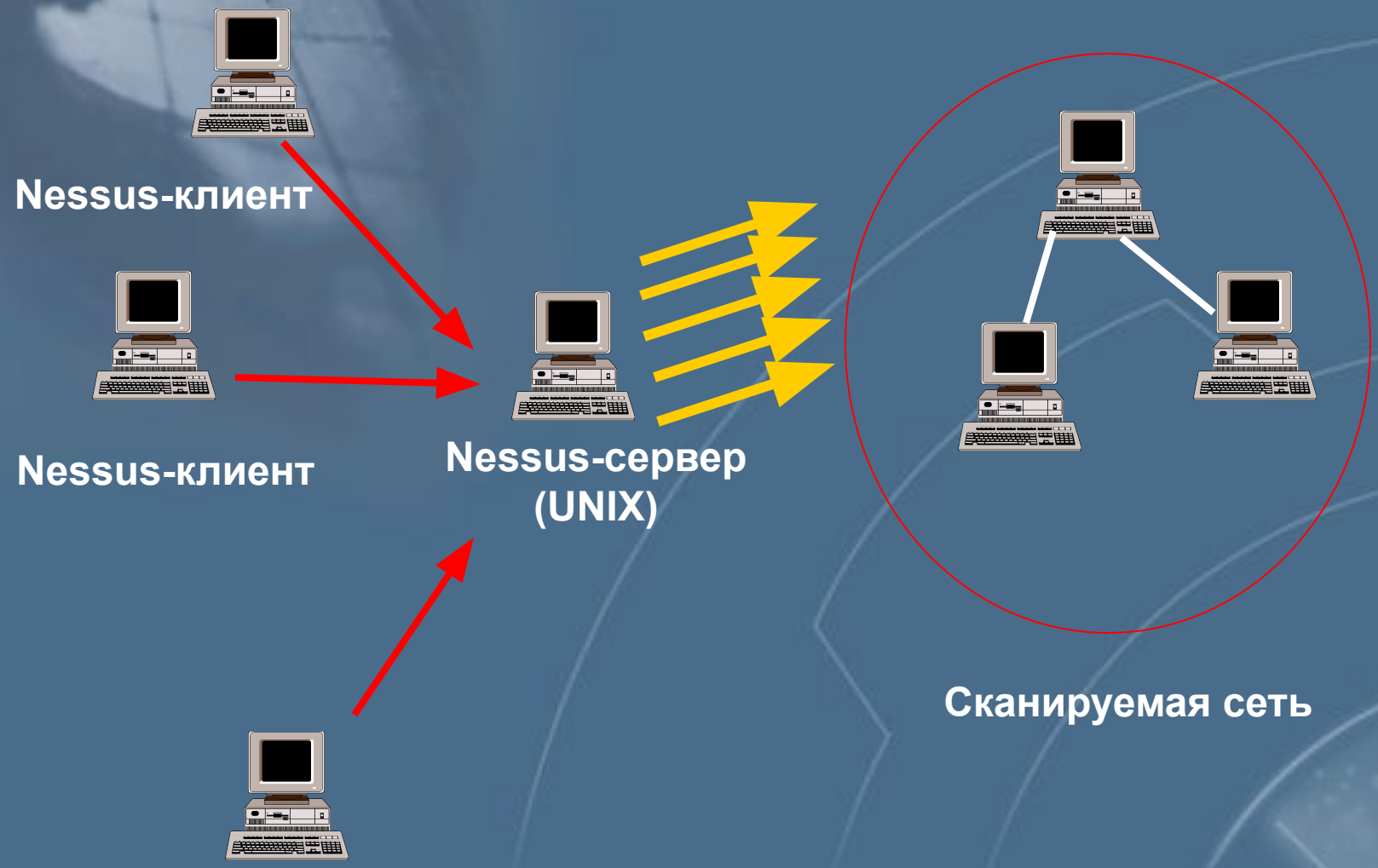


Сетевой сканер Nessus

- Модульная архитектура
- Язык описания атак NASL
- Система генерации отчётов
- Идентификация служб

