

Воркшоп: подготовка окружения пентестера

1. Из чего складывается рабочее окружение

- Железо: ноутбук, стационарный ПК, виртуальные машины, VPS, mini-PC.
- ОС: Windows, macOS, Linux, Kali Linux и отдельная Windows VM под AD/PowerShell-задачи.
- Софт: open-source, enterprise, cloud, on-premise и маленькие собственные автоматизации.
- Рабочее место: дом, офис, дорога, удаленный сервер, выездной проект.
- Интернет: домашний, мобильный, VPS/серверный канал.

Ключевой тезис: рабочее окружение пентестера — это не “поставить Kali”, а собрать устойчивую операционную среду под конкретный тип задач.

- Новички обычно упираются не в сложность атак, а в грязное окружение: виртуализация, Wi-Fi, сертификаты, Python-зоопарк, абузы от хостера и хаос в результатах.
- Правильно собранная среда экономит больше времени, чем установка еще десяти новых инструментов.

2. Референсные сборки, которые можно рекомендовать студентам

Уровень	Железо / ОС	Состав	Для чего подходит
Базовый старт	Ноутбук/ПК + host OS	Kali VM + отдельный браузерный профиль + Burp Community + nmap + ffuf + tmux + pipx	Первые лабы, web, recon, обучение, домашняя практика.
Средний уровень	Ноутбук/ПК + host OS	Kali VM + Windows VM + VPS + базовый набор API-ключей + свои маленькие wrapper-скрипты	Когда начинаются длинные практики, internal/AD, длительные задачи и удаленная работа.
Домашняя lab-сборка	PC или mini-PC	Несколько VM, isolated lab network, snapshot discipline, VPN/доступ до VPS	Стабильная домашняя практика, полигон, эксперименты, длительные стенды.

3. Практика №1 — Kali Linux в виртуальной машине

Задача: за 10–12 минут показать базовую, воспроизводимую и безопасную для обучения точку входа: Kali VM с snapshot и отдельной лабораторной сетью.

Официальные ссылки:

[Get Kali \(официальные образы и готовые VM\)](#)

[Импорт готовой Kali VM в VirtualBox](#)

[Официальные образы Kali и VM для VMware/VirtualBox](#)

- Для большинства новичков лучший старт — host OS + Kali VM. Это дает snapshots, быстрый откат и не требует ломать основную систему.
- Не качайте образы с посторонних сайтов. Используйте только официальные страницы Kali.
- Не бриджите демонстрационную VM напрямую в домашнюю или корпоративную сеть без понимания, зачем это нужно.

```
# После первого входа в Kali VM
whoami
hostnamectl
ip a
sudo passwd kali
sudo apt update && sudo apt full-upgrade -y
sudo apt autoremove -y
sudo reboot

# Базовая структура сети для учебной VM
# Adapter 1: NAT -> интернет для обновлений и загрузки инструментов
# Adapter 2: Host-Only / Internal Network -> учебный сегмент для стенда и сканирования
```

Ключевой тезис:

- “закрыв ноутбук — остановил длительный scan/recon”. Это хороший мостик к теме VPS и автономности.
- Wi-Fi внутри VM и шумная активность из домашней сети часто дают не идеальную картинку: падает скорость, деградирует роутер, возникают ложные проблемы с качеством канала.

4. Практика №2 — минимальный Debian/VPS для удаленных задач

Задача: показать, как выглядит аккуратная удаленная коробка для recon, always-on задач и хранения небольших автоматизаций.

Официальная база по hardening Debian: [Securing Debian Manual](#)

- VPS хорош там, где ноутбук неудобен: долгие задачи, фоновый сбор данных, удаленный доступ, простые API-обвязки.
- VPS — не магическая анонимность. У хостера есть abuse-процедуры, а неаккуратная активность быстро приводит к жалобам и блокировкам.
- Не используйте чужие или скомпрометированные VPS. Это незаконно и организационно токсично.

```
# Первичный вход на свежий Debian/VPS
ssh root@<VPS_IP>
apt update && apt full-upgrade -y
apt install -y sudo curl wget git ufw fail2ban openssh-server ca-certificates
adduser student
usermod -aG sudo student
rsync --archive --chown=student:student ~/.ssh /home/student

# В новом сеансе проверьте вход под обычным пользователем
ssh student@<VPS_IP>
sudo -v
```

```

whoami
hostnamectl

# Минимальный firewall и автозапуск важных сервисов
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow OpenSSH
sudo ufw enable
sudo systemctl enable --now ssh
sudo systemctl enable --now fail2ban

# SSH-hardening: сначала убедитесь, что вход по ключу работает,
# и только потом отключайте парольную аутентификацию.
ssh-keygen -t ed25519 -C "workshop@student"
ssh-copy-id student@<VPS_IP>

sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak
sudo sed -i 's/^#\?PasswordAuthentication .*/PasswordAuthentication no/'
/etc/ssh/sshd_config
sudo sed -i 's/^#\?PermitRootLogin .*/PermitRootLogin no/' /etc/ssh/sshd_config
sudo systemctl restart ssh

