Fakultät Informatik, Mathematik und Naturwissenschaften

Studiengang

Informatik, Master

Master-Thesis

zum Thema

Schutz vor Betrug in Klassen- und Gruppenarbeiten mit Schüler-Tablets in Schulen

Vorgelegt von	Herr B. Sc. Alexander Kern
Abgabe am	09.12.2014
Gutachter 1	Herr Prof. Dr. rer. nat. Klaus Hänßgen
Gutachter 2	Herr Prof. Dr. rer. nat. habil. Michael Frank

Inhaltsverzeichnis

Eidesstattliche Versicherung

Ich erkläre hiermit, dass ich diese Arbeit selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche einzeln kenntlich gemacht. Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht worden. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Leipzig, 7. Oktober 2014

1. Einblick

1.1. Einleitung

In den nächsten Jahrzehnten wird sich die Schule der Zukunft herausbilden. Kann man sich heute schon vorstellen wie diese aussehen wird, und wie viel von der eigenen Prognose wird, wenn es so weit ist, übrigbleiben? Der Bildungssektor wurde bisher zunehmend technisiert und dieser Trend kann sich fortsetzen, aber nicht immer ist das sinnvoll und nicht auf jede Art und Weise. Mit dieser Arbeit lässt sich ein Bild entwerfen zu dieser Fragestellung im Bezug zu Klassenarbeiten mit Tablets. Manche Technologien werden sich erst im Laufe der Zeit herausbilden und anwenden lassen. So kann sich auch eine Evolution von computergestützten Klassenarbeiten ergeben. Wie in jeder fortschreitenden Entwicklung wird es neue Chancen, aber auch neue Probleme und Nachteile geben, die sich weiterhin im Wandel befinden. Diese Arbeit berührt diese Angelegenheiten, beschränkt sich dabei jedoch auf die Themen Klassen- und Gruppenarbeiten und anderen computergestützten Leistungsüberprüfungen (Testate, Prüfungszulassungen, etc.). Insbesondere geht es darum, wie verhindert wird, dass Tablet-User betrügen (Spicken) in Klassen- und Gruppenarbeiten, die benotet werden können sollen, aber auch Randthemen, wie Geo-Lokalisierung, werden behandelt. Es wird im Rahmen dieser Arbeit davon ausgegangen, dass die Tablets im Besitz der jeweiligen Bildungseinrichtung sind.

Um Betrugsversuche zu unterbinden, wird darauf eingegangen, welche digitalen Möglichkeiten zu Betrügen ein Tablet-Benutzer hat, und wie das entweder unterbunden wird oder festgestellt werden kann.
Verschiedene Sicherheitsthemen werden abschnittweise thematisiert und wie deren Programmierung modelliert wird. Dabei wird die Frage beantwortet, was für vorhandene Lösungen es gibt, die relevant sind für die spezialisierte Lösung die diese Arbeit beschreibt.

Klassenarbeiten und Testate werden bisher in der Regel an Schulen mit Stift und Papier durchgeführt, mit der Anwesenheit des Lehrers.
Klassenarbeiten, die mit einem Computer bzw. mit einem Tablet durchgeführt werden, bieten den Vorteil dass der Lehrer in einem Teil der Fälle oder teilweise nicht selbst die Arbeiten überprüfen muss. Weiterhin können mit Computern verschiedene Aufgaben automatisiert werden, wie z.B. das Austeilen der Aufgabenstellungen an die Tablets der Schüler oder das Versenden des Web-Links zur Aufgabenstellung. Die Abgabe kann mittels Cloud und Filehooks automatisch erfolgen ohne Click auf einen Button. Mittels

Datamining, das sich statistischen Methoden bedient, können die Klausuren analysiert werden. Z.B.: Wie oft wurde welche Fragestellung falsch beantwortet, von welcher Art Schüler, den leistungsstarken oder den leistungsschwächeren Schülern? Welche Korrelationen (lineare Zusammenhänge ohne zwingende Bekanntheit der Kausalität) gibt es? Die Langzeit-Archivierung ist auch kein Problem mit entsprechenden Diensten oder Datenträgern.

Diese Arbeit behandelt nicht Anwendungssoftware oder Webseiten zum Durchführen von Klausuren, z.B. mit Formularfeldern, die ausgefüllt werden können und eine Leistungsüberprüfung durchgeführt werden kann. Es geht nicht um didaktische Fragestellungen und auch nicht um das Verbieten von Webseiten.

Es geht in der Arbeit vornehmlich darum, ob und inwiefern es realisierbar ist, allgemein Leistungs-Überprüfungen von Gruppen oder Einzelpersonen auf einem Computer bzw. Tablet durchzuführen. Dabei stellt sich die Frage des Verhältnisses der Sicherheitsmaßnahmen gegen Sicherheitslücken von digitalen Leistungs-Überprüfungen und den zusätzlichen Verwaltungs-Aufwand, den eine Aufsichtsperson damit hat.

Wenn die Fragen dazu geklärt sind, ist es sinnvoll sich mit der Gestaltung von z.B. Formularfeldern für Fragen und Antworten in einer Klausur zu befassen.

Gewöhnliche IT-Sicherheits-Lösungen, wie z.B. Antivirensoftware oder Firewalls, etc. dienen dem Schutz vor der Gefahr von Außen, d.h. z.B. vor Wirtschaftsspionage. Die Sicherheitsmaßnahmen in digitalen Leistungs-Überprüfungen bedürfen eines Schutzes von Innen, d.h. vor dem Benutzer des Tablets selbst.

1.2. Problemstellung

Diese Arbeit untersucht Tablet-Betriebssysteme dahingehend, wie es mit ihnen möglich ist, das Tablet auf Ausnutzung von Sicherheitslücken in digitalen Leistungs-Überprüfungen zu überwachen oder dieses nicht zu ermöglichen.

Dazu gibt es die Möglichkeit, einen so genannten Kioskmodus umzusetzen, der spezifiziert wird für den Bildungssektor. Ein Kioskmodus von einem Betriebssystem oder für ein Betriebssystem von einem Dritt-Hersteller ist im Allgemeinen zur Einschränkung des Systems da. Damit wird der Handlungsspielraum eines Benutzers begrenzt, je nach dem wie konfiguriert wird. Es können Rechte und Funktionalitäten des Betriebssystems eingeschränkt werden. Mit dem Kioskmodus kann versteckt werden, welches Betriebssystem im Hintergrund läuft. Z.B. basieren auch Geldautomaten und Computer in Bibliotheken oder Informationsterminals auf einem Kioskmodus.

Es gibt mehrere Varianten, wie ein Kioskmodus umgesetzt werden kann. Ein Kioskmodus kann angepasst sein an spezielle Nutzungsarten, für die ein anderer Kioskmodus nicht sinnvoll ist, durch das Design des Kioskmodus oder die Konfiguration.

Die Verwendung eines Kioskmodus schränkt Rechte und Funktionen insoweit ein, dass mit dem Tablet in Leistungsüberprüfungen nicht betrogen werden kann, bis eine Schwachstelle gefunden wurde.
Mit digitaler Überwachung von Tablets lässt sich feststellen, wann betrogen wird. Dann ist kein Kioskmodus nötig.
Das bedeutet, dass es diese beiden Möglichkeiten gibt, entweder Rechte und Funktionen einzuschränken oder vor Betrugsversuchen zu überwachen.

Beide Konzepte lassen sich kombinieren, was eine dritte Möglichkeit darstellt, die diese Arbeit nicht behandelt.

Praktisch basiert die Programmierung auf Betriebssystemfunktionen. Diese werden für die Arbeit recherchiert und festgehalten, wie sie für einen Kioskmodus oder eine Überwachung eingesetzt werden müssen.

Diese Arbeit behandelt dazu die Positionserkennung eines Gerätes auf der Erde, der Geo-Lokalisierung. Damit können Tablets eine vorher programmierte automatisierte Reaktion im Gerät ausführen, wenn sich der Tablet-Benutzer z.B. dem Klassenraum nähert.

Es wird sich auf Android 4.x und Windows 8.1, bzw. im Speziellen der Enterprise-Version, beschränkt.

Diese Arbeit beschäftigt sich mit dem Android von Google und nicht mit Modifikationen von Android z.B. von Handyherstellerfirmen und deshalb wird auf diese Spezialfälle nicht näher eingegangen.

1.3. Aufbau der Arbeit

In Kapitel 2 wird die Ausgangslage dargelegt. Es wird grundlegendes geklärt: Welchen Betrugsversuchen muss entgegnet werden, welche Betriebssysteme behandelt diese Arbeit und welche Funktionalitäten und definierten Einschränkungen werden gefordert. Da es sich um eine Netzwerksoftware handelt, werden die Anforderungen an das zugehörige Netzwerkprotokoll gestellt.

In Kapitel 3 wird schließlich das entworfene Konzept vorgestellt, mit der das Problem dieser Arbeit gelöst wird. Der Lösung wird ein Name gegeben: Anti-Cheat-Plus.
Dabei wird unterschieden wie die Problemstellung der Arbeit auf verschiedenem Wege

unter Windows und Android gelöst wird.

Kapitel 4 behandelt schließlich die Details des Konzeptes, bei denen weitere Fragen durch die Vertiefung auftauchen, die darin beantwortet werden. Die Beschreibung der einzelnen Inhalte von Kapitel 4 befindet sich am Anfang des 4. Kapitels.

Im darauf folgenden 5. Kapitel wird ein weiteres Mal die Sicherheit der Lösung von Anti-Cheat-Plus analysiert. Es geht darin nicht um Sicherheitsfragen von Details wie in Kapitel 4, sondern um die Hinterfragung des Gesamtkonzeptes. Dabei wird direkt im jeweiligen Abschnitt beschrieben, wie Problemen entgegnet werden muss.

Eine funktionierende Lösung nützt nichts, wenn sie zu langsam ist, wenn dadurch der Ablauf im Klassenraum beeinträchtigt wird. Deshalb wird in Kapitel 7 gemessen, wie schnell die Netzwerkkommunikation vonstatten geht.

Das letzte Kapitel 8 schließt die Arbeit ab, mit einer Zusammenfassung, in der die Ergebnisse der Arbeit zusammengetragen werden. Darauf folgt der Ausblick, in dem weiterführende Fragen und Sachverhalte diskutiert werden, die mit dieser Arbeit aufgekommen sind.

