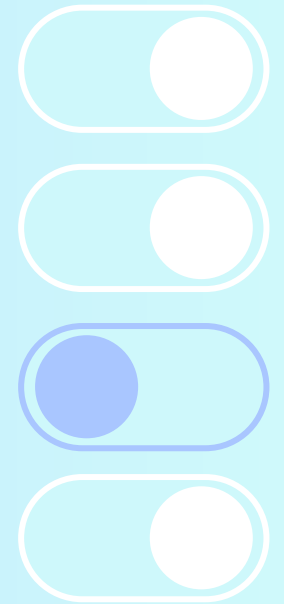


Кауч

Управление уязвимостями конфигураций:
импортозамещаем  **nessus** с умом

Спикер:

Кирилл Евтушенко,
генеральный директор Кауч



.audit – основной способ проверок настроек в Nessus

```
1 <check_type:"Unix">
2
3
4 <if>
5   <condition type:"AND">
6     <custom_item>
7       type      : CMD_EXEC
8       description : "Check if Linux kernel installed"
9       cmd       : "uname -s"
10      expect    : "^(?i)Linux"
11    </custom_item>
12  </condition>
13
14  <then>
15    <if>
16      <condition type:"OR">
17
18        <custom_item>
19          type      : CMD_EXEC
20          description : "Check if OpenSSH enabled"
21          cmd       : "systemctl is-enabled sshd-server 2>/dev/null"
22          expect    : "enabled"
23        </custom_item>
24
25        <custom_item>
26          type      : CMD_EXEC
27          description : "Check sshd version"
28          cmd       : "sshd -v 2>/dev/null"
29          expect    : "(?i)sshd"
30        </custom_item>
31      </condition>
32
33    </if>
34  </then>
35  <custom_item>
36    type      : CMD_EXEC
37    description : "Check SSH LogLevel via sshd -T and first LogLevel in main sshd_config"
38    info      : "Checks the effective LogLevel from sshd -T and the first LogLevel directive found in the main config file only."
39    solution  : "Set LogLevel INFO or VERBOSE in the main sshd_config file."
40    cmd       : "{
41      if [[ $(cat /etc/*-release 2>/dev/null | grep '^NAME' | tr -d '\n') =~ ALT$ ]]; then
42        ...
43      fi
44    }"
45    expect    : "PASS"
46  </custom_item>
```

Декларативный подход

- DSL (domain-specific language) Tenable
- Текстовый формат с «псевдо-блочной» структурой
- Похоже на смесь:
 - INI
 - DSL
 - Описания конфигураций
- Типы проверок:
 - FILE_CONTENT_CHECK
 - SERVICE_CHECK
 - REG_CHECK
 - FILE_CHECK
 - ...
 - AUDIT_POWERSHEL
 - **CMD_EXEC**

Пример: простой конфиг openssh

```
1 # This is the sshd server system-wide configuration file. See
2 # sshd_config(5) for more information.
3
4 Port 22
5 AddressFamily any
6 ListenAddress 0.0.0.0
7 ListenAddress ::
8
9 HostKey /etc/ssh/ssh_host_rsa_key
10 HostKey /etc/ssh/ssh_host_ecdsa_key
11 HostKey /etc/ssh/ssh_host_ed25519_key
12
13 # LogLevel
14 LogLevel verbose
15
16 # Ciphers and keying
17 RekeyLimit default none
18
19 ...
```

Пример: openssh – audit

```
1 <custom_item>
2   type      : FILE_CONTENT_CHECK
3   description : "Ensure SSH LogLevel is set to INFO or VERBOSE"
4   file      : "/etc/ssh/sshd_config"
5   regex     : "^[[:space:]]*LogLevel[[:space:]]+(INFO|VERBOSE) "
6 </custom_item>
```

Или

```
1 <custom_item>
2   type      : CMD_EXEC
3   description : "Ensure SSH LogLevel is set to INFO or VERBOSE"
4   cmd       : "sshd -T 2>/dev/null | grep -i '^loglevel'"
5   expect    : "(?i)loglevel[[:space:]]+(info|verbose) "
6 </custom_item>
```

