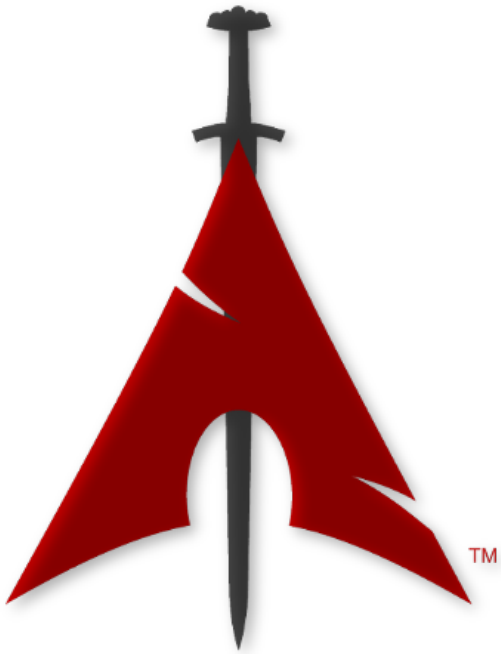

The BlackArch Linux Guide

https://www.blackarch.org/

Contents

Chapter 1

Einfuehrung

1.1 Uebersicht

Das BlackArch Linux Handbuch ist in verschiedene Teile aufgeteilt:

- Einfuehrung - Gibt einen breiten Ueberblick, eine Einfuehrung, und weitere hilfreiche Projektinformationen
- Nutzerhandbuch - Alles was ein typischer Nutzer wissen muss um BlackArch zu benutzen
- Entwicklerhandbuch - Wie kann man zu BlackArch beitragen und entwickeln
- Tool Guide - Tiefgehende Details zu Tools und Beispiele zur Benutzung (WIP)

1.2 Was ist BlackArch Linux?

BlackArch ist eine vollwertige Linux Distribution fuer Penetration Tester und Security Researcher. Es basiert auf ArchLinux und Nutzer koennen BlackArch Komponenten einzeln oder in Gruppen installieren.

Das Toolset wird mittel eines inoffiziellem Benutzer Repositories verteilt, so dass man BlackArch auf einer existierenden Arch Linux Installation installieren kann. Pakete koennen individuell oder ueber Kategorien installiert werden. Inoffizielles Nutzerrepository

Das konstant wachsende Repository beinhaltet aktuell ueber 1300 tools. Alle tools werden intensiv getestet bevor sie zur Codebasis hinzugefuegt werden, um die Qualitaet des Repositories zu gewaehrleisten.

1.3 Geschichte von BlackArch Linux

Coming soon...

1.4 Unterstuetzte Plattformen

Coming soon...

1.5 Mitmachen

Man kann ueber folgende Wege mit dem BlackArch Team in Kontakt treten: Website: https://www.blackarch.org/

Mail: team@blackarch.org

IRC: irc://irc.freenode.net/blackarch

Twitter: https://twitter.com/blackarchlinux

Github: https://github.com/Blackarch/

Chapter 2

Benutzerhandbuch

2.1 Installation

Der folgende Abschnitt zeigt, wie man das BlackArch Repository einrichtet und Pakete installiert. BlackArch unterstuetzt sowohl die Installation von Binaerpaketen als auch die Installation ueber selbstkompilierten Quellcode.

BlackArch ist kompatibel mit regulaeren Arch installationen. Es verhaelt sich wie ein inoffizelles Nutzerreporisotry. Wenn stattdessen ein ISO benoetigt wird, siehe den Abschnitt Live ISO.

2.1.1 Installation basierend auf einer vorhandenen ArchLinux Installation

Fuehre strap.sh als root aus und folge den Anweisungen.

Hier ein Beispiel.

```
curl -O https://blackarch.org/strap.sh
sha1sum strap.sh # should match: 86eb4efb68918dbfdd1e22862a48fda20a8145ff
sudo ./strap.sh
```

Jetzt lade eine frische Kopie der Master Paket Liste und synchronisiere die Pakete:

```
sudo pacman -Syyu
```

2.1.2 Paketinstallattion

Jetzt koennen Tools aus dem BlackArch Repository installiert werden.