1.4. Kern-Ergebnisse und Aussagen dieser Arbeit

Die Problemstellung dieser Arbeit ist prinzipiell lösbar, es müssen jedoch lückenlose Sicherheitsmaßnahmen durchgeführt werden. Z.B. muss im ganzen Anti-Cheat-Plus-Programm auf dem Schülertablet abdeckend sichergestellt werden, dass Abstürze als solche erkannt werden und unterschieden werden können vom manuellen Beenden. Ebenfalls flächendeckend dürfen alle Funktionalitäten des Schülertablets beim Prozess der Leistungsüberprüfung mit dem Tablet keine Möglichkeiten des Ausnutzen von Sicherheitslücken zulassen. Deshalb besteht das Gesamtkonzept aus mehreren einzelnen Bestandteilen. Die Sicherheitsvorkehrungen von Anti-Cheat-Plus begrenzen den Handlungsspielraum des Tablet-Nutzers (=„innen”) (indirekt auch bei der Variante mit Beobachtung), anders als Sicherheitslösungen wie Intrusion-Detection-Systeme und Firewalls, die vor äußeren Bedrohungen schützen. TLS schützt gewöhnlich vor Bedrohungen von außerhalb des benutzten Gerätes, wird in dieser Arbeit jedoch letztendlich und indirekt zum Schutz vor der Verwendung verbotener Informationen, u.a. durch Identifikation des Tablet-Nutzers, eingesetzt. Die automatische Erkennung des Klassenraumes, um zeitgesteuert Einschränkungen zu aktivieren und zu deaktivieren, kann nicht in jedem Fall garantiert funktionieren. So eine automatische Erkennung eignet sich für andere technische Reaktionen, die weniger riskant sind, als Funktions- und Rechteeinschränkungen. Anti-Cheat-Plus benötigt nicht so viel Zeit für die Netzwerktransaktionen, dass die Klassenarbeit gravierend zeitlich eingeschränkt wird.

2. Vorbedingungen und Voraussetzungen zur Spezifikation

2.1. Betrugsversuchs-Arten

Diese Auflistung in Abschnitt 2.1 wurde niedergeschrieben, damit Lehrkräfte vorbereitet sind Betrugsversuche als solche zu erkennen und mit dieser Arbeit Vorkehrungen dagegen getroffen werden können.

Zunächst wird hier die Unterteilung getroffen in computergestützte Betrugsversuche, Betrugsversuche ohne Computer (bzw. Tablets) und der Kombination beider Betrugsversuchs-Arten. Wegen der Aufgabenstellung dieser Arbeit wird nicht näher auf Betrugsversuche ohne Computerverwendung eingegangen.

- computergestützte Betrugsmethoden ohne Einbeziehung der nicht-technischen Umgebung
  - Webdienste, andere Netzwerkdienste, Dateien, Dateinamen, beliebige Datenträger,
  - Ethernet / LAN, Bluetooth, (Sensoren), Wi-Fi / WLAN, Infrarot (obsolet), GPS
  - abtippen oder per Copy'n Paste übertragen
  - vorausgefüllte Eingabefelder in der Prüfungs-App
  - Datenträger mit Mitschülern austauschen
  - Google Glass tragen, Smart-Watch (Computer-Armband-Uhr),
  - Notizen in einem (programmierbaren) Taschenrechner
- Betrugsvarianten mit Kombinationen des Tablets und Möglichkeiten der Umgebung
  - Kameranutzung (Fotografieren)
  - beim Nachbarn abschauen
  - dem Nachbarn das Tablet sichtbar angewinkelt hinhalten
  - mit einem Spiegel (ggf. von der anderen Seite transparent, z.B. als Brille)
  - kleiner schlecht erkennbarer Ohrhörer (Digital- oder Analog-Technik)

- Spicker mit antiker oder anderer obsoleter Verschlüsselungs-Arten, so dass Spicken schwerer nachweisbar wird als ohne, z.B. Cäsar-Chiffre, Vigenère-Chiffre, One-Time-Pad, Matrix-Verschlüsselungsarten - mit Taschenrechner oder manuell entschlüsselbar
- Verwendung von Tinte, die nur durch speziell farbiges Licht lesbar ist, die das Tablet ausstrahlen kann

2.2. Wahl des Betriebssystems

Die Aufgabestellung bezieht sich auf Tablets. 2014 dominieren 3 Betriebssysteme den Tabletmarkt: Android ,iOS und Windows (RT, 8, 8.1, RT 8.1). Die Windowsversionen, die keine RT-Versionen sind, gibt es in weiteren Ausführungen: z.B. Pro, N, Pro mit Mediacenter, Enterprise, Pro N oder ohne weiteres Textkürzel hinter der Versionsnummer, sondern nur 8 oder 8.1.

Mit den Dateirechten und den Gruppenrichtlinien die zusätzlich mit der Fernwartung aus dem Active Directory von Windows Server 2012 domänenweit steuerbar sind ermöglicht Windows 7 Pro - 8.1 Enterprise und RT, RT 8.1 die Einrichtung eines Kioskmodus ohne dass Software von einem Dritthersteller benötigt wird. Beschrieben ist das für Windows 7 in [1]. In den Windowsversionen nach der 7 (Windows 7 hat eigentlich Version 6.1) ist die Vorgehensweise ähnlich. In Windows 8 / 8.1 ohne Enterprise im Namen sind Gruppenrichtlinien nicht konfigurierbar.

In [2] werden Gruppenrichtlinien im Detail beschrieben. Mit ihnen lassen sich Einstellungen auf dem Rechner erzwingen. Dadurch sind z.B. wegen erzwungenen Änderungen an den Energieoptionen Einsparungen der Stromkosten möglich. Konfigurationen sind möglich, die die Sicherheit verbessern, insofern, dass der Rechner nur für bestimmte Anwendungsgebiete benutzbar ist. Das erhöht z.B. die Produktivität von Schülern, u.a. weil sie nicht bewusst und unbewusst abgelenkt werden.

In Windows RT / RT 8.1 lassen sich nur Windows Store Apps installieren, die zudem in einer Sandbox laufen. Mit einem Jailbreak (eine Rechteerweiterung bzw. Entsperrung) lässt sich das umgehen, jedoch werden damit Sicherheitsvorkehrungen umgangen, die eigentlich das Gerät u.a. schützen. Es lassen sich keine Dienste mit Systemrechten und auch keine Treiber installieren. Deshalb müssen Produkte explizit auf die Windows-RT-Kompatiblität hinweisen, denn die Treiber liefert nur Microsoft zum Endkunden. Es gibt neben Windows RT / RT 8.1 für ARM-Prozessoren noch Windows Embedded Compact (d.h. Windows Embedded für ARM), das auch Touchscreens unterstützt. Dieses beschränkt sich nicht auf die Installation von Apps, die nur in der Sandbox laufen können. Dafür ist die Programmierung eines Kioskmoduses machbar. Jedoch ist dies wie bereits geschrieben nicht nötig, weil Windows die Boardmittel für die Konfigurierung ei-

nes solchen schon mitliefert. In allen Windowsversionen ab 8 ist zudem ein Kioskmodus in der Benutzeraccounteinstellung einstellbar, jedoch nicht mit dem Funktionsumfang, den die Gruppenrichtlinien bieten. Windows Embedded Compact ist eine Weiterführung von Windows CE.

2.3. Anforderungen

Das in diesen Abschnitten 2.3 bis 2.4 Beschriebene gilt sowohl für Windows als auch für Android, sofern nicht explizit abgegrenzt wird.

Die Anwender sind der Lehrer und die Schüler einer Klasse im Klassenraum. Das zu spezifizierende (letztendlich „spezifizierte") Protokoll, des Server- und Client-Programms gegen unerlaubte Informationsbeschaffung, ist dazu da um in dem Netzwerk einer Schulklasse Nachrichten zu verarbeiten. Der Lehrer muss Befehle veranlassen. Die Tablets der Tablet-User müssen diese empfangen können. Der Lehrer muss begrenzten Zugriff auf die Tablets der Schüler, von der Ferne, von seinem Rechner aus, haben. Der Lehrer kann entweder den so genannten Kioskmodus initialisieren, d.h. normalen Benutzer-Zugang sperren, und Zugang aus dem Kioskmodus in normale Benutzeraccounts wieder erlauben oder er aktiviert / deaktiviert die Überwachung auf Betrugsversuche. Diese Überwachung darf nur das nötigste gegen Betrugsversuche beobachten und auch nur aktiviert sein, wenn der Tablet-Nutzer geprüft wird.

Der Tablet-Benutzer muss die entweder erlaubten (bei Beobachtung durch den Lehrer) oder einzig startbaren Programme (in einem Kioskmodus) sehen können. Ggf. wird ihm in einem von dem Lehrer begrenzten Maß das Vornehmen von Einstellungen erlaubt.

Es müssen verschiedene Profile für den Kioskmodus oder den Beobachtungsmodus hergestellt werden, z.B. für Klassenarbeiten, Gruppenarbeiten, Programmierungs-Leistungstest, Testate. Ein Profil setzt sich aus mehreren Einstellungen zusammen. Weitere Profile sind anlegbar. Diese Teil-Thematik gehört nicht zum Kern dieser Arbeit; es wird damit nichts neues hervorgebracht, so dass diese Arbeit Profile nicht weiter vertieft.

Es gibt mehrere Varianten, den Netzwerk-Beobachtungsmodus und fernwartbaren Kioskmodus, der hier beschrieben wurde zu entwerfen. Es ist möglich alles zur Überwachung manuell durch den Lehrer steuern zu lassen und teilweise oder vollständige Automatisierung umzusetzen. Zunächst wird auf die manuelle Umsetzung eingegangen und dann darauf aufbauend auf Möglichkeiten der Automatisierung.