openssh – audit сына маминной подруги

```

1  ...
2  cmd : "{
3  if [[ $(cat /etc/*-release 2>/dev/null | grep '^NAME' | tr -d '\n') =~ ALT$ ]]; then
4      base_dir='/etc/openssh'
5  else
6      base_dir='/etc/ssh'
7  fi
8
9  config_file="\$base_dir/sshd_config\"
10 status=1
11 output=''
12
13 sshd_t_conf=$(sshd -T 2>&1)
14 if [[ $? -ne 0 ]]; then
15     output+="sshd -T error: $sshd_t_conf"$'\n'
16     status=0
17 else
18     output+="sshd -T output:"$'\n'
19     while IFS= read -r line; do
20         line=$(printf '%s' \"$line\" | sed 's/^[[:space:]]+//; s/[[:space:]]+$//')
21         if [[ \"$line\" =~ ^LogLevel[[:space:]]+([^\s:]+)$ ]]; then
22             value_lc=$(printf '%s' \"${BASH_REMATCH[1]}\" | tr '[:upper:]' '[:lower:]')
23             output+=" $line"$'\n'
24             if [[ \"$value_lc\" != 'info' && \"$value_lc\" != 'verbose' ]]; then
25                 status=0
26             fi
27         fi
28     done <<< \"$sshd_t_conf\"
29 fi
30
31 output+=$'\nmain config:'$'\n'
32
33 first_param=''
34 while IFS= read -r line; do
35     stripped=$(printf '%s' \"$line\" | sed 's/^[[:space:]]+//; s/[[:space:]]+$//')
36     [[ -z \"$stripped\" ]] && continue
37     [[ \"$stripped\" =~ ^# ]] && continue
38
39     if [[ \"$stripped\" =~ ^[Ll][Oo][Gg][Ll][Ee][Vv][Ee][Ll][[:space:]]+([^\s:]+)$ ]]; then
40         first_param=\"$stripped\"
41         value_lc=$(printf '%s' \"${BASH_REMATCH[1]}\" | tr '[:upper:]' '[:lower:]')
42         output+=" $stripped"$'\n'
43         if [[ \"$value_lc\" != 'info' && \"$value_lc\" != 'verbose' ]]; then
44             status=0
45         fi
46     fi
47     break
48 done < \"$config_file\"
49
50 if [[ -z \"$first_param\" ]]; then
51     output+=' LogLevel directive not found in main config file'$'\n'
52 fi
53
54 printf '%d' \"$status\"
55 }"
56 expect : "^1$"
57 ...

```

openssh: повышаем обороты

```
1 # This is the sshd server system-wide configuration file.  See
2 # sshd_config(5) for more information.
3
4 Port 22
5 AddressFamily any
6 ListenAddress 0.0.0.0
7 ListenAddress ::
8
9 include /etc/ssh/sshd_config.d/10-extra.conf
10
11 # LogLevel
12 LogLevel verbose
13
14 HostKey /etc/ssh/ssh_host_rsa_key
15 HostKey /etc/ssh/ssh_host_ecdsa_key
16 HostKey /etc/ssh/ssh_host_ed25519_key
17
18 include /etc/ssh/sshd_config.d/20-extra.conf
19
20 # Ciphers and keying
21 RekeyLimit default none
22 ...
```

openssh: еще выше

```
1 /etc/ssh/sshd_config.d/10-extra.conf:
2
3 LogLevel info
4 DisableForwarding yes
5 PubKeyAuthentication yes
6
7 Match User DevUser
8     LogLevel Info
9     Include /etc/ssh/match_includes.d/30-extra.conf
10
11 Match User AdminUser Address 10.10.10.5
12     PermitRootLogin yes
13     AllowTcpForwarding yes
14     LogLevel fatal
15
```

openssh: game over

```
87     while IFS= read -r line; do
88         if [[ \"$line\" =~ ^[[:space:]]*[Mm]atch[[:space:]] ]]; then
89             include_type='IN-MATCH'
90         fi
91
92         if [[ \"$line\" =~ ^[[:space:]]*[Ii]nclude[[:space:]] ]]; then
93             local included_files=()
94             local g search_dir each real_each
95             local -a tmp=()
96
97             while IFS= read -r g; do
98                 [[ -z \"$g\" ]] && continue
99
100                if [[ $g != /* ]]; then
101                    g=\"$base_dir/$g\"
102                fi
103
104                search_dir=$(dirname \"${g%%[?*\\]*}\")
105                [[ -d \"$search_dir\" ]] || continue
106
107                mapfile -d '' tmp <<(find \"$search_dir\" -path \"$g\" -print0 2>/dev/null)
108
109                for each in \"${tmp[@]}\"; do
110                    real_each=$(canon_path \"$each\")
111                    included_files+=(\"$real_each\")
112
113                done
114            done
115
116            if [[ \"$param_global_found\" -eq 0 ]]; then
117                output+=\"$'\n\nLogLevel directive not found in global config. Base value: loglevel INFO\n'
118            fi
119
120            printf '%d' \"$status\"
121        }\"
122        expect      : \"^1$\"
123    </custom_item>
```