1. Um alle verfuegbaren Tools aufzulisten:

   ```
   pacman -Sgg | grep blackarch | cut -d' ' -f2 | sort -u
   ```

2. Um alle Tools zu installieren:

   ```
   pacman -S blackarch
   ```

3. Um eine Toolkategorie zu installieren:

```
pacman -S blackarch-<category>
```

4. Um die BlackArch Kategorien zu sehen:

```
pacman -Sg | grep blackarch
```

2.1.3 Paketinstallation auf Quellcodebasis

Alternativ koennen BlackArch-Pakete auch aus Quellcode gebaut werden. Die PKGBUILDS koennen auf github gefunden werden. Um das gesamte Repository zu bauen, kann das Blackman tool genutzt werden.

- Als erste muss Blackman installiert werden. Wenn das BlackArch Reposity auf ihrer Maschine eingerichtet ist, kann Blackman installiert werden:

```
pacman -S blackman
```

- Blackman kann von Quellcode gebaut und installiert werden:

```
mkdir blackman
cd blackman
wget https://raw2.github.com/BlackArch/blackarch/master/packages/blackman/PKGBUILD
# Sicherstellen dass die PKGBUILD nicht boesartig veraendert worden sind.
makepkg -s
```

- Blackman kann auch aus dem AUR installiert werden:

```
<Verwendeter AUR Helfer> -S blackman
```

2.1.4 Grundlegende Verwendung von Blackman

Blackman ist sehr einfach zu nutzen, auch wenn sich die flags von dem unterscheiden, was man typischerweise von pacman erwarten wuerde. Die Grundlegende Benutzung wird im folgenden gezeigt.

- Herunterladen, kompilieren and installieren von Paketen:

```
sudo blackman -i package
```

- Herunterladen, kompilieren und installieren einer ganzen Kategorie:

```
sudo blackman -g group
```

- Herunterladen, kompilieren und installieren aller BlackArch Tools:

```
sudo blackman -a
```

- Auflistung aller BlackArch Kategorien:

```
blackman -l
```

- Auflistung der Tools einer Kategorie:

```
blackman -p category
```

2.1.5 Installing from live-, netinstall- ISO or ArchLinux

BlackArch Linux kann von unseren live- oder netinstall-ISOs intalliert werden.
Siehe https://www.blackarch.org/download.html#iso. Die folgenden Schritte sind noetig wenn die ISO gebootet ist.

- Installieren des blackarch-installer Pakets:

```
sudo pacman -S blackarch-installer
```

- Run

```
sudo blackarch-install
```

Chapter 3

Entwicklerhandbuch

3.1 Das Arch Build System und Repositories

PKGBUILD Dateien sind Build Skripte. Jedes beschreibt makepkg(1) wie ein Paket gebaut wird. PKGBUILD Dateien werden in Bash geschrieben.

Fuer weitere Informationen, lese (oder ueberfliege) folgende Seiten:

- Arch Wiki: Erzeuge Packages
- Arch Wiki: makepkg
- Arch Wiki: PKGBUILD
- Arch Wiki: Arch Packetierungs Standards

3.2 Blackarch PKGBUILD Standards

Der Einfachkeit halber sind unsere PKGBUILDs dem des AUR sehr aehnlich, die kleinen Unterschiede werden im weiteren Text beschrieben. Jedes Paket muss mindestens zu blackarch gehoeren, es wird aber auch viele beziehungen ueber mehrere pakete die zu mehreren Gruppen gehoeren geben.

3.2.1 Gruppen

Um es Nutzern zu ermoeglichen eine ganze Reihe von Paketen schnell und einfach zu installieren, wurden Pakete in Gruppen eingeteilt. Gruppen ermoeglichen es den benutzern mit einem einfachen "pacman -S <group name>" eine Menge von Paketen zu bekommen.

3.2.1.1 blackarch

Die blackarch gruppe ist die basis-Gruppe zu der alle Pakete gehoeren muessen. Das ermoeglicht es den Nutzern einfach alle Pakete zu installieren.

Was sollte hier drin sein: Alles.