2.3.1. Anforderungen an die Manuelle Variante

Der Lehrer muss alle Prozesse und Dienste der Tablet-Benutzer über den Weg des lokalen (Funk-)Netzwerkes in Erfahrung bringen können. Wenn es Abweichungen vom Normalzustand bei einem Tablet-Nutzer gibt und Applikationen gestartet sind, die nicht vorgesehen bzw. unbekannt sind, muss der Lehrer mit einem Click auf einen Hyperlink zu einer allgemeinen oder spezialisierten Suchmaschine samt übergebenen Suchbegriff nach dieser Applikation im Internet suchen können und sich so ein Urteil bilden. Gibt es Unregelmäßigkeiten auf den Tablets mehrerer Tablet-Benutzer, das heißt der Lehrer sieht in seinem Beobachtungs-Programm mehrere Programme / Dienste auf mehreren Tablets, die da nicht sein sollen, dann muss es Möglichkeiten geben, wie er sich zunächst einen besseren Überblick verschaffen kann, als mit einer ungeordneten Anzeige. Es können Mengenoperationen wie Durchschnitt und Vereinigung und Sortierungen von Prozessnamen zugehörig zu Tablets durchgeführt werden, um Auffälligkeiten zu entdecken, welche Prozesse ein Tablet-User hat, die die anderen Tablet-User nicht haben. Das bedeutet, dass der Lehrer (die Aufsichtsperson) sich seine eigene Art von Ordnung einrichten kann, wie er es selbst für optimal findet den Überblick zu bewahren. Ggf. wird das nicht nötig sein, wenn der Tablet-Nutzer sich erst in einen Benutzeraccount einloggen muss, der von vornherein mit einem Kioskmodus geschützt ist, weil dann nur definierte Programme startbar sind und keine anderen. Jedoch ist es immer denkbar, dass der Schüler sein Tablet gehackt / manipuliert hat. Ggf. kann der Lehrer die Informationen speichern und laden welche Prozesse und Dienste der Tablet-Nutzer beim letzten Mal gestartet worden sind, was ggf. auch automatisch geschehen kann.

Der Lehrer muss unter Windows des Weiteren Benutzeraccounts global für alle Rechner deaktivieren und aktivieren können und ein Abmelden von Benutzern initieren können. Der Lehrer muss alles auf die Standardeinstellungen zurücksetzen können und feststellen können, ob ein Tablet von einem Tablet-Nutzer heimlich gerootet wurde oder ein Jailbreak (eine Rechteerweiterung bzw. Entsperrung) installiert wurde.

Letztendlich muss der Tablet-Benutzer die Lösungen seiner Aufgaben abgeben können. Dies kann über das Netzwerk erfolgen, webbasiertes Aufgabenlösen ist dazu eine Alternative.

2.3.2. Anforderungen an die automatische Variante

Automatische Abläufe nehmen der Aufsichtsperson Aufgaben und Aufwand ab. Dadurch benötigt die Aufsichtsperson kein IT-Expertenwissen mehr, wie beschrieben in Unter-Abschnitt 2.3.1.

Damit nicht mehr auffällige Prozesse von der Aufsichtsperson manuell untersucht werden müssen, muss eine Liste von allen Programmen die es für die betreffenden Betriebssys-

teme gibt geführt und ständig aktuell gehalten werden oder alle Programme, die auf den Tablets installiert sind, werden immer von einer Organisation in regelmäßigen Abständen dokumentiert und bewertet. Dabei müssen auch Programme, die ohne Installer auf das Gerät gelangt sind, einbezogen werden, denn zum Überspielen dieser auf das Tablet, werden keine Administratorrechte benötigt. Dies betrifft alle ausführbaren Dateien.

Damit die Liste immer aktuell ist, bietet es sich an, zum Abrufen dieser, einen (Internet-)Netzwerkdienst bereitzustellen.
Solche Listen existieren bereits in Firewall- und Antiviren-Software. Jedoch wird dabei in „gefährliche" und „ungefährliche" Applikationen eingeteilt. Dabei geht es um den Schutz vor Malware und Hackern.
Eine Liste von Programmen für das Anwendungsgebiet dieser Arbeit, muss jedoch unterscheiden in Programme, die zugelassen werden dürfen und in welche die nicht zugelassen werden dürfen. Ggf. kann es Grauzonen dazwischen geben, denn es geht nicht nur um gewöhnliche Klassenarbeiten, sondern z.B. um Programmiertestate im Informatikunterricht an Schulen. Das bedeutet, dass nicht nur Abstufungen zwischen dem Zulassen und Nicht-Zulassen denkbar sind, sondern auch um Zulassen und Nichtzulassen nach Art der Leistungsüberprüfung (Klassenarbeit , Gruppenarbeit, etc.).

Um die Aufsichtsperson zu entlasten, ist es zielführend keine Abstufungen zwischen dem Zulassen und nicht Zulassen zu implementieren. Jedoch ist situationsbezogenes White- (inklusiv-)oder Blacklisting nach Art der Leistungsüberprüfung nützlich, damit der Einsatzzweck nicht nur einer ist, z.B. nicht nur für Klassenarbeiten.

Um stattgefundenes Jailbreaking und Rooting und Manipulationen am System automatisiert feststellen zu können, gibt es Methoden der forensischen Analyse (Einbruchserkennung). Das zu behandeln sprengt jedoch den Rahmen dieser Arbeit.

Die hier in Unterabschnitt 2.3.2 beschriebene Automatisierung basiert prinzipiell auf dem Aktuell-Halten von Listen mehrmals vor den Leistungsüberprüfungen und dem Abruf dieser Listen vor der Leistungsüberprüfung. Für diese Vorgänge wird keine Spezialisierung benötigt, weswegen dies in dieser Arbeit nicht weiter behandelt wird. Wie man jedoch manuell eine einzelne App auf Sicherheitslücken für Leistungskontrollen für diese Listen untersucht, erfährt man indirekt durch das Lesen dieser ganzen Arbeit.

2.4. Anforderungen des Netzwerk-Protokolls

Aus den unter 2.3 beschriebenen Sichten ergeben sich die Anforderungen an das zugehörige Anwendungs-Netzwerk-Protokoll.

- Prozesse und Dienste übermitteln: Name, Streufunktion (z.B. MD5-Hash), ggf. Dateigröße, (Dienste: ob aktiviert oder deaktiviert und ob sie immer manuell gestartet

werden müssen oder sich automatisch starten beim Startvorgang)

- Benutzeraccountinformationen senden und ändern, darunter ob aktiv / inaktiv; Befehl zum Ändern in deaktiviert / aktiviert von Accounts
- Broadcast-Ping und Unicast-Ping von allen Tablets / PCs des Klassenraums zu allen anderen Tablets / PCs in einem Klassenraum (der in Abständen von Minuten oder Sekunden stattfindet, um Auffälligkeiten zu registrieren) (Diese Art Ping funktioniert nur mit dem Überwachungsprogramm und ist nicht zu verwechseln mit dem Shellbefehl Ping.)
- ggf. Sperrungsdetails senden und empfangen (Kioskmodus) , einstellen / auslesen, z.B. Apps, Shortcuts, Widgets, Dateisystemrechte
- Übertragung der aktuellen Zeit und ggf. Zeitmessungen zwischen Befehlsaufrufen des Protokolls, deren Ausführung auf dem Client oder Server und der Bestätigung über Erfolg oder Misserfolg
- Übertragung von Materialien (Aufgabenblatt / Lösungen) erlauben / blocken

Die Tablets der Schüler müssen einen Dienst als Server installiert haben, den der Lehrer ansteuern kann mit seinem Fernsteuer-Client.

3. Konzept der Spezifikation von Anti-Cheat-Plus

Name:
Das Server- und Client-Programm, das in dieser Arbeit spezifiziert wird, für Android (inklusiv-)oder Windows wird ab sofort Anti-Cheat-Plus genannt.

Entweder Beobachtung oder Einschränkung:
In Kapitel 4 wird u.a. thematisiert welche Rechte und Funktionen eingeschränkt werden müssen (Abschnitt 4.5) oder im anderen Fall, was alles überwacht werden muss, damit Tablet-User fair bleiben beim Testat (Abschnitt 4.7).

Verschlüsselte und identifizierende Netzwerk-Verbindung mit Sicherstellung der Integrität:
Für den Netzwerk-Beobachtungs- und den fernbedienbaren Kioskmodus werden TLS-Sockets verwendet.

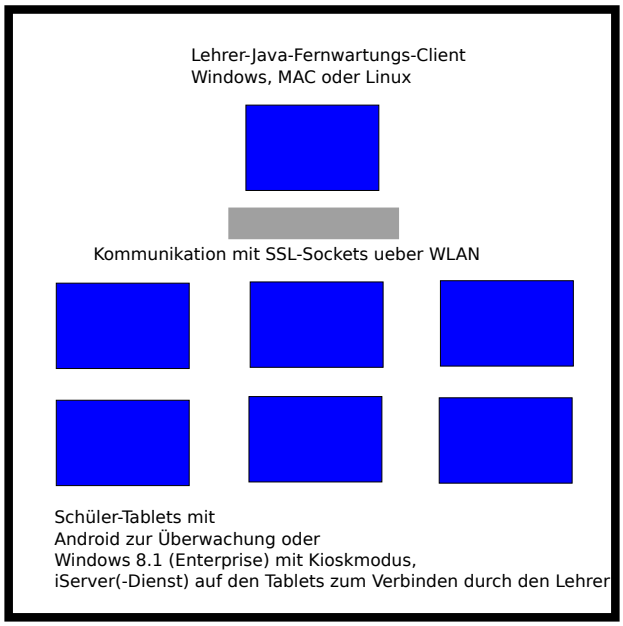
TLS-Sockets ermöglichen die Authentifikation sowohl des Servers- als auch des Clients, womit sichergestellt werden kann, dass nur der berechtigte Lehrer Fernwartungszugriff hat und dass der Lehrer die Tablets zugehörig zu den jeweiligen Schülern identifizieren kann. In Unterabschnitt 4.1.1 und Abschnitt 4.2 wird diese Sicherheitsproblematik vertieft.

Die Tablet-User können die TLS-Verbindungen zu anderen Tablet-Usern im Klassenraum wegen der Verschlüsselung nicht lesen. Außerdem bieten TLS-Sockets Integritäts-Schutz vor Manipulation.

Design des Netzwerkprotokolls:
Es wird ein Netzwerkprotokoll spezifiziert, das zur Fernwartung der Tablets unter Windows dient. Auf diesem Protokoll kann aufgebaut werden, wenn es um die Überwachung der Tablet-Benutzer geht. Die nötigen Betriebssystemfunktionen dazu sind recherchiert, in Kapitel 4 beschrieben, die Protokollgrundstruktur ist spezifiziert in Unter-Abschnitt 4.3.1, so dass eine Erweiterung des Protokolls mit den Überwachungsfunktionen für Android trivial ist.

Automatische, regelmäßige Detektion von vorhandenen Programmen zur Vorsor-

Abbildung 3.1.: Netzwerk-Schema im Klassenraum



ge für die Betrugsversuch-Erkennung und Festlegung erlaubter Programme:
Weil Prozessnamen leicht geändert werden können und damit nicht eindeutig sind, muss von Apps in Android und von Programmen in Windows, frühestens nach jedem Update des Betriebssystems und von Apps, von neuem der Hashwert der ausführbaren Datei (z.B. *.exe) im Schüler-Tablet berechnet, und dem Lehrer gesendet werden. Wenn das manuell Minuten vor einer Klausur geschieht und nicht regelmäßig automatisch, vergeht unter Umständen so viel Zeit, dass es nicht mehr zur Niederschrift der Tablet-Nutzer kommen kann und wird.