```

Ключевой тезис: на VPS главное — не “поставить побольше тулов”, а быстро сделать понятную и не хрупкую базу: user/sudo, SSH по ключам, firewall, обновления, контроль сервисов.

5. Практика №3 — SSH, workspace и базовая hygiene

Задача: собрать удобный рабочий baseline, чтобы студенту было где хранить заметки, артефакты и команды.

```

# Универсальный набор для Kali / Debian
sudo apt install -y tmux git curl wget jq yq ripgrep fd-find bat tree unzip zip python3-pip
python3-venv pipx
pipx ensurepath

# Структура каталогов под проект/лабораторную/клиентский тест
mkdir -p ~/work/{notes,loot,scans,pcaps,reports,scripts,targetsets}
mkdir -p ~/work/demo-project/{notes,loot,scans,pcaps,reports,scripts,targetsets}
tree -L 2 ~/work

# tmux: минимальный набор команд для показа студентам
# Новый сеанс
tmux new -s pentest
# Отсоединиться
Ctrl+b d
# Вернуться
tmux attach -t pentest
# Новое окно
Ctrl+b c
# Разделить панель по вертикали
Ctrl+b %
# Разделить панель по горизонтали
Ctrl+b "

# Удобный ~/.ssh/config для нескольких VPS / lab-host'ов
mkdir -p ~/.ssh && chmod 700 ~/.ssh
cat >> ~/.ssh/config <<'EOF'
Host lab-vps
    HostName <VPS_IP>
    User student

```

```
IdentityFile ~/.ssh/id_ed25519
ServerAliveInterval 30
ServerAliveCountMax 3

Host kali-lab
    HostName 192.168.56.101
    User kali
EOF
chmod 600 ~/.ssh/config
```

Ключевой тезис:

- Структура каталогов и тмх важнее, чем еще 20 модных утилит.
- Хаос в заметках и выводах — одна из главных причин, почему новичок не может воспроизвести собственный результат даже через сутки.

6. Практика №4 — браузерный профиль и Burp Suite Community

Задача: показать web-ready окружение без загрязнения основного браузера.

Официальные ссылки:

[Burp Suite documentation](#)

[Установка CA-сертификата Burp в браузер](#)

- Не ставьте Burp CA в daily browser. Делайте отдельный тестовый профиль.
- Burp Community достаточно для первого этапа: перехват, Repeater, базовый анализ HTTP(S)-трафика.
- Эта настройка — не “магия web hacking”, а инструментальная hygiene.

```
# Установка Burp в Kali обычно доступна из репозитория
sudo apt install -y burpsuite firefox-esr

# Запуск Firefox с отдельным профилем (пример)
firefox-esr --ProfileManager

# Создайте профиль, например: burp-lab
# Затем запускайте его отдельно для веб-демо и лабораторных
```

Пошагово в демо: 1) запускаем Burp, 2) настраиваем browser проху на 127.0.0.1:8080, 3) открываем http://burp, 4) экспортируем/устанавливаем CA-сертификат только в тестовый профиль, 5) показываем перехват собственного lab-трафика.

Ключевой тезис:

- Человек, который “слышал про Burp”, и человек, который неделями проходит web-лабы в удобной среде, — это разные уровни зрелости.
- Вот такие бытовые настройки и превращают интерес к вебу в устойчивую практику.

7. Практика №5 — Python, pipx, uv, Go и API-ключи

Задача: подготовить внятный toolchain для CLI-инструментов и маленьких собственных скриптов.