Недостатки .audit

- Нет:
 - Циклов
 - Функций
 - Удобных условий
 - Объектов
 - Связей между проверками
 - Инклюдов/модулей/переиспользования
 - Разбора структурированных форматов
- Классический для сканеров match-подход
- Проблемы с:
 - include
 - override
 - runtime/детекты

Что приходится делать

- Писать очень-очень много кода на баше или пауэршелле
- Задействовать установленные на проверяемых серверах утилиты (и даже устанавливать их специально)
 - Парсинг XML, JSON, YAML и пр. (иначе – grep 🧐)
 - Подключение к БД или другому ПО
- Вызывать локальные интерпретаторы (python, perl...) для облегчения bash-нагрузки

Проблемы: отсутствие (или присутствие 🧐) нужных прав, необходимых утилит, конкретных версий ПО/библиотек и пр.

Не все проблемы

- Compliance Policy / Scan может состоять из нескольких audit-файлов, но каждый «коннектор» в отдельном audit-файле
 - Файлы аудита не связаны между собой
 - Сложно писать проверки на нескольких уровнях (СУБД: есть параметры в БД и ФС)
- Проблемы при работе с неизвестным Nessus прикладным ПО, требующим аутентификации: сложности с хранением и передачей учетных данных (нужен NASL, возможности ограничены)
- Проверки де-факто выполняются на стороне сервера

NASL

- Высокий порог входа/нужно учить язык, мало документации
- Сложное описание логики проверок, контекст (КВ, креды, типы проверок)
- Часто, фактически, обертка над баш-кодом
- Логика на NASL часто получается сложнее, чем на баше
- Минимум возможностей по отладке и обработке ошибок
- Формально умеет, но на деле неполноценно работает со структурированными форматами
- Уступает современным языкам программирования примерно во всем

А теперь
переезжаем
со всем ЭТИМ
багажом



Проверки в Кауче: наглядно

Редактировать требование

Пожалуйста, используйте эту функцию с осторожностью!

Номер: *

Заголовок: *

Описание: *

Команда `su` позволяет пользователю запускать команду или оболочку от имени другого пользователя. Обычно команду `su` может выполнить любой пользователь, однако доступ к ней может контролироваться с помощью модуля `PAM pam_wheel.so`. Ограничение использования `su` и использование вместо него `sudo` позволяет системным администраторам лучше контролировать повышение привилегий пользователей для выполнения команд. Утилита `sudo` также обеспечивает лучший механизм ведения журнала и аудита, поскольку она может регистрировать каждую команду, выполненную через `sudo`, тогда как `su` может записывать только факт того, что пользователь выполнил программу `su`. Необходимо использовать `sudo` для запуска команд с повышенными привилегиями и максимально ограничивать доступ к запуску `su`.

Аудит: *

Проверьте, что в файле `/etc/pam.d/su` присутствует строка ограничения запуска команды `su` в соответствии с требуемым значением. Проверьте, что группа `wheel` включает только тех пользователей, которым явно разрешен прямой запуск команды `su`. Список членов группы `wheel` можно получить с помощью следующей команды:

```
# lid -g wheel
```

Требуемое значение: `auth required pam_wheel.so use_uid`

Устранение: *

Исключите из группы `wheel` всех пользователей, у которых нет необходимости прямого запуска команды `su`. Во многих случаях даже при необходимости работы от имени другой учетной записи пользователь может использовать комбинацию «`sudo su`» и необходимость в прямом запуске `su` отсутствует. Раскомментируйте или добавьте в файл `/etc/pam.d/su` строку для контроля запуска команды `su` в соответствии с требуемым значением.

Метод:

Риск:

Высокий

Команда сбора:

```
1 egrep "^\s*auth\s+required\s+pam_wheel\.so" /etc/pam.d/su 2>&1
```

Скрипт валидации: * [? ТЕСТ](#)

```
def check(text):  
1     x = re.split(r'_#\[\r?\n?', text)  
2     return int(bool(re.search('\s*auth\s+required\s+pam_wheel\.so\s*(.*)?use_uid(\s|#|$)', x[0], re.MULTILINE))), x[0] or 'pam_wheel.so in not configured'
```

Команда конфигурации:

```
1 PTF=/etc/pam.d/su;  
2 grep -E '\s*auth\s+required\s+pam_wheel\.so' $PTF || sed -ri '0,/^(\s*auth\s+sufficient\s+pam_rootok\.so/s/^(\s*auth\s+sufficient\s+pam_rootok\.so.*$&\nauth required pam_wheel.so use_uid/' $PTF;  
3 [[ -z $(grep -E '\s*auth\s+required\s+pam_wheel\.so.*\s+use_uid(\s|#|$)' $PTF) ]] && sed -ri 's/^(\s*(auth\s+required\s+pam_wheel\.so.*$)/\1 use_uid/' $PTF || true;
```