Kamera und Infrarotsensor als mögliche verbotene Informationsquelle:
Anstelle die Kamera zu nutzen kann man auch gleich einen Zettel verwenden und außerdem wird das Nutzen der Kamera-App dem Lehrercomputer als Betrugsversuch gemeldet. Deshalb muss die Kamera nicht deaktiviert werden. Mit Infrarot verhält es sich auch so. Die Überwachungsfunktion des Tablets unterbindet Apps zum Leuchten in speziellen Farben für das Spicken mit Spezialtinte.

Erlaubte Apps mit nicht erlaubten Bestandteilen:
Wenn der Browser erlaubt wird, dann kann es sinnvoll sein, dass Webseiten verboten und erlaubt werden, aber diese Arbeit behandelt dieses Thema nicht.

Betrugsversuchs-Vorbereitungen durch Manipulation von Außerhalb vom Betriebssystem:
Unterabschnitt 4.1.2 geht auf ein Hardwaresicherheitsproblem und das BIOS-Setup-Problem ein.

Unerlaubter Datenträgeraustausch durch den Tablet-Nutzer:
Gegen den Datenträgeraustausch hilft es ohne Datenträgern zu arbeiten oder der Lehrer überwacht ständig den Raum, um so etwas zu bemerken. Es gibt Geräte, die man erst ausschalten muss, damit die Micro-SD-Karte ausgetauscht werden kann.

Ggf. nicht nachweisbare Betrugsversuche:
Die Informationsbeschaffung über manuell getätigte Verschlüsselungsverfahren, zum Verschleiern der Tat, dauert für eine Klassenarbeit zu lange, so dass die Zeit dafür eher zu schlechteren Resultaten des Tablet-Nutzers führt, als ohne den Einsatz dieser.

Wahl der Programmiersprachen:
Für die Windows-Tablets kommt Visual Studio mit C# zur Programmierung des Servers infrage und für die Android-Tablets Java. Auf dem PC des Lehrers arbeitet ein Java-Fernwartungs-Client (Anti-Cheat-Plus). TLS-Sockets sind zwischen Java und C# kompatibel. Java-Programme sind lauffähig unter Windows, Solaris, Linux und OS X mit der Java-Laufzeitumgebung. Auf Android laufen nicht die Java-Anwendungen für PCs, weil Handys und Tablets ein anderes Bedienungskonzept haben und weil auf einem PC die Java-Virtual-Machine von Oracle lauffähig ist und in Android stattdessen Dalvik von Google.

Grundlegendes Problem:
Insgesamt darf kein Sicherheitsproblem außer Acht gelassen werden, denn das bedeutet ansonsten, dass die ganze Sicherheitslösung nutzlos ist. Es wird in der Arbeit untersucht, welches Sicherheitsproblem eine Sicherheitslücke darstellt.

Definition „Sicherheitslücke”:
Wenn in dieser Arbeit der Begriff Sicherheitslücke verwendet wird, dann ist damit in der Regel gemeint, dass es eine Möglichkeit gibt, dass ein Prüfling verbotene Informationsquellen über das Tabelt in einer Leistungsüberprüfung mit dem Tablet verwenden kann. Ein Ziel dieser Arbeit ist es alle Sicherheitslücken aufzudecken und zu behandeln.

3.1. Windows - Kiosk-Modus gegen Betrugsversuche

Modelle für die Sicherheit, bei einem Kioskmodus und deren Bedeutung für die Spezifikation:
Zur Beschränkung von Funktionalitäten und Begrenzung von Benutzerrechten im Rahmen eines Kioskmodus wird das Zwiebelschalenmodell mit Berücksichtigung von Hintertüren in der IT-Sicherheit als Schema herangezogen. Das Zwiebelschalenmodell besagt, dass wenn eine Schutzschicht (z.B. ein TLS basiertes VPN) überwunden wurde, es weitere Schichten geben kann, die noch zu umgehen / zu durchbrechen sind für einen

Angreifer. Das Prinzip der Hintertür besagt, dass eine oder mehrere Schichten mit einem Mal überwunden werden können. Das bedeutet, dass wenn es prinzipiell eine Hintertür gibt, alle Schichten die damit überwunden werden können nicht benötigt werden im Design des Sicherheitskonzepts oder es werden keine Hintertüren zugelassen. Im Abschnitt 4.1.1 und 5.2 wird dieser Zusammenhang anhand der Netzwerkverbindung von Schüler-Tablet und Lehrercomputer beschrieben.

Realisierung des Kioskmodus:
Wie ein Kioskmodus unter Windows realisiert wird, steht bereits in verlinkten Quellen für Windows 7 siehe Unterabschnitt 4.5.6. Deshalb ergänzt diese Arbeit was in Windows 8 anderes und weiteres zu tun ist.

3.2. Android - Tabletüberwachung zum Erkennen von unerlaubter Informationsbeschaffung

Kernbestandteil der Überwachung (Apps):
In Abschnitt 4.7 wird beschrieben, was überwacht werden muss, damit Klausur-Betrugsversuche aufgedeckt werden. Des Weiteren wird darin eine Methode beschrieben, mit der Tablet-Nutzer daran gehindert werden, nicht unbeabsichtigt Betrugsversuche zu begehen, für den Fall wenn sie nicht erlaubte Applikationen starten. Das wird mit einem dafür entwickelten App-Launcher realisiert, der den Umfang der startbaren Apps reduziert.

Grundsätzliches Problem bei Überwachung gegen das Ausnutzen von Sicherheitslücken (die geforderte Lückenlosigkeit):
Auch wenn unter Android kein Kioskmodus mit dieser Arbeit vorgesehen ist, müssen Möglichkeiten unter Android eingeschränkt werden bzw. automatisch analysiert werden, dahingehend ob Programmbestandteile zur verbotenen Informationsbeschaffung verwendet werden können. In Abschnitt 4.5.1 und 4.5.2 werden Methoden beschrieben mit denen Tablet-Nutzer unerlaubt Informationen beschaffen können, und was dagegen unternommen werden kann.

Bestandteile dieses grundsätzlichen Problems (Zwischenablage und Autovervollständigung):
Einer Überwachung der Zwischenablagen aller Tablets der Schüler im Klassenraum durch den Lehrer, wenn Klassenarbeiten geschrieben werden, kann man von ihm nicht erwarten. Eine Technologie, um alle gespeicherten Autovervollständigungen von allen Formularen einer App zu entnehmen ist aufwändiger, als Apps darin zu untersuchen ob Formulareingaben für Autovervollständigungen gespeichert werden oder gespeichert sind. Deshalb dürfen Apps nicht zugelassen werden, die eine Autovervollständigungsfunktio-

nalität enthalten, siehe Abschnitt 4.5.2. Wenn bekannt ist, wo die Daten für die Autovervollständigung für die Felder auf dem Gerät liegen, dann gibt es noch die Option dieses rechtzeitig zu löschen.

Unterstützung des Tablet-Nutzers vor unbeabsichtigten Betrugsversuchen wegen seiner Gewohnheiten:
Abschnitt 4.6.1, 4.6.2 und 4.6.4 behandeln das Unterbinden von Tasteneingaben, was den Tablet-Nutzer darin unterstützen soll, nicht unbeabsichtigt Apps zu starten, die der Lehrer verboten hat, z.B. wenn Gruppenarbeiten durchgeführt werden.

Abschnitt 4.6.5 dient letztendlich dem Ziel, dass der Tablet-Nutzer nicht mit wenigen Fingerbewegungen die Einstellungen erreicht, was dem Lehrer als Betrugsversuch gemeldet wird.

Zusätzliche, nicht notwendige Blockade vor Betrugsversuchen:
Abschnitt 4.5.4 und 4.5.9 behandelt das Deaktivieren der Kamera und den Sensoren. Diese müssen in Android nicht deaktiviert werden, jedoch kann deren Deaktivierung unter Umständen hilfreich sein als weitere Sicherheitsschicht. Sofern das System grundsätzlich Lücken aufweist, und dies nicht von Anfang an bemerkt wird, sind solche weiteren Sicherheitsschichten sinnvoll.

4. Kern-Elemente und Konzepte der Programmierung

Zunächst wird basierend auf dem Konzept der Spezifikation aufbauend geklärt, was der Kioskmodus beschränkt und dies nachhaltig in der Wirkung bestehen bleibt, sofern vom Tablet-Benutzer Gegenmaßnahmen getroffen werden. Dann wird spezifiziert, inwieweit Überwachungen durchgeführt werden müssen und darauf folgend wird die Frage nach Authentifizierung angeschnitten.

Abschnitt 4.2 kombiniert die vorhandenen Quellen über TLS miteinander und setzt sie in den Bezug zum Status von Sicherheitsfragen zur heutigen Zeit und auf die Besonderheiten beim Einbau von TLS mit Java wird eingegangen. Im nächsten Abschnitt wird das Netzwerkprotokoll von Anti-Cheat-Plus spezifiziert.

Schließlich wird aufgrund der Nützlichkeit darauf verwiesen, welche vorhanden Quelltexte verwendbar sind, die zunächst scheinbar nichts mit dieser Arbeit zu tun haben.
Abschnitt 4.5 und 4.6 behandelt schließlich den ganzen Einbau von Funktions- und Rechteeinschränkungen mit Spezifikations-Aspekten für Android und Windows. Darin steht, warum auch in Android Einschränkungen sinnvoll sind.

Im nächsten Abschnitt geht es um das was unter Android anders gelöst wird, den Beobachtungsmodus statt eines Kioskmodus.

Am Schluss dieses Kapitels wird beschrieben, wie durch Betreten und Verlassen des Klassenraums mittels Geo-Lokalisation und Entfernungsbestimmung ein automatisches Verhalten der Tablets in Gang gesetzt werden kann.

4.1. Notwendiger und möglicher Rechte- und Funktions-Entzug (Windows)

Gesperrt werden muss:

- Das Starten von einem definierten Teil aller Programme
- sichtbare und erreichbare Bestandteile der Festplatte, externer Datenträger (z.B. SD-Karte) und Netzwerkdatenträger - Das bedeutet in der Praxis, dass Laufwerke

ausgeblendet und versteckt werden und ein Ordner aus diesem Laufwerk zu einem Laufwerk mit Laufwerksbuchstaben definiert wird. Somit ist alles außer dieser Ordner durch den Benutzer einsehbar. Ggf. können mehrere Ordner dafür eingesetzt werden.