Официальные ссылки:

[pipx install docs](#)

[uv installation docs](#)

[Go install docs](#)

[Shodan API docs](#)

[SecurityTrails API docs](#)

[VirusTotal API overview](#)

```
# Базовая подготовка Python CLI-среды
sudo apt install -y python3 python3-pip python3-venv pipx
pipx ensurepath
hash -r

# uv: быстрый менеджер для локальных Python-проектов и утилит
curl -LsSf https://astral.sh/uv/install.sh | sh
# после перезапуска shell:
uv --version

# Go: простой путь через пакетный менеджер
sudo apt install -y golang-go
# либо используйте официальный способ с go.dev

go version
go env GOPATH
mkdir -p "$(go env GOPATH)/bin"
grep -q 'GOPATH' ~/.bashrc || cat >> ~/.bashrc <<'EOF'
export PATH="$PATH:$(go env GOPATH)/bin"
EOF

# Пример хранения API-ключей локально, без git и без shell history
mkdir -p ~/.config/workshop-secrets
chmod 700 ~/.config/workshop-secrets
cat > ~/.config/workshop-secrets/recon.env <<'EOF'
export SHODAN_API_KEY='PUT_YOUR_KEY_HERE'
export SECURITYTRAILS_API_KEY='PUT_YOUR_KEY_HERE'
export VT_API_KEY='PUT_YOUR_KEY_HERE'
EOF
chmod 600 ~/.config/workshop-secrets/recon.env

# загружать при необходимости
source ~/.config/workshop-secrets/recon.env
```

- API-ключи не должны попадать в git, скриншоты, dotfiles-репозитории и shell history.
- pipx — для отдельных CLI-приложений, uv/venv — для собственных маленьких проектов, Go — для части современного гесоп-стека.
- Нужно быть не “разработчиком продукта”, а уметь писать маленькие прикладные glue-скрипты под свои повторяющиеся задачи.

8. Практика №6 — установка популярного набора инструментов

Задача: не превратить машину в свалку, а поставить минимум по ролям: recon, web, network, utility.

```
# Базовый набор из репозитория Kali/Debian
sudo apt install -y nmap ffuf amass tcpdump tshark wireshark sqlmap seclists jq yq git tmux
rlwrap

# Несколько популярных Go-инструментов (если нужен ProjectDiscovery stack)
go install github.com/projectdiscovery/subfinder/v2/cmd/subfinder@latest
go install github.com/projectdiscovery/httpx/cmd/httpx@latest
go install github.com/projectdiscovery/nuclei/v3/cmd/nuclei@latest

# Пример Python CLI через pipx
pipx install semgrep
pipx list
```

Ключевой тезис: ставим по ролям, а не “всё сразу”. Сначала — стабильный минимум; дальше — добавляем инструменты под реальные потребности.

Чего не делать

- Не тащить сомнительные сборки и пиратский offensive-софт: вы не контролируете бинарь и не знаете, что он делает.
- Не жить в режиме “сейчас поставлю 200 инструментов на всякий случай”.
- Не думать, что одна Kali решает все: под AD и native Windows-задачи отдельная Windows VM остается полезной.

9. Практика №7 — как работать с результатами, а не терять их

Задача: показать, как сохранять вывод в файлы и организовывать результат так, чтобы его можно было найти, прочитать и использовать в отчете.

```
cd ~/work/demo-project
mkdir -p scans notes loot reports

# Пример: складываем результаты в понятные файлы
nmap -sn 192.168.56.0/24 -oA scans/01_discovery
nmap -sS -Pn -T3 --top-ports 100 --max-rate 300 192.168.56.101 -oA scans/02_top100
nmap -sV --version-light --script safe 192.168.56.101 -oA scans/03_service_safe

# И делаем заметки сразу, а не “потом из памяти”
cat >> notes/2026-04-01_demo.md <<'EOF'
# Demo notes
- Host found: 192.168.56.101
- Safe scan only
- Need follow-up: verify web service manually
EOF
```

Ключевой тезис:

- Разница между “команда была в истории shell” и “результат сохранен в скан-файлах плюс заметка о контексте”.
- В реальной работе именно организация артефактов отличает пригодный к отчету результат от разрозненного хаоса.

10. Практика №8 — мини-автоматизация без переусложнения

Задача: показать, что автоматизация на старте — это не полноценный фреймворк, а 5–20 строк shell/python, которые экономят время.

```
# Пример shell-функции для быстрого старта нового проекта
cat >> ~/.bashrc <<'EOF'
mkpentest() {
    local name="$1"
    mkdir -p ~/work/"$name"/{notes,loot,scans,pcaps,reports,scripts,targets}
    cd ~/work/"$name" || return
    printf '# %s\n\n- scope:\n- targets:\n- notes:\n' "$name" > notes/README.md
}
EOF
source ~/.bashrc

# Использование
mkpentest acme-lab

# Мини-пример Python wrapper для чтения env и запуска своей логики
mkdir -p ~/work/demo-project/scripts
cat > ~/work/demo-project/scripts/check_keys.py <<'EOF'
import os
for name in ["SHODAN_API_KEY", "SECURITYTRAILS_API_KEY", "VT_API_KEY"]:
    print(f"{name}: {'set' if os.getenv(name) else 'missing'}")
EOF
python3 ~/work/demo-project/scripts/check_keys.py
```

Ключевой тезис: свой код нужен не для того, чтобы “переписать Nmap”, а чтобы склеить рутину, нормализовать вывод и не повторять руками одно и то же.

