aursec - A blockchain approach to securing software packages

Lukas Krismer & Bennett Piater

November 3, 2016

Universität Innsbruck - QE - Christian Sillaber

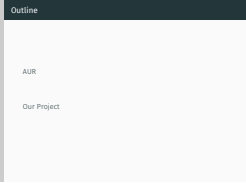
2016-11-03

aursec - A blockchain approach to securing software packages

Outline

AUR

Our Project



1 min L | Liebe Anwesende, Mein Name ist Lukas, und Bennett und ich werden euch heute unser Bachelorarbeitsthema näherbringen. Da oben steht so eine schöne große Überschrift ...

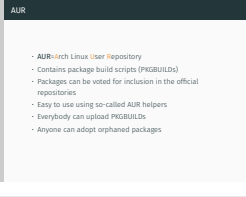
Aursec - A blockchain approach to securing software packages

Und wir möchten heute ein bisschen darauf eingehen wie man Softwarepakete, in unserem Fall von der AUR, sicherer machen kann. Am Anfang werden wir auf die AUR und ihre Probleme eingehen und später zeigen wir dann, wie wir Teilprobleme mithilfe einer Blockchain lösen wollen. Zum Schluss schauen wir dann noch unser Zeitmanagement und unsere Verantwortlichkeiten an.

AUR

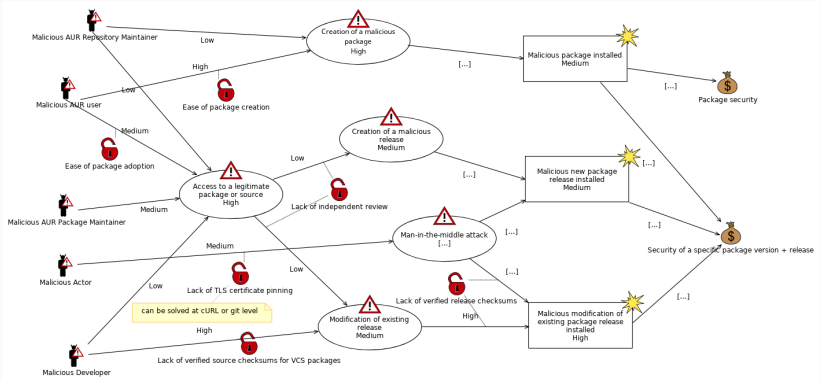
AUR

- **AUR**=Arch Linux User Repository
- Contains package build scripts (PKGBUILDs)
- Packages can be voted for inclusion in the official repositories
- Easy to use using so-called AUR helpers
- Everybody can upload PKGBUILDs
- Anyone can adopt orphaned packages



2min L | Kommen wir zur AUR, also der Arch Linux User Repo. Sie ist eine inoffizielle Repo von Arch Nutzern für Arch Nutzer. Sie beinhaltet Skripte (sog. PKGBUILDs) die für die Erstellung von Paketen benötigt werde . Weiters kann man für Pakete abstimmen um damit eine Aufnahme in die offizielle Community Repo zu fördern. Durch die AUR-helper ist die AUR sehr einfach zu bedienen. So sind yaourt, aurutils und co sehr ähnlich zu bedienen wie Packetmanager. Ein wichtiger Punkt der AUR ist weiters, dass jeder PKGBUILDs hochladen kann. Dies ist sowohl ein sehr großer Vorteil, da es dadurch für fast jedes Problem ein passendes Paket in der AUR gibt, als auch Nachteil, da die PKGBUILDs fast ausschließlich von Ihren Nutzern validiert werden. So können sich bei unbekannten

Threat Assessment



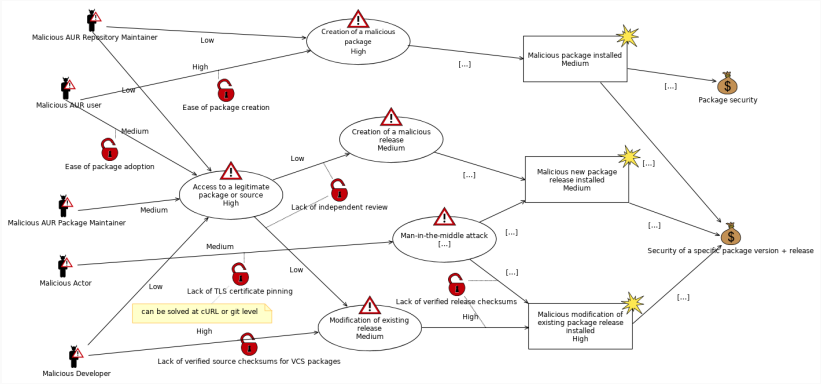
Threat Assessment

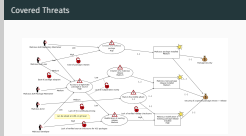
2 min B | Besonderes Augenmerk auf:

- Die grundlegenden Probleme der AUR sind praktisch unlösbar
- Zu viele haben Zugang zu Quellen und/oder Buildskripten
- Daher: Server-Seitige Signaturen würden nur MITM verhindern
- Bösartige Pakete, Releases oder Veränderungen sehr einfach

Our Project

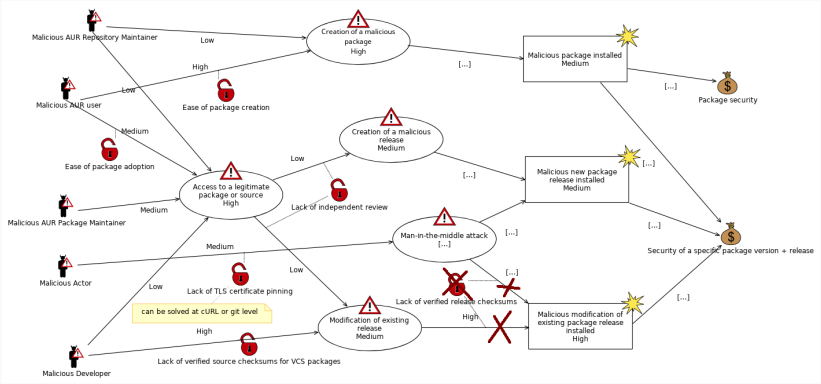
Covered Threats

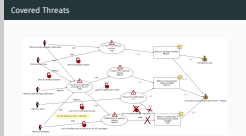




1 min L | Kommen wir zu unserem Projekt. Wenn wir uns wieder unsere Threatanalyse anschauen, sehen wir hier ”zeig” , dass die validierung der Version große Probleme beheben könnte. Und genau dies wollen wir tun. Mithilfe einer Blockchain sollte es uns Möglich sein dieses Problem zu beseitigen und somit auch die Urprobleme, wie Modifikationen eines Releases oder ein teil des man-in-the-middle Agriffs zu beheben

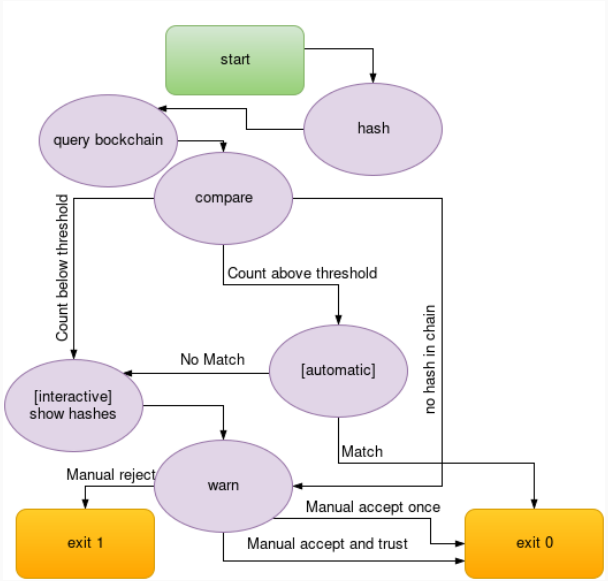
Covered Threats

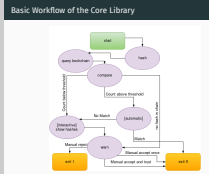




1 min L | Kommen wir zu unserem Projekt. Wenn wir uns wieder unsere Threatanalyse anschauen, sehen wir hier ”zeig” , dass die validierung der Version große Probleme beheben könnte. Und genau dies wollen wir tun. Mithilfe einer Blockchain sollte es uns Möglich sein dieses Problem zu beseitigen und somit auch die Urprobleme, wie Modifikationen eines Releases oder ein teil des man-in-the-middle Agriffs zu beheben