- kompletter Bluetooth-Sender und -Empfänger, ggf. Sensoren, WLAN-/LAN/WAN-Adressbereiche, also Ports und IPs, mit Whitelisting oder Blacklisting (d.h. Verbieten oder Erlauben von Teilen), Verbot von USB-Anschlüssen generell und Firewire und Serial- / Parallelport (Einige Schnittstellen müssen ggf. nicht gesperrt werden, wenn keine Software erlaubt ist, die davon Gebrauch nimmt.)
- URLs von Webseiten zu sperren wird in dieser Arbeit nicht thematisiert
- Antiviren- und Antispywareprogramme werden nicht benötigt, da Viren nicht starten können, weil nur definierte Programme im Kioskmodus erlaubt werden. Wenn Programme einen Virus in der Exe-Datei angehängt haben, dann werden diese auch nicht gestartet, da es einen Hashwertvergleich gibt, der überprüft, ob es das richtige Programm ist. Antiviren- und Antispywareprogramme müssen verboten werden oder am Starten gehindert werden, wenn mit diesen unter Umständen Einblick in das Dateisystem möglich sein kann. Das bedeutet, dass ein Benutzer über den Umweg des Antivirenprogramms Einblick in Bereiche des Dateisystems haben kann, die er nicht haben soll. Gibt es diese Gefahr nicht, dann können sie erlaubt werden.
- Ggf. Windows-Dienste von Drittanbietern, da diese ein zunächst unkalkulierbares Risiko für ein Einfallstor darstellen, sofern sie nicht analysiert wurden.
- Sofern möglich und nötig Sperren der Interprozesskommunikation (Socket, Named-Pipes, anonyme Pipes / Unnamed-Pipes, Shared Memory)
- Webcam und periphere Geräte
- Ein- und Ausgang- Sound-Funktionalität
- ggf. die Netzwerkverbindung zwischen den Tablets der Schüler

4.1.1. Möglichkeiten zum Umgehen und Durchbrechen von Rechte- und Funktionsentzug

- Tablets können gerootet sein - Das bedeutet, dass der Benutzer i.d.R. ohne nötiges Passwort Administratorrechte erlangt hat.
- Schadsoftware (Malware) kann den Sperrprozess unvorhersehbar beeinflussen.
- Serverprogramm / Clientprogramm / Protokoll kann mit Reverse-Engineering entgegnet werden, d.h. Das Überwachungsprogramm wird ersetzt durch einen Fork von diesem.

- Exploits können ausgenutzt werden, d.h. es werden Daten in das Programm eingeschleust, die nicht vorgesehen sind, die zu Reaktionen führen, die dem Benutzer bzw. Hacker Möglichkeiten eröffnet, die für ihn nicht vorgesehen sind.
- Netzwerkverkehr kann mit einem Sniffer, wie z.B. Wireshark ausgelesen werden, der bei TLS jedoch verschlüsselt vorliegt
- Ganz allgemein können Hintertüren (Backdoors) gefunden werden. Das bedeutet, dass eine beliebige Art von Sicherheitsschicht nicht durchbrochen, sondern umgangen wird.
- Reverse Engineering der Sperr-App mittels Decompilieren oder Lesen von Java-Bytecode
- Entschlüsselung von Verschlüsseltem und Einbruch in Accounts mit Brutforce, Wörterbuchangriff, Exploits, Eingriff in den Ablauf von Protokollen, z.B. Autorisierungs-Protokolle, Recherchieren von Sicherheitslücken
- 2 Schüler tauschen ihre Accounts gegenseitig aus.
- Diebstahl von Zertifikaten und privaten Schlüsseln, Extrahierung derer, durch den Schüler seines eigenen Tablets
- Schüler macht ein Tablet zu einem Honeypot, den der Lehrer sperrt. Er löst die Aufgaben schließlich auf einem nicht gesperrten Tablet.
- Der Lehrer-Rechner wird ausspioniert / ferngesteuert / verändert von den Tablet-Nutzern mittels einer Schadsoftware.
- Ab Android 4.3 kann man Rechte von Apps nachjustieren, d.h. der App vorher erlaubte Rechte später entziehen. Das kann ein Problem werden, wenn Tablet-Benutzer die Rechte der Überwachungs-App nachjustieren.

4.1.2. Sperrungen außerhalb des Betriebssystems

- Das Booten von anderen Medien muss im Setup des BIOS bzw. von UEFI deaktiviert sein. Ansonsten gibt es Vollzugriff auf Datenträger.
- Im Setup des BIOS oder in UEFI muss ein Passwort gesetzt sein, damit der Tablet-Nutzer nicht doch von anderen Medien booten kann, weil er ansonsten umstellen kann von welchem Medium gebootet werden kann.
- Sicherheits-Problem: In manchen Geräten, z.B. PC-Tower wird das Passwort und alle BIOS-Einstellungen gelöscht, wenn ein Akku kurzzeitig entfernt wird, und in der Fassung des Akkus im Mainboard elektrisch negativ und elektrisch positiv mit dem Schraubenzieher kurzgeschlossen wird. Auf dem Mainboard ist das ein 1 bis 2 cm großer zylinderförmiger Akku.

4.1.3. Notwendige software-gesteuerte Beobachtungen

Liegt ein Kioskmodus vor, so muss überwacht werden, ob dieser ggf. umgangen wurde, z.B. indem der Tablet-Nutzer die Festplatte / SSD ausgebaut, manipuliert und wieder eingebaut hat. Alternativen zu gewöhnlichen Festplatten sind ROM-Datenträger, die nur Lese-Rechte besitzen oder Datenträger die asymmetrisch oder hybrid verschlüsselt sind, für die der Tablet-Nutzer nur Leserechte besitzt, aber der Lehrer zusätzlich Schreibrechte.

Anstelle eines Kioskmodus kann auch registriert werden, ob der Tablet-User schummelt (in dieser Arbeit unter Android!). Deshalb müssen alle digitalen Möglichkeiten zu schummeln auf dem selben Gerät überwacht werden, in diesem Fall.

Beobachten bei Verwendung eines Kioskmodus in Windows:

- Modifikationen im Dateisystem
- Welche Dienste und Prozesse gestartet sind
- Hashwert der gestarteten Dienste und Prozesse
- Welche Accounts auf dem Tablet des Schülers offen sind
- Wie viele Accounts Administratorrechte besitzen

Beobachten ohne Kioskmodus in Android:

- Welche Dienste und Prozesse sind gestartet?
- Was ist der Hashwert der gestarteten Dienste und Prozesse?
- Ist etwas in der Zwischenablage schon vorher gespeichert?
- Welche weiteren Geräte sind, z.B. mit Bluetooth, verbunden?
- Auf welche Dateien wurde zugegriffen?

4.1.4. Authentifikations-Möglichkeiten

Es gibt eine ganze Reihe an etablierten Möglichkeiten mit der sich Lehrer und Schüler authentifizieren und autorisieren können. Zunächst muss sich der Server des Tablet-Nutzers und der Client des Lehrers sowieso zueinander authentifizieren mittels TLS-Sockets. Der Server muss bei sich TLS generell immer beim Client authentifizieren. Je nach Umsetzung, der Verwendung von Methoden der TLS-Bibliotheken ist programmierbar, ob Clients sich wie Server authentifizieren müssen, jedoch beim Server statt beim Client, mit Zertifikaten oder nicht. Für den Fall der Überwachungs-App und dem Kiosk-Fernwartungs-Programm ist Client-Authentifizierung jedoch prinzipiell verpflichtend, damit sichergestellt wird, dass der *Lehrer* wirklich die Tablets überwacht, die in dem Moment

für Klassenarbeiten verwendet werden. Problematisch wird es,wenn die Tablet-Nutzer Zugriff auf die Zertifikate haben und sich diese austauschen oder Honeypots erstellen.

Daneben gibt es die Möglichkeit mehrere Authentifikationsschichten zu verwenden. Das bedeutet, dass man sich mehrmals manuell anmelden muss, dies mehrmals automatisch geschieht oder beides kombiniert wird.

Folgende weitere Authentifikationsmethoden gibt es:

- RADIUS (Remote Authentication Dial-In User Service): Einstellungen von Benutzern lassen sich zentral verwalten. Es gibt Benutzernamen und Passwörter. Anwendung: Modems, VPNs, DSL, WLAN, ISDN. Es gibt Erweiterbarkeit für beliebige Funktionalitäten, z.B. Drosselung. Der Nachfolger heißt Diameter.
- Kerberos (ein verteilter Authentifizierungsdienst): Es gibt die Möglichkeit 3 Parteien zu haben: Client, Server und Kerberosserver, der die Authentifikation und Authorisierung des Servers und Clients mittels eines Protokolls regelt. Anwendung: u.a. im Active Directory via Windows Server 2012
- SASL wird von Protokollen zur Authentifizierung verwendet. Dabei wird eine Authentifikationsmethode von mehreren möglichen ausgehandelt. Verwendung findet SASL in: SMTP, IMAP, POP3, LDAP, XMPP (Jabber)
- JAAS ermöglicht Authentifikationen mittels: LDAP, SAML, PKI-Zertifikaten, SQL-Datenbanken
- Die lokale Anmeldung ist die Anmeldung mit einem Benutzeraccount des jeweiligen Betriebssystems.
- Denkbar als Zukunftsmodell ist, dass jeder Schüler eine Chipkarte besitzt mit der seine Identität eindeutig feststellbar ist und mit dem seine Lösungsabgabe verknüpft ist.

4.2. TLS

Für dieses Kapitel sind Grundkenntnisse in Kryptologie bzw. Kryptographie erforderlich. TLS ist ein Verfahren, um über ein Netzwerk mittels Sockets sicher zu Kommunizieren. Ab Version 3.1 wurde SSL in TLS umbenannt. Für TLS wird von digitalen Unterschriften, hybrider Verschlüsselung (d.h. Kombination aus symmetrischer und asymmetrischer Verschlüsseung), Zertifikaten, Hashfunktionen und Verschlüsselungsmodi wie z.B. „Cipher Block Chaining” Gebrauch gemacht.

Optional ist mit TLS die, für den Benutzer nicht direkt sichtbare, Authentifizierung des Clients beim Server möglich, d.h. nicht nur die des Servers beim Client. Verschlüsselung

nützt nichts, wenn einer der Parteien ausgetauscht wurde, die andere Partei davon nichts erfahren hat, und dann ein Schlüsselaustausch stattfindet, weil dann ein Angreifer einer der beiden Parteien sein kann. Damit die Tablet-Benutzer nur vom Lehrer-Client überwacht werden können, ist daher die programmiertechnisch optionale Authentifizierung des Clients in der Spezifikation von Anti-Cheat-Plus verpflichtend.