3.2.1.2 blackarch-anti-forensic

Pakete die dazu benutzt werden, forensische Aktivitaeten zu umgehen. Das beinhaltet Verschluesselung, Steganographie und alles was es ermoeglicht Datei/Ordner Attribute zu manipulieren. Das alles beinhaltet Tools die allgemein veraenderungen an einem System durchfuehren mit dem Zweck, Information zu verstecken.

Beispiele: luks, TrueCrypt, Timestomp, dd, ropeadope, secure-delete

3.2.1.3 blackarch-automation

Pakete zur tool oder workflow Automatisierung.

Beispiele: blueranger, tiger, wiffy

3.2.1.4 blackarch-backdoor

Pakete zur Ausnutzung oder Oeffnung von backdoors auf bereits verwundbaren Systemen.

Beispiele: backdoor-factory, rrs, weevely

3.2.1.5 blackarch-binary

Pakete die auf irgendwelchen Binaerdateien arbeiten.

Beispiele: binwally, packerid

3.2.1.6 blackarch-bluetooth

Pakete die alles exploiten was mit dem Bluetooth Standard(802.15.1) zu tun hat.

Beispiele: ubertooth, tbear, redfang

3.2.1.7 blackarch-code-audit

Pakete die bestehenden Code analysieren um Sicherheitsluecken zu finden.

Beispiele: flawfinder, pscan

3.2.1.8 blackarch-cracker

Pakete die zum cracken von kryptographischen Funktionen, zum Beispiel Hashes.

Beispiele: hashcat, john, crunch

3.2.1.9 blackarch-crypto

Pakete die mit kryptographie arbeiten, mit der Ausnahme vom cracken.

Beispiele: ciphertest, xortool, sbd

3.2.1.10 blackarch-database

Pakete die Datenbank-Exploits auf jedem Level betreffen.

Beispiele: metacoretex, blindsql

3.2.1.11 blackarch-debugger

Pakete die es dem Nutzer erlauben in Echtzeit zu sehen, was ein bestimmtes Programm tut.

Beispiele: radare2, shellnoob

3.2.1.12 blackarch-decompiler

Pakete die versuchen kompilierte Programm in Quellcode zu konvertieren.

Beispiele: flasm, jd-gui

3.2.1.13 blackarch-defensive

Pakete die Versuchen den Nutzer vor Malware und Attacken anderer Nutzer zu schuetzen.

Beispiele: arpon, chkrootkit, sniffjoke

3.2.1.14 blackarch-disassembler

Aehnlich zu blackarch-decompiler> Hier gibt es vermutlich einige Programme die in beide Kategorien fallen, mit dem Unterschied das diese Pakete Assembler ausgeben statt den puren Quellcode.

Beispiele: inguma, radare2

3.2.1.15 blackarch-dos

Pakete die DoS (Denial of Service) Angriffe nutzen.

Beispiele: 42zip, nkiller2

3.2.1.16 blackarch-drone

Pakete die zur Verwaltung von echten Drohnen verwendet werden.

Beispiele: meshdeck, skyjack

3.2.1.17 blackarch-exploitation

Pakete die exploits anderer Programme oder Dienste nutzen.

Beispiele: armitage, metasploit, zarp

3.2.1.18 blackarch-fingerprint

Pakete die Fingerabdruecke biometrischer Systeme exploiten.

Beispiele: dns-map, p0f, httprint

3.2.1.19 blackarch-firmware

Pakete die Schwachstellen in Firmware ausnutzen.

Beispiele: Noch keine, asap hinzufuegen.

3.2.1.20 blackarch-forensic

Pakete die benutzt werden um Daten auf physischen Festplatten oder Speicher zu finden.

Beispiele: aesfix, nfex, wyd

3.2.1.21 blackarch-fuzzer

Pakete die die Fuzzy Testprinzipien nutzen, zum Beispiel zufaelligen Input "reinzuwerfen" und zu sehen was passiert.

Beispiele: msf, mdk3, wfuzz

3.2.1.22 blackarch-hardware

Pakete die alles verwalten oder ausnutzen was mit physischer Hardware zu tun hat.

Beispiele: arduino, smali

3.2.1.23 blackarch-honeypot

Pakete die als "honeypots" fungieren. Zum Beispiel Programme die sich als verwundbare Dienste ausgeben und Hacker in eine Falle locken sollen.