Basic Workflow of the Core Library





3 min L | 0) herunterladen des PKGBUILDs des gewollten Paketes 0) Wir generieren lokal einen Hash des PKGBUILDs des Paketes 2) Wir fragen für das Paket den Hash aus der Blockchain ab 3) Wir bekommen die Anzahl des Hashes + Hash für dieses Paket 4) 3 Möglichkeiten 1) kein Hash vorhanden Warnung Manuelles akzeptieren oder ablehnen 2) unterhalb des Treshold. Der Treshold ist eine Grenze bei der unser Programm automatisch den Hash als aussagekräftig und glaubhaft sieht. beide Hashes werden angezeigt Manuelles akzeptieren oder ablehnen 3) über dem Treshold 1) falsche Hash beide Hashes werden angezeigt Manuelles akzeptieren oder ablehnen 2) richtiger Hash (gewollte Fall) automatisches akzepieren –> automatisch in die Blockchain schreiben

2 Möglichkeiten vom manuellen akzeptieren, glaubwürdigkeit des

Components

- Program on a private Ethereum blockchain
- Shell library
- AUR package
- Integration in aurutils
- Threat analysis of the AUR and our software
- Web- and/or CLI-Interface for stats/events

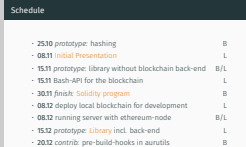
Components
- Program on a private Ethereum blockchain
- Shell library
- AUR package
- Integration in aurutils
- Threat analysis of the AUR and our software
- Web- and/or CLI-Interface for stats/events

2 min B

- Das eigentliche Programm zum Speichern der Hashes
- Unsere Library, die den Workflow automatisiert
- Ein Paket für die AUR
- Integration in einen der Besten AUR-Helper
  –> Im Zuge dessen allgemein nützliche Beiträge dazu
- Threat-analysen, um die Gefährdungsstufe und die Qualität unseres Beitrags einzuschätzen
- Ein Interface, mit dem die Aktivität der Blockchain überwacht werden kann

Schedule

- **25.10** *prototype:* hashing B
- **08.11** Initial Presentation L
- **15.11** *prototype:* library without blockchain back-end B/L
- **15.11** Bash-API for the blockchain L
- **30.11** *finish:* Solidity program B
- **08.12** deploy local blockchain for development L
- **08.12** running server with ethereum-node B/L
- **15.12** *prototype:* Library incl. back-end L
- **20.12** *contrib:* pre-build-hooks in aurutils B



2 min B

Wir haben eine sehr **detaillierte Planung** ausgearbeitet. Einerseits benötigen wir sie, um effizient **kooperieren** zu können und zügig voran zu kommen; Andererseits soll sie uns auch ein Maximaltempo vergeben, denn wir tendieren beide eher dazu, uns zu **überarbeiten**.

- Solidity-program auf Blockchain
- Library-Prototyp
- Beiträge zum AUR-Helper aurutils über Weihnachten

Schedule

- **10.01** *contrib:* TLS-public-key-pinning in aurutils B
- **10.01** configuration and trust-cutoff L
- **15.01** *test:* Integration in aurutils B
- **15.02** AUR package incl. private blockchain B
- **01.03** *finish:* libary and aurutils-Hook B
- **31.03** *finish:* Web- and/or CLI-Interface L
- **21.04** Draft paper for feedback
- **??.05** *finish:* Paper
- **??.05** Final presentation L

Schedule
- **10.01** contrib: TLS-public-key-pinning in aurutils B
- **10.01** configuration and trust-cutoff L
- **15.01** test: Integration in aurutils B
- **15.02** AUR package incl. private blockchain B
- **01.03** finish: libary and aurutils-Hook B
- **31.03** finish: Web- and/or CLI-Interface L
- **21.04** Draft paper for feedback
- **??.05** finish: Paper
- **??.05** Final presentation L

2 min B

- am 15.01 mit aurutils testbar
- AUR-Paket zur einfachen Verbreitung
- Programmierung endet am 31. März
- Meiste Schreibarbeit im April und besonders über Ostern
- Abgabe bequem for den Klausuren