Der Tablet-Benutzer kann mit 2 Tablets arbeiten oder einem Tablet und einem darin emulierten Tablet. Dabei überwacht der Lehrer ein reales oder virtuelles Tablet fern, damit der Tablet-Benutzer mit diesem nicht Sicherheitslücken ausnutzt. Jedoch schreibt der Tablet-Benutzer die Klassenarbeit mit dem anderen virtuellen oder realen Tablet und beschafft sich dabei verbotene Informationsquellen, ohne dass der Lehrer dies auf elektronischem Weg bemerkt. Zwischen beiden Geräten herrscht eine Verbindung, so dass der Tablet-User am Ende die Arbeit zu dem anderen übertragen, und somit die Arbeit abgeben kann.
Damit der Lehrer sich sicher sein kann, dass er das richtige Tablet überwacht, muss sich dieses authentifizieren. Dazu wird mit TLS die Authentifizierung automatisch mit Zertifikaten und Schlüsseln durchgeführt.

TLS wird für die Spezifikation in das Überwachungsprogramm, mit Zugriff auf Bibliotheksfunktionen, eingebaut. TLS ist eine Spezifikation, die in verschiedenen Lösungen eingebaut wurde, z.B in jedem Browser und Webserver, in Dateimanagern und FTP-Programmen, die FTPS unterstützen, wie z.B: Filezilla. Darunter in Backends, Frontends (als GUI und kommandozeilenbasiert), Programm-Bibliotheken. Anwendung findet TLS als die Sicherheitsschicht von HTTPS, OpenVPN, einem Teil der Android-Apps. Als TLS-Programmbibliothek, die dazu dient damit andere Programme TLS-Sockets nutzen können, ist Openssl der bekannteste Vertreter und Keytool für Java von Oracle.

4.2.1. TLS mit Java

Für Java gibt es freie Implementierungen, um Kryptographie zu gewährleisten z.B. „Bouncy Castle" [1], u.a. für TLS. Jedoch bietet auch die grundlegende API von Java Möglichkeiten TLS zu realisieren. Dabei liefert Oracle das Kommandozeilenprogramm Keytool mit als Pendand zu Openssl, jedoch mit geringerer Anzahl an Funktionen und proprietären Formaten. Die Formate von Openssl und Keytool lassen sich ineinander umwandeln. Unter [3] findet sich ein Vergleich von Keytool und OpenSSL. Um offizielle Zertifizierungsstellen verwenden zu können, die für Server nötig sind, wenn ein Client das Serverzertifikat nicht validieren kann, um es zu validieren, reicht Keytool nicht aus, denn das Signieren von Zertifikaten unterstützt hier nur Openssl. Wenn man mit Java einen Webserver mit Webseite programmiert, so ist es sicherheitstechnisch notwendig das Webseiten-Serverzertifikat von einer CA unterschreiben zu lassen, weil diese bei Ausklammerung

[1] https://www.bouncycastle.org/

der Problematik mit den Geheimdiensten eine vertrauliche Quelle sind. Für das Unterschreiben durch die CA (Zertifikatsstelle) müssen Telefonnummer, Adresse, Ausweisnummer und anderen Daten zur Identifikation hinterlassen werden. Ein Zertifikat zu einer Webseite muss dabei den Domainnamen dieser Webseite beinhalten, um Sicherheit zu gewährleisten. Durch diese Identifikation und Identifizierbarkeit wird sichergestellt, dass nicht mit dem falschen Partner ein verschlüsselter Datenaustausch stattfindet.

4.2.2. TLS in Java programmieren

Bild 4.1 zeigt den Aufbau der Quellen des Java-Android- und Standard-Java-Projektes, mit denen die Funktionstüchtigkeit und die Laufzeiten getestet wurden.

Der Quellcode befindet sich im Anhang. Die Anleitung unter [4] hat beim Erstellen dieser Arbeit zum Erfolg geführt. Es ist aber auch möglich allein mit der API-Referenz von Oracle TLS-Sockets umzusetzen [5].

Mit der Methode setNeedClientAuth(true), programmiert auf Server- und Clientseite, wird die clientseitige Autorisierungsverpflichtung aktiviert.

Neben der Programmierung muss sich um die Schlüssel und Zertifikate und Container gekümmert werden. Der Client verweist auf einen Keystore und Truststore und der Server jeweils auch. Diese Container haben ein proprietäres Format von Oracle und sind beide gleich aufgebaut, jedoch für einen unterschiedlichen Zweck. Der Unterschied zwischen einem Truststore und einem Keystore besteht darin, welche Schlüssel und Zertifikate sie beinhalten und für welchen Zweck[6]. Der Keystore beinhaltet private Schlüssel und Zertifikate zu entsprechenden öffentlichen Schlüsseln, die angefordert werden, wenn dieser Keystore, einer eines Servers ist, oder wenn Client-Authentifikation angefordert wird. Ein TrustStores beinhaltet Zertifikate von dritten mit denen die eigene Java-Applikation kommuniziert oder er beinhaltet Zertifikate, die von einer CA (Zertifizierungsstelle) unterschrieben sind, womit die Identität sichergestellt werden kann. Der TrustStore wird benötigt, um zu bestimmen welche Verbindung vertrauenswürdig ist, ob „der andere" derjenige ist, der er vorgibt zu sein. Mit dem Keystore wird entschieden, welche Beglaubigung zum entfernten Rechner gesendet wird für den Verbindungsaufbau (Handshake). Für die in TLS verpflichtende Authentifikation des Servers auf der Clientseite, werden Zertifikate im Truststore verwendet.
Der Keystore enthält private Schlüssel, die nur dann gebraucht werden, wenn ein Server betrieben wird oder wenn Client-Authentifikation aktiviert ist auf der Serverseite. Der Truststore beinhaltet öffentliche Schlüssel und Zertifikate von der CA, die benötigt werden, wenn das Vertrauen der jeweils anderen Seite bestätigt werden muss.
Man kann die gleiche Datei als Trust- und Keystore verwenden, wenn das persönliche Zertifikat mit dem Zertifikat des Signierers darin gespeichert ist.

Abbildung 4.1.: Eclipse, TLS-Programmierung

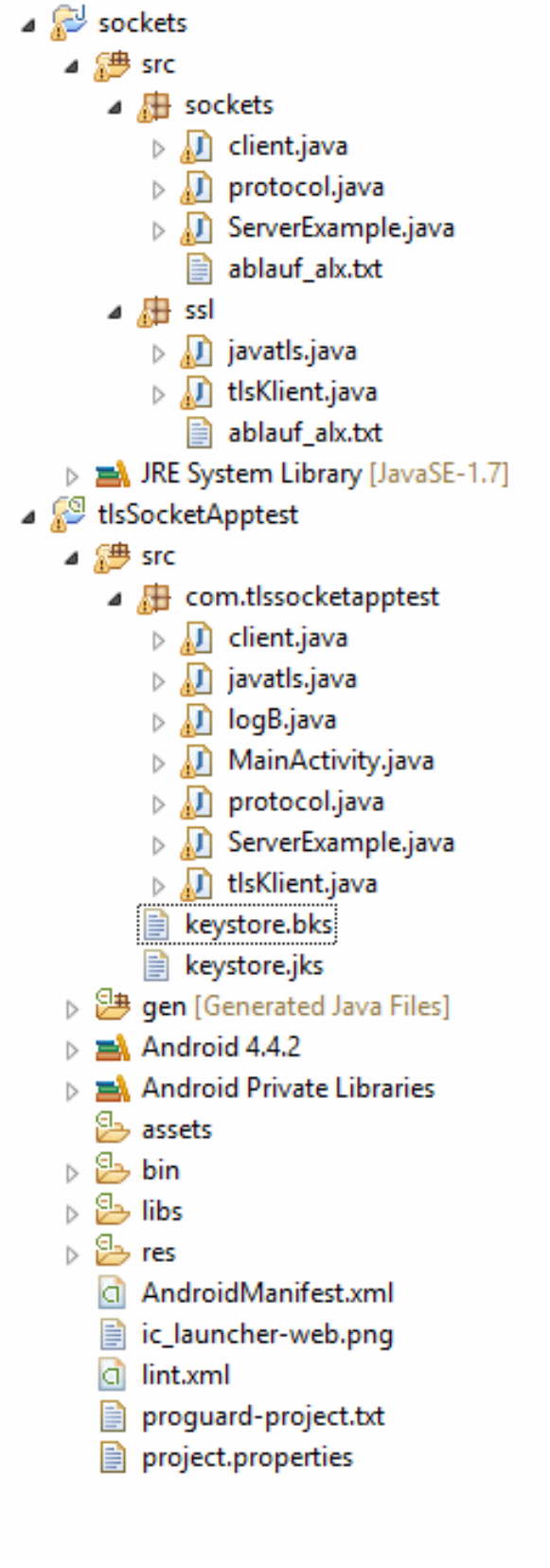


Tabelle 4.1.: TLS Dateiformate

Format	Dateiendungen	Bedeutung / Funktion
DER	.der, .crt	X.509-Zertifikate
PEM	.crt .pem, .csr.pem, .key.pem	Zertifikate, Schlüssel, CSR
CSR	.csr, .csr.pem	i.d.R. Unterschreibungsanfragen
JKS	.jks	Java Keystore und Truststore
PKS#12	.pfx, .p12	Container: Schlüsselpaare+Zertifikate
CER	.cer	Zertifikat u.a. für Java
PKCS#8	.key	öfftl. und priv. Schlüssel

Zertifikate enthalten mindestens den öffentlichen Schlüssel und einige textliche Informationen [7] : Name / Alias, Ablaufdatum, Serialnummer, Name der Organisation, Unterschrift einer Zertifizierungsstelle.

Das Format PKCS #12 (einer der PKCS Public-Key-Standards) definiert ein Containerdateiformat um mehrfach passwortgeschützt private Schlüssel und beiligende Schlüssel zu speichern. Der Java Keystore nutzt dieses Format. Clientzertifikate haben das Format PKCS#12.

Unter [3] und [8] finden sich verschiedene Formate, die für TLS Eine Bedeutung haben. Diese wurden in Tabelle 4.1 zusammengefasst. Die Begriffe in dieser Tabelle, die nicht selbsterklärend sind, werden in diesem Abschnitt 4.4.2 erklärt, u.a. weil sie auch noch in Tabelle 4.2 vorkommen.