11. Практика №9 — безопасное lab-only сканирование

Задача: показать мягкий профиль исследования учебного сегмента и проговорить, что именно делает scan безопаснее и понятнее.

Официальная база по Nmap: [Nmap usage / reference](#) | [Nmap book / docs](#)

- Этот блок выполняется только на своем лаб-сегменте или на стенде с явным разрешением.
- Смысл демо — показать порядок действий: discovery -> ограниченный портовый срез -> мягкое service detection -> сохранение результатов.
- Слишком агрессивные тайминги и высокая скорость не делают новичка профессионалом. Часто они просто увеличивают шум, снижают точность и создают ненужные проблемы.

```
# 1) Ищем живые хосты в учебном сегменте
nmap -sn 192.168.56.0/24 -oA scans/01_discovery

# 2) Ограниченный TCP-срез по топ-100 портам, умеренный темп
nmap -sS -Pn -T3 --top-ports 100 --max-rate 300 192.168.56.101 -oA scans/02_top100

# 3) Легкое определение сервисов + только safe-скрипты NSE
nmap -sV --version-light --script safe 192.168.56.101 -oA scans/03_service_safe
```

Ключевой тезис:

Анти-кейсы

- Неправильный профиль сканирования в неподходящее время может создать operational issue даже без “эксплуатации”.
- MITM, spoofing, перебор паролей и Wi-Fi-демо — не вещи для романтизации в ознакомительном воркшопе. Они сильно зависят от контекста, разрешений и качества среды.
- Тор, прокси и удаленный сервер не отменяют следы, ограничения и детектирование. Это не магическая кнопка “стать невидимым”.

12. Финальный checklist для студента

- У меня есть базовый baseline: host + Kali VM или Debian VPS.
- Я использую только официальные образы и официальные страницы загрузки.
- Я обновил систему и поменял пароль/настроил доступ по ключам.
- У меня есть snapshot `clean-base` и понятная структура рабочих каталогов.
- Я умею запускать отдельный браузерный профиль под Burp.
- Я понимаю, где хранить API-ключи и почему их нельзя коммитить.
- Я поставил только минимально нужный набор инструментов, а не “всё подряд”.
- Я сохраняю результаты в файлы и веду заметки.
- Я понимаю, что lab-only команды выше не предназначены для случайного запуска по чужим сетям.
- Я понимаю, что профессиональный рост начинается после сборки среды — в регулярной практике, разборе ошибок и повторении.

13. Список официальных ссылок из воркшопа

- Stepik: «Профессия — Белый Хакер»: <https://stepik.org/course/169003/promo>
- CyberEd: «Пентестер»: <https://cyber-ed.ru/professiya-pentester/>
- Kali: Get Kali: <https://www.kali.org/get-kali/>
- Kali: import pre-made VirtualBox VM: <https://www.kali.org/docs/virtualization/import-premade-virtualbox/>
- Kali: official images and VM downloads: <https://www.kali.org/docs/introduction/download-official-kali-linux-images/>
- Debian: Securing Debian Manual: <https://www.debian.org/doc/manuals/securing-debian-manual/index.en.html>
- PortSwigger: Burp docs: <https://portswigger.net/burp/documentation/contents>
- PortSwigger: installing Burp CA certificate: <https://portswigger.net/burp/documentation/desktop/external-browser-config/certificate>
- pipx install docs: <https://pipx.pypa.io/latest/how-to/install-pipx/>
- uv install docs: <https://docs.astral.sh/uv/getting-started/installation/>
- Go install docs: <https://go.dev/doc/install>
- Shodan API docs: <https://developer.shodan.io/api>
- SecurityTrails API docs: <https://docs.securitytrails.com/docs/overview>
- VirusTotal API overview: <https://docs.virustotal.com/docs/api-overview>
- Nmap usage reference: <https://svn.nmap.org/nmap/docs/nmap.usage.txt>
- Nmap book: <https://nmap.org/book/toc>