Характеристики

Параллельное подключение и сканирование



Практическая работа 20

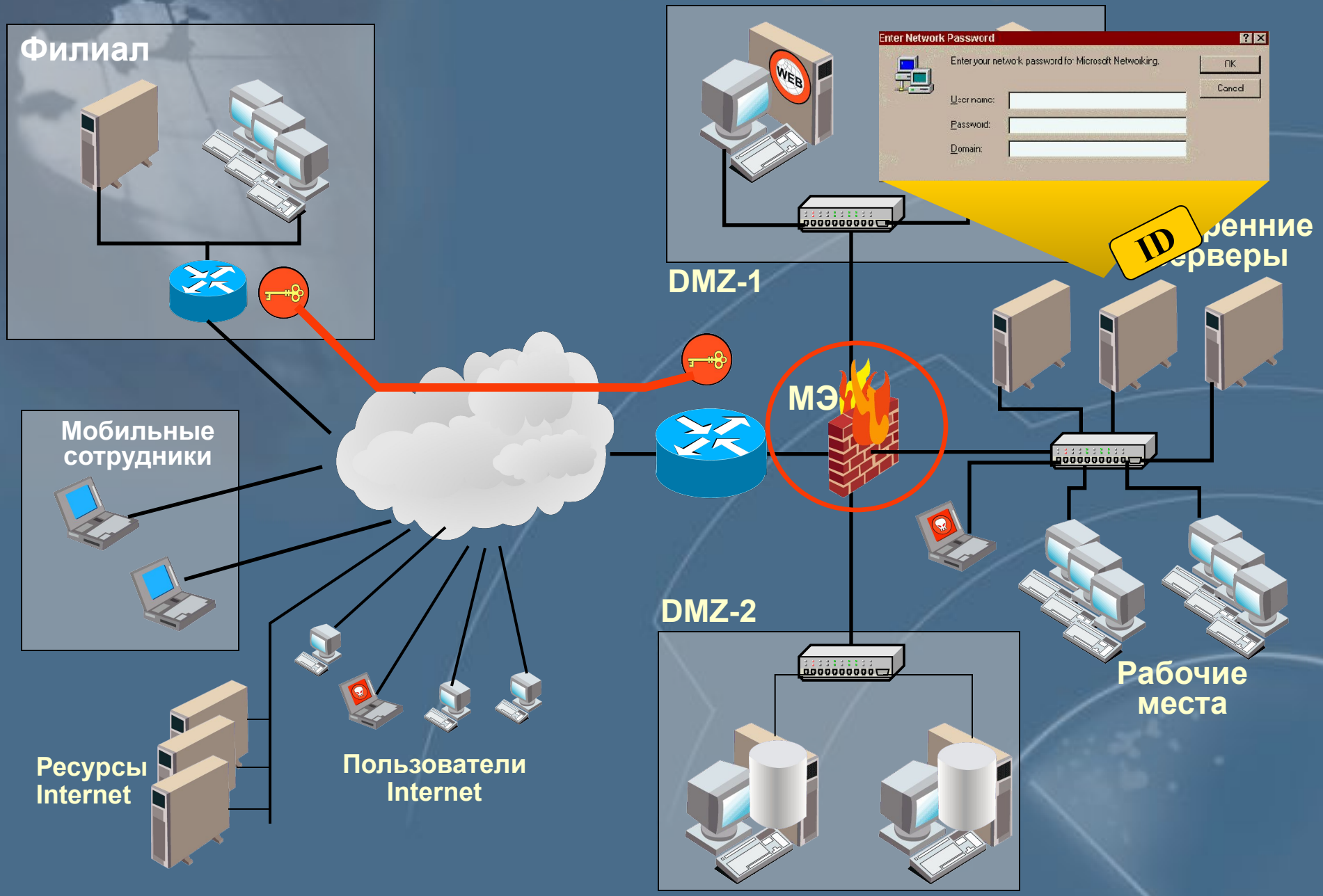
Изучение возможностей сканера Nessus

Обеспечение безопасности сетей - - итог

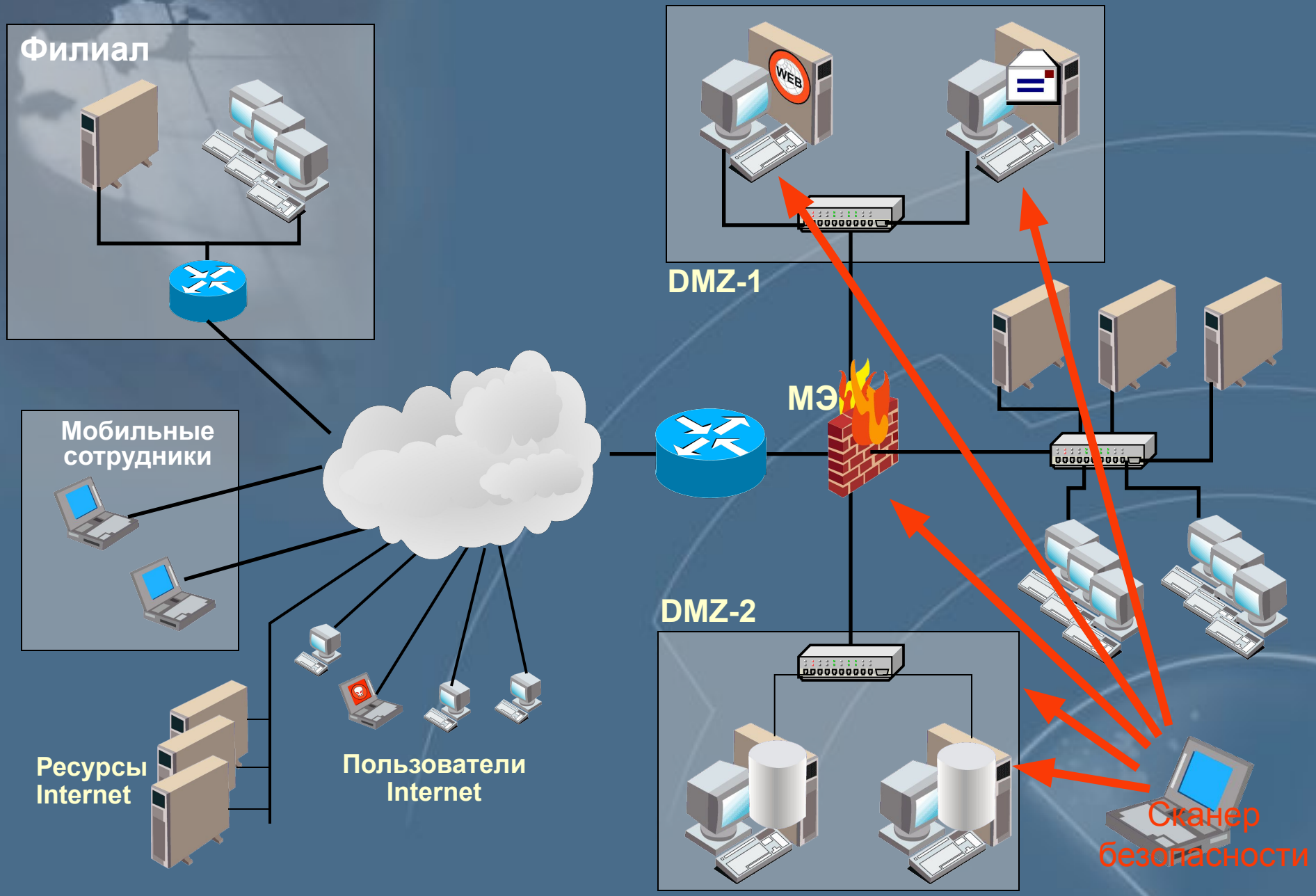
Для защиты сети необходимо использовать комплекс средств защиты, включающий в себя:

- ***Средства защиты узлов и ЛВС, обеспечивающие аутентификацию, разграничение доступа, шифрование и т.д.***
- ***Средства анализа защищённости и устранения уязвимостей***
- ***Средства обнаружения атак***

Средства защиты периметра



Средства анализа защищённости



Средства обнаружения атак