Es ist möglich Formate ineinander umzuwandeln. In unterschiedlichen Programmiersprachen und Betriebssysteme können unterschiedliche Formate Anwendung finden. Außerdem unterstützt das Oracle Keytool kein Unterschreiben, weshalb Umwandlungen nötig sein können, wenn mit Openssl signiert wurde, denn Keytool kann nicht signieren.

Die Tabelle 4.2 vergleicht Keytool mit OpenSSL und einer jeweils eigenen Neuentwicklung, sofern jemand eine Alternative zu Keytool und OpenSSL haben will Eine CRL ist eine Liste gesperrter Zertifikate (viertletzte Zeile). OCSP ist ein Protokoll, um Signaturen zu überprüfen, siehe letzte beide Zeilen in dieser Tabelle 4.2. Eine CA ist eine Zertifizierungsstelle für digitale Zertifikate, als Wurzel einer Public-Key-Infrastructure (PKI). PKCS #1 bis #15 sind Kryptographie-Standards. CSR ist eine Anfrage, damit ein Zertifikat unterschrieben werden kann. Ansonsten wird in der Tabelle verglichen, welche Operationen und Formate verwendet werden können, welche Informationen aus welchen Dateien gewonnen werden können, welche Schlüssel und Zertifikate in welche Containerformate gelagert und extrahiert werden können.

Tabelle 4.2.: Vergleich Keytool mit Openssl

Die Tabelle ist aus [3] zitiert:

Anwendungsfall	Keytool	Openssl	eigenes
RSA-Schlüssel erzeugen	ja	ja	ja
CSR erzeugen	teils	ja	ja
selbstsigniertes Zert. erzeugen	teils	ja	ja
Zert. aus CSR. signieren	nein	ja	ja
Infos aus JKS anzeigen	ja	nein	ja
Infos aus PEM anzeigen	nein	ja	ja
Schlüsselpaar als PEM speichern	nein	ja	ja
S-Paar aus PEM in JKS imporieren	nein	nein	ja
S-Paar aus PKCS#12 in JKS importieren	teils	nein	ja
Schlüssel und Zert. in PKCS#12 wandeln	nein	ja	ja
Zert. in JKS importieren	ja	nein	ja
Verwendungs als CA	nein	ja	ja
CRL erzeugen	nein	ja	ja
Zert. gegen CRL prüfen	nein	ja	ja
TLS-Verbindung testen	nein	ja	ja
OCSP-Testserver betreiben	nein	ja	ja
Zertifikat gegen OCSP prüfen	nein	ja	ja

Der TLS-Handshake wird auf einer Seite von Microsoft ausführlich beschrieben: [8]
In Java wird dieser mit der Methode startHandshake() in Gang gesetzt, nachdem er mit mehreren Code-Zeilen vorbereitet wird.

Der Handshake ist der systematische Verbindungsaufbau in beschriebenen Schritten, der durch die Verwendung von Programmbibliotheken stattfindet. Man muss nun nicht diese ganzen Schritte direkt so programmieren. Man muss diesen Prozess im Quelltext vorbereiten und initieren, sofern man eine TLS-Verbindung aufbauen möchte. Der Aufruf des Handshakes ist dabei nur ein Funktionsaufruf einer Funktion.

Ein selbst signiertes Zertifikat ist ein Zertifikat, das durch den eigenen privaten Schlüssel signiert wurde. Damit kann die Identität eines Servers nicht mehr sichergestellt werden, weil die CA fehlt, die dessen Identität sicherstellt, es sei denn das Zertifikat war beim Client schon bekannt. Wenn dann nicht diesem Zertifikat vertraut wurde, weil es schon vorher bekannt war, kann jeder das Zertifikat erstellt haben mit zugehörigem Schlüsselpaar. Auf die Art sind verschiedene Angriffe denkbar, mit Vortäuschung falscher Schlüssel. Wenn das selbstsignierte Zertifikat nicht vorher der anderen Seite bekannt war (z.B. dem Client), dann lässt sich nicht ohne Weiteres überprüfen, ob es das echte Zertifikat ist. Für so etwas gibt es CAs. Trotzdem akzeptiert Java selbstsignierte Zertifikate.
Die Tabelle 4.3 zeigt, welche Schlüssel und Zertifikate in welcher der beiden Containerdateien vorgesehen sind, und auf welcher Seite, Server oder Client. Im Quellcode

Tabelle 4.3.: TrustStore und Keystore Containerinhalt mit aktivierter Clientauthentifikation Statt je ein Zertifikat, sind auch mehrere möglich.

Rechner	Container	Inhalt
Server	Keystore	Servers privater Schlüssel, selbst-/CA-signiertes Zert.
	Truststore	Servers selbstsigniertes Zert. oder CA Zert. des Clients
Client	Keystore	Clients privater Schlüssel, server-/CA-signiertes Zert.
	Truststore	Server selbstsigniertes Zert. oder CA Zert. des Servers

müssen die beiden Dateien auf beiden Seiten referenziert werden. Des Weiteren zeigt die Tabelle welchen Zustand die Dateien besitzen müssen, signiert / selbstsigniert / nicht signiert.

Damit die Verbindung sicher ist, muss mit *SSLSocket.setEnabledProtocols* generell eine möglichst hohe Versionsnummer von TLS gewählt werden, weil mit fortlaufenden Jahren vorherige Versionsnummern nicht mehr sicher sind. Dazu muss mit *SSLSocket.setEnabledCipherSuites* sichergestellt werden, dass nur zum jeweils derzeitigen Zeitpunkt sichere Krypto-Algorithmen eingesetzt werden für hybride, symmetrische, asymmetrische Verschlüsselung, die Streufunktion, das Schlüsselaustauschverfahren und den Modus also z.B. der „Cipher Block Chaining Mode". Honeypot-Encryption gibt es bisher noch nicht in TLS, eine Methode bei der ein Angreifer im Glauben gelassen wird, er hätte den Text entschlüsselt, da nicht nur ein Schlüssel Ergebnisse bringt. Aktuell (August 2014) gilt nicht nur MD5 und DES als unsicher. Die Sicherheit von SHA-1, 3DES und RC4 wird momentan von einem Teil der Experten angezweifelt, und ist nicht immer 100% geklärt, aber auf Details dazu geht diese Arbeit nicht ein. Ab Java 8 und Android L (Nachfolger von Android 4.4) ist SHA-2 einsetzbar bei Verwendung der Oracle API oder der API von Google. Ab dem 2. Oktober 2012 darf ein Verfahren offiziell SHA-3 genannt werden [9]. Da ein Gerät mit Android 4.1 vorlag, war SHA-2 nicht verwendbar. SHA-1 kann für TLS in Ciphersuiten jedoch problemlos eingesetzt werden, denn Kollisionsangriffe stellen keine Gefahr da, wenn Man-In-The-Middle-Angriffe nur mit Leserechten einbezogen werden.

Die Programmierung der TLS-Initialisierung weicht bei Android von der von Oracle Java ab, Quellcode siehe mitgelieferte CD. Android verwendet ein anderes Containerformat (BKS statt JKS) und andere Strings für die Krypto-Verfahren der Ciphersuiten.

Die Beweise der Funktionstüchtigkeit der TLS-Implementierung sind die Messergebnisse in Kapitel 7.

4.3. Netzwerkprotokoll

Beim Entwurf von Netzwerk-Software ist es möglich vorhandene Protokolle zu übernehmen, diese weiter zu entwickeln oder ein neues Protokoll zu entwickeln. Im Fall von Anti-Cheat-Plus wurde sich dafür entschieden ein neues Protokoll zu entwickeln und Entwurfsprinzipien vorhandener Protokolle zu übernehmen, siehe Unterabschnitt 4.3.1. Es wurden zunächst keine dokumentierten Protokolle für fernwartbare Kioskmodi gefunden.

Im Unterabschnitt 4.3.2 wird die Designentscheidung zu asynchronen Sockets begründet und auf spezielle Probleme beim Debugging eingegangen.

Darauf folgt in Unterabschnitt 4.3.3 die Begründung der Designentscheidung einen Windows-Dienst zu verwenden.

In Abschnitt 4.3.4 wird schließlich die Funktionalität vom Protokoll für Windows spezifiziert.

4.3.1. Design eines neuen Protokolls

Es werden Bytes (Datentyp „Byte”) übertragen, die ASCII codiert sind. Jeder Befehl endet nach seinen Argumenten mit der Zeichenkette „<EOF>”, zwischen den Argumenten und dem Befehlsnamen muss diese das Leerzeichen trennen. In einer Switch-Case-Anweisung wird zu dem passenden Befehl die passende Methode herausgesucht, die diesen ausführt. Wenn der Befehlsname aus mehreren Wörtern besteht, werden diese mit dem Unterstrich „_” getrennt oder die Teilwörter beginnen mit einem Großbuchstaben. Die Befehlsnamen sind in Englisch.

Es gibt z.B. die Befehle unter Windows:

- „GetAccountinfos” - Damit werden die lokalen Benutzeraccounts gelistet.
- „Accounts_Disable” und „Accounts_Enable” mit denen Windowsbenutzer-Accounts deaktiviert und aktiviert werden können, damit sich ein Tablet-Nutzer nicht nebenbei in einen Nicht-Kiosk-Account einloggen kann. Um ein Administratorkonto deaktivieren zu können, müssen zwischendurch die Admin-Rechte entzogen und nach dem Deaktivieren wieder hinzugefügt werden. Ansonsten verweigert das Deaktivieren Windows!

Erweiterbar ist dieses Protokoll insofern, dass Anti-Cheat-Plus beliebig abgeändert werden kann, da es sich nicht um keinen Standard handelt und es keine weiteren Programme gibt, die dieses Protokoll einsetzen werden. ASCII wurde mit dem Ziel verwendet, dass das Protokoll menschen-lesbar ist und für Englisch genügt ASCII. Eine Erweiterung des Protokolls hindert nicht daran, von ASCII auf eine andere Kodierung zu wechseln.

Damit das Protokoll sicher ist, müssen alle Eingabedaten zur Abwehr von Exploitausnutzungen validiert werden, z.B. durch Black- (inklusv-)oder Whitelisting von erlaubten und verbotenen Befehlen und Befehlsbestandteilen, z.B. Parameter und mittels regulären Ausdrücken, die beschreiben, was erlaubt (exlusiv-)oder verboten ist. Verschiedene Pufferüberläufe müssen abgefangen werden (z.B. Datentyp-Unter und Überläufe), die durch Socket-Eingabedaten hervorgerufen werden. Es muss überprüft werden, ob Eingabedaten bestimmte Systemfunktionen bedienen bzw. direkt oder über Umwege mit der Programmlogik erreichen können. Weitere und Speziellere Maßnahmen finden sich in entsprechender Literatur.