Beispiele: artillery, bluepot, wifi-honey

3.2.1.24 blackarch-keylogger

Pakete die Tastendruecke auf anderen Systemen aufnehmen und speichern.

Beispiele: None yet, amend asap.

3.2.1.25 blackarch-malware

Pakete die zu Malware zaehlen oder Malware erkennung.

Beispiele: malwaredetect, peepdf, yara

3.2.1.26 blackarch-misc

Pakete die nicht unbedingt in eine spezielle Kategorie passen.

Beispiele: oh-my-zsh-git, winexe, stompy

3.2.1.27 blackarch-mobile

Pakete die Mobile Plattformen manipulieren.

Beispiele: android-sdk-platform-tools, android-udev-rules

3.2.1.28 blackarch-networking

Pakete die IP Netzerke betreffen.

Beispiele: TODO

3.2.1.29 blackarch-nfc

Pakete die NFC (near-field communication) nutzen.

Beispiele: nfcutils

3.2.1.30 blackarch-packer

Pakete die Packer bedienen oder beinhalten.

Packer sind Programme die malware in anderen Executables einbetten.

Beispiele: packerid

3.2.1.31 blackarch-proxy

Pakete die als Proxy fungieren, also zum Beispiel Netzwerkverkehr durch einen anderen Knoten im Internet umleiten.

Beispiele: burpsuite, ratproxy, sslnuke

3.2.1.32 blackarch-recon

Pakete die aktiv verwundbare exploits suchen. Eine Obergruppe fuer aehnliche Pakete.

Beispiele: canri, dnsrecon, netmask

3.2.1.33 blackarch-reversing

Uebergruppe fuer jegliche decompiler, disassembler oder aehnliche Programme.

Beispiele: capstone, radare2, zerowine

3.2.1.34 blackarch-scanner

Pakete die ausgewaehlte Systeme auf Schwachstellen scannen.

Beispiele: scanssh, tiger, zmap

3.2.1.35 blackarch-sniffer

Pakete die mit dem analysieren von Netzwerkverkehr zu tun haben.

Beispiele: hexinject, pytactle, xspy

3.2.1.36 blackarch-social

Pakete die hauptsaechlich soziale netzwerke angreifen.

Beispiele: jigsaw, websploit

3.2.1.37 blackarch-spoof

Pakete die versuchen den Angreifer zu spoofen, sodass der Angreifer nicht als Angreifer fure das Opfer zu erkennen ist.

Beispiele: arpoison, lans, netcommander

3.2.1.38 blackarch-threat-model

Pakete die zum Reporten/Aufnehmen des Threat-Models in einem speziellen Szenario benutzt werden.

Beispiele: magictree

3.2.1.39 blackarch-tunnel

Pakete die dazu genutzt werden, Netzwerkverkehr zu einem gegebenen Netzwerk zu tunneln.

Beispiele: ctunnel, iodine, ptunnel

3.2.1.40 blackarch-unpacker

Pakete die dazu genutzt werden, vorgepackten Schadcode von einer executable auszupacken.

Beispiele: js-beautify

3.2.1.41 blackarch-voip

Pakete die auf VOIP Programmen und Protokollen arbeiten.

Beispiele: iaxflood, rtp-flood, teardown

3.2.1.42 blackarch-webapp

Pakete die auf internet-zugewandten Anwendungen arbeiten.

Beispiele: metoscan, whatweb, zaproxy

3.2.1.43 blackarch-windows

Diese Gruppe ist fuer native Windows Pakete die unter wine laufen.

Beispiele: 3proxy-win32, pwdump, winexe

3.2.1.44 blackarch-wireless

Pakete die auf drahtlosen Netzwerken arbeiten.

Beispiele: airpwn, mdk3, wiffy

3.3 Repository Struktur

Das primaere git repo fuer BlackArch befindet sich hier: https://github.com/BlackArch/blackarch.
Es gibt ausserdem verschiedene Sekundaere Repositories hier: https://github.com/BlackArch.

Innerhalb des Hauptrepos gibt es drei wichtige Verzeichnisse:

- docs - Dokumentation.
- packages - PKGBUILD Dateien.
- scripts - Nuetzliche kleine Skripte.