Копировать

Сохранить

Кауч

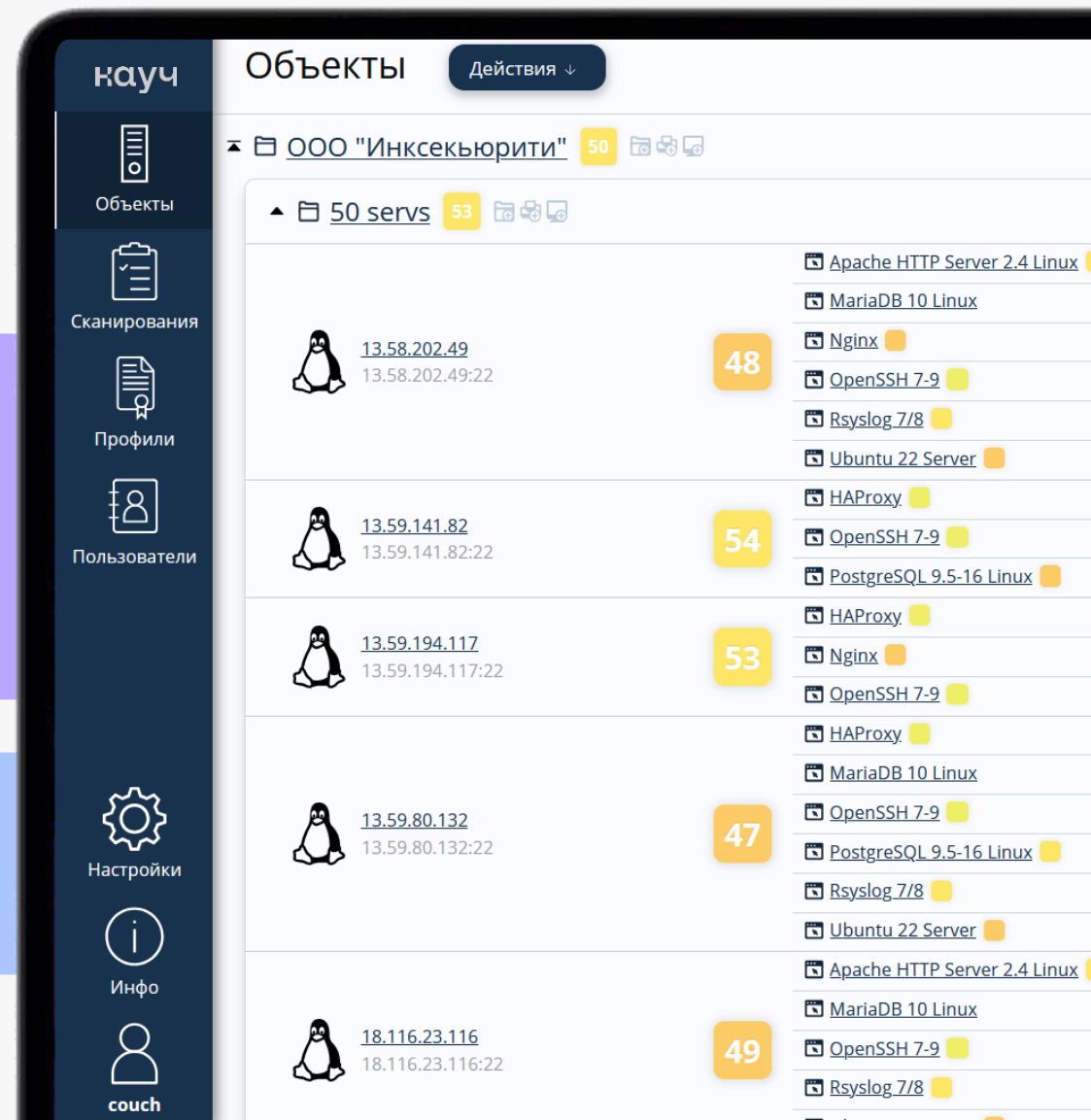
All-in-one платформа для управления безопасностью конфигураций ↗

Единая среда для ИБ и ИТ и всех операций, связанных с управлением безопасностью настроек ИТ-ресурсов

Автоматизированная настройка ресурсов по клику или офлайн с помощью готовых команд и скриптов, доступных «из коробки»

Проверка соответствия ресурсов по клику, расписанию или офлайн с помощью мощного встроенного сканера конфигураций

Гибкое и простое создание политик безопасности любой сложности



couch.ru

Приходите
к нам на стенд...

...или на сайт
couch.ru

кауч



Вот с кем мы уже работаем