3.3.1 Scripts

Hier eine Referenz fuer Skripte im `scripts/` Verzeichnis:

- baaur - Coming soon: Wird Pakete in das AUR hochladen.
- babuild - Baut ein Paket.
- bachroot - Managen eines chroot zum testen.
- baclean - Raeumt alte .pkg.tar.xz Dateien aus dem Paket Repository.
- baconflict - Wird bald `scripts/conflicts` ersetzen.
- bad-files - Findet schlechte Dateien in gebauten Paketen.
- balock - Anlegen oder loesen des Repository locks.
- banotify - IRC benachrichtigen ueber Paket pushes.

- barelease - Veroeffentlicht Pakete in das Repository.
- baright - Gibt die BlackArch Copyright Informationen aus.
- basign - Signiert Packete.
- basign-key - Signiert einen Schluessel.
- blackman - Verhaelt sich aehnlich wie pacman, baut aber aus git. (Nicht zu verwechseln mit nrz's Blackman)
- check-groups - Ueberprueft groups.
- checkpkgs - Ueberprueft Pakete auf Fehler.
- conflicts - Sucht nach Dateikonflikten.
- dbmod - Modifiziert eine Paketdatenbank.
- depth-list - Erzeugt eine Liste sortiert nach Abhaengigkeitspfad.
- deptree - Erzeugt einen Abhaengigkeitsbaum, der nur blackarch Pakete enthaelt.
- get-blackarch-deps - Liefert eine List von blackarch Abhaengigkeiten fuer ein Paket.
- get-official - Liefert offizielle Pakete zum Release.
- list-loose-packages - Listet Pakete die weder in Gruppen noch Abhaengigkeiten anderer Pakete sind.
- list-needed - Liste fehlender Abhaengigkeiten.
- list-removed - Liste von Pakete die im Paketrepository sind aber nicht im git.
- list-tools - Liste der Tools.
- outdated - Sucht nach veralteten Paketen im Repository im Vergleich zum git Repository.
- pkgmod - Modifiziert ein Buildpaket.
- pkgrel - Zaehlt die pkgrel in einem Paket hoch.
- prep - Aufraeumen des PKGBUILD Datei-Styles und Fehlersuche.
- sitesync - Synchronisiert zwischein einer lokalen Kopie des Paketrepositories und der Remote.
- size-hunt - Sucht nach grossen Paketen.
- source-backup - Backup von package source Dateien.

3.4 Beitragen zum BlackArch Repository

Dieser Abschnitt zeigt, wie Beitraege im BlackArch Linux Projekt gemacht werden. wir akzeptieren Pull Requests jeglicher Groesse, von kleinen Tippfehler-Korrekturen bis zu neuen Paketen.
Fuer Hilfe, Vorschlage oder Fragen Kontaktiere uns.

Jeder ist willkommen. Alle Beitraege werden geschaetzt.

3.4.1 Benoetigte Tutorials

Bitte lies folgende Tutorials bevor du mitmachst:

- Arch Packaging Standards
- Paketerzeugung
- PKGBUILD
- Makepkg

3.4.2 Schritte zum Mitmachen

Um Aenderungen zum BlackArchLinux Projekt zu submitten, folge diesen Schritten: steps:

1. Fork das Repository von `https://github.com/BlackArch/blackarch`
2. Hacke die benoetigten Dateien (z.B. PKGBUILD, .patch files, usw).
3. Committe deine Aenderungen.
4. Pushe deine Aenderungen.
5. Bitte uns darum deine changes zu mergen, am liebsten durch einen Pull Request.

3.4.3 Beispiel

Das folgende Beispiel zeigt, wie ein neues Paket zum BlackArch Projekt submitted wird. Wir benutzen yaourt (pacaur kann auch benutzt werden) um eine bereits existierende PKGBUILD Datei fuer **nfsshell** aus dem AUR herunter zu laden und nach unseren Beduerfnissen anzupassen.

3.4.3.1 Fetch PKGBUILD

Die *PKGBUILD* Datei mit yaourt oder pacaur holen:

```
user@blackarchlinux $ yaourt -G nfsshell
==> Download nfsshell sources
x LICENSE
x PKGBUILD
x gcc.patch
user@blackarchlinux $ cd nfsshell/
```

3.4.3.2 Aufraeumen der PKGBUILD

Aufraeumen der *PKGBUILD* Datei und ein bisschen Zeit sparen:

```
user@blackarchlinux nfsshell $ ./blackarch/scripts/prep PKGBUILD
cleaning 'PKGBUILD'...
expanding tabs...
removing vim modeline...
removing id comment...
removing contributor and maintainer comments...
squeezing extra blank lines...
removing '|| return'...
removing leading blank line...
removing $pkgname...
removing trailing whitespace...
```

3.4.3.3 PKGBUILD anpassen

Anpassen der *PKGBUILD* Datei:

```
user@blackarchlinux nfsshell $ vi PKGBUILD
```

3.4.3.4 Das Paket bauen

Bau das Paket:

```
==> Making package: nfsshell 19980519-1 (Mon Dec  2 17:23:51 CET 2013)
==> Checking runtime dependencies...
==> Checking buildtime dependencies...
==> Retrieving sources...
-> Downloading nfsshell.tar.gz...
% Total    % Received % Xferd  Average Speed   Time    Time     Time
CurrentDload  Upload   Total   Spent    Left  Speed100 29213  100 29213    0
0  48150      0 --:--:-- --:--:-- --:--:-- 48206
-> Found gcc.patch
-> Found LICENSE
...
<lots of build process and compiler output here>
...
==> Leaving fakeroot environment.
==> Finished making: nfsshell 19980519-1 (Mon Dec  2 17:23:53 CET 2013)
```

3.4.3.5 Installieren und testen des Pakets

Installiere und teste das Paket:

```
user@blackarchlinux nfsshell $ pacman -U nfsshell-19980519-1-x86_64.pkg.tar.xz
user@blackarchlinux nfsshell $ nfsshell # test it
```

3.4.3.6 Adde, commite and pushe das Paket

Fuege das Paket hinzu, mach den Commit und Pushe.

```
user@blackarchlinux ~/blackarchlinux/packages $ mv ~/nfsshell .
user@blackarchlinux ~/blackarchlinux/packages $ git commit -am nfsshell && git push
```

3.4.3.7 Erzeuge einen Pull Request

Erzeuge einen Pull Request auf github.com

```
firefox https://github.com/<contributor>/blackarchlinux
```

3.4.3.8 Fuege eine upstream remote hinzu.

Es ist eine gute Idee wenn man upstream auf einem Fork arbeitet, den eigenen Fork zu pullen und das Haupt-BlackArch repository als eine Remote hinzuzufuegen.

```
user@blackarchlinux ~/blackarchlinux $ git remote -v
origin <the url of your fork> (fetch)
origin <the url of your fork> (push)
user@blackarchlinux ~/blackarchlinux $ git remote add upstream https://github.com/blackarch/blac
user@blackarchlinux ~/blackarchlinux $ git remote -v
origin <the url of your fork> (fetch)
origin <the url of your fork> (push)
upstream https://github.com/blackarch/blackarch (fetch)
upstream https://github.com/blackarch/blackarch (push)
```

Standardmaessig sollte git direkt auf origin pushen, aber stelle sicher das deine git konfiguration richtig konfiguriert ist. Das sollte kein Problem sein, solange du commit rechte hast, da du ohne diese nicht upstream pushen kannst.

Wenn du nicht committen kannst, koenntest du mehr erfolg mit *git@github.com:blackarch/blackarch.git* haben.

3.4.4 Requests

1. Fuege keine **Maintainer** oder **Contributor** Kommentare zu *PKGBUILD* Dateien hinzu. Fuege maintainer und contributor Namen zu der AUTHORS sektion im BlackArch guide hinzu.
2. Der Konsistenz willen, bitte folge dem generellen Stil anderer *PKGBUILD* Dateien im repo und nutze doppel-space Einrueckungen.

3.4.5 Generelle tips

namcap kann Pakete auf Fehler ueberpruefen.

Chapter 4

Tool Handbuch

Coming soon...

4.1 Coming Soon

Coming soon...