Deadlocks und Livelocks spielen zumindest für das Protokoll keine Rolle, weil es keine unterschiedlichen Threads innerhalb des Protokollbereiches im Quelltext gibt, so dass keine Variablen synchronisiert werden müssen. Insgesamt, außerhalb des Bereiches des Protokolls, muss jedoch synchronisiert werden, weil bei asynchronen Sockets Threads eingesetzt werden. Das Protokoll ist nur prototypisch programmiert worden.

4.3.2. Sockets

Auf die Grundlagen von Socket-Programmierung wird in dieser Arbeit nicht eingegangen.

Es ist möglich synchrone Sockets zu verwenden und mit Threads zu arbeiten, damit es keine Blockierungen gibt, wenn auf eine Nachricht gewartet wird. Etablierte Programme verwenden asymmetrische Sockets, die auch mit Threads arbeiten. Synchrone Sockets blockieren Lese- und Schreibzugriff, asynchrone nicht. In der Programmierung von asynchronen Sockets wird ein Event zu einem Eventhandler gesendet und es können weiter Daten empfangen oder gesendet werden. Synchrone Sockets senden oder empfangen Daten und blockieren so lange, bis der Übermittlungs-Prozess zuende geführt wird, und können dann erst wieder senden oder empfangen. Asynchrone Sockets haben eine komplexere umfangreichere Grundstruktur im Quelltext, als synchrone Sockets, aber dadurch ist schon eine gewisse Modularität vorgegeben auf der aufgebaut werden kann. Sie erweitern synchrone Sockets. Wenn man programmiertechnisch etwas ausprobieren will eignen sich synchrone Sockets, um schnell Ergebnisse zu haben. Für Anwendungsprogramme, die z.B. für den Verkauf bestimmt sind, eignen sich asynchrone Sockets, weil die Blockierung synchroner Sockets dabei nicht jedesmal behandelt werden muss, um z.B. weitere Clients zum Server verbinden zu lassen.

Im C#-Quelltext des prototypischen Fernwartungsprogramms wurden asynchrone Sockets umgesetzt, siehe Anlage.

4.3.3. Windows-Dienst

Unter [10] findet sich eine funktionierende Anleitung zum Programmieren von Windows-Diensten. Ein Dienst muss kein Netzwerk-Server sein. Dienste sind Prozesse, die in speziellen Registry-Einträgen verlinkt sind. Sie können noch mehr Rechte haben als der Administrator und zwar Systemrechte. Das ist auswählbar im Quellcode. Windows-Dienste sind Hintergrundprozesse mit gewissen Grundfunktionalitäten, d.h. Starten, Stoppen, Anhalten, Fortsetzen.

Ein Dienst bietet sich für Anti-Cheat-Plus an, weil Tablet-Benutzer mit Benutzerkonten ohne Administratorrechte keinen Einfluss auf das Starten und Beenden von Diensten haben.

4.3.4. Fernwartung unter Windows

Für Windows wurde im Rahmen dieser Arbeit Anti-Cheat-Plus als Fernwartungsprogramm prototypisch programmiert.
Anti-Cheat-Plus für Windows muss beherrschen:

- Auflisten der lokalen Windowsbenutzernamen, und ggf. deren Daten
- Deaktivieren von (lokalen) Windows-Accounts, damit nur noch der Kioskbenutzeraccount verfügbar ist
- Aktivieren von (lokalen) Windows-Accounts, damit sich der Tablet-User wieder in seinen Account einloggen kann
- Deaktivieren von Geräten, z.B. Bluetooth (zur Sicherheit), Sensoren, alles was im Windows-Gerätemanager an Geräten auftaucht
- Aktivieren von Geräten
- Unicast-Ping vom Lehrer-Client zum Schüler-Server auf dem Tablet
- Broadcast-Ping vom Lehrer-Client zum Schüler-Server auf dem Tablet
- Erweiterbarer Sperrungsbefehl für zu Definierendes zu Sperrendes

Es kann passieren, dass der letzte aktive Administrator-Account deaktiviert wird und sich dann kein Administrator mehr einloggen kann. Deshalb muss die Fernwartung einen Administratoraccount hinterlassen, der mit einem Passwort versehen ist, das vom Client des Lehrers festgelegt wurde und somit nur der Lehrer kennt. Der Tablet-Benutzer darf **niemals (!)** Administratorrechte gehabt haben und haben werden, damit der Kioskmodus sicher funktionieren kann, d.h. nicht manipuliert werden kann.

4.3.5. Atomare Netzwerkbefehle

Eine der 4 ACID-Eigenschaften (Datenbanken) ist die Atomizität.

Im Beispiel von Anti-Cheat-Plus für Android oder Windows bietet die Atomizität die Funktion, dass wenn ein Ablauf bei dem Daten geschrieben werden mittendrin unterbrochen wird, wieder der ursprünglicher Zustand hergestellt wird, der vor dieser Schreiboperation vorlag. In der Praxis bedeutet das, dass entweder die ganze Operation durchgeführt wird oder diese gar nicht durchgeführt wird. Deswegen nennt sich das atomisch.

Um Atomizität umzusetzen müssen die betroffenen Daten dupliziert werden, die jeweiligen Änderung in den duplizierten umgesetzt werden und am Ende werden Zeiger auf die neuen Daten gelegt (welche Art von Zeigern auch immer, z.B. symbolische Links im Dateisystem). Im Fehlerfall wird ein Zeiger wieder auf die alten Daten gesetzt und das Duplikat gelöscht.

Wenn die Unterbrechung jedoch so aussieht, dass das zugehörige Netzwerkprogramm beendet oder das System neu gestartet wird, muss direkt nach dem Systemneustart mit Anti-Cheat-Plus, ggf. mit einem zusätzlichen Dienst, die Reparatur automatisch durchgeführt werden, damit Atomizität sichergestellt wird.

Atomische Netzwerkbefehle haben den Vorteil, dass Unterbrechungen nicht zu inkonsistenten Systemzuständen führen können, was im Extremfall bedeuten kann, dass das Betriebssystem neu aufgesetzt werden muss.

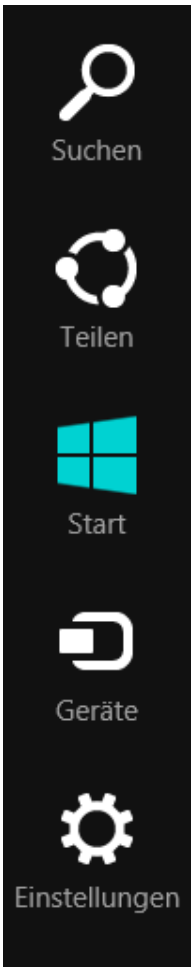
4.4. Windows-API und nutzbarer Code ähnlicher Projekte

Betriebssystemfunktionen, die ins System eingreifen sind in der Regel weniger ausführlich dokumentiert, als die Bereiche der Windows-API, die sich an Programmieranfänger richten. Von daher ist es nützlich vorhandenen Quellcode zu lesen, von quelloffenen Kioskmodi oder Programmen die ähnliche Betriebssystemfunktionen verwenden müssen. Diese Information in diesem Abschnitt ist nützlich, wenn ein Programmierer mit Systemfunktionen arbeiten muss. Dann weiß er, dass er sich früher dazu entscheiden kann, nach Quellcode von vorhandenen Projekten zu suchen und diesen zu lesen, weil das schneller zum Ziel führen kann, das er im Auge hat, als Dokumentationen zu lesen.

Versionen der Classic Shell[2] vor 3.9.0 sind quelloffen. Die Classic Shell ersetzt den Start-Button von Windows 8 / 8.1 mit dem Startmenü von Windows 7, das komplett nachgebildet wurde, und nicht versteckt in Windows 8 / 8.1 vorhanden ist. Dazu muss auf die API der Taskleiste zugegriffen werden und das spielt eine Rolle bei der Programmierung eines

[2] http://www.classicshell.net/

Abbildung 4.2.: Charmsleiste



Kioskmodus. Es gibt nämlich Kioskmodi, die die Taskleiste verändern und den Desktop versperren. Wenn sich dafür entschieden wird, einen Kioskmodus so zu gestalten, dann ist es zielführend den Quellcode der Cassic Shell zu untersuchen.

Der Kioskmodus, der im Rahmen dieser Masterarbeit für Windows 8.1 bewerkstelligt wird, benötigt keine Manipulation der Taskleiste mittels einer Programmierung. Die Gruppenrichtlinien von Windows genügen für diese Arbeit, um Funktionen der Taskleiste zu sperren.

4.4.1. Deaktivieren der Charmsleiste (Windows)

Die Charmsleiste erscheint in Windows 8.1 rechts durch verschiedene Wischgesten. Von da aus kann man u.a. zu den Einstellungen navigieren. Ein Kioskmodus hat eingeschränkte Rechte, und auch wenn mit den Gruppenrichtlinien der Zugang zur Systemsteuerung blockiert werden kann, ist es ein zusätzlicher Schutz die Charmsleiste zu deaktivieren.

Die Charmsleiste ist zu Deaktivieren, weil sie für Klassen- und Gruppenarbeiten störend sein kann und andererseits ein Sicherheitsproblem darstellt, wenn der Tablet-Nutzer von da aus die Konfigurationen von Windows ändern, nach Dateien suchen und Dateien mit anderen teilen kann, und sich somit in Klassenarbeiten unerlaubte Informationen beschaffen kann.

Das Deaktivieren der Charmsleiste wird erreicht, indem der Registry-Editor gestartet wird, durch Ausführen des Programmaufrufs „regedit". In der Verzeichnishierarchie des Registry-Editors wird dann navigiert zu:
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\ImmersiveShell .
Dort legt man einen Schlüssel an namens EdgeUI , wenn er noch nicht vorhanden ist. Darin wird ein DWORD-Wert eingefügt, 32 Bit. Dieser hat die Bezeichnung DisableCharmsHint, und der eingetragene Wert muss 1 sein. Ist damit die Charmsleiste noch nicht deaktiviert, so hilft in der Regel ein Ab- und Anmelden nach dem Verändern von Registryeinträgen, damit die Wirkung von solchen Registry-Einträgen einsetzt.

4.4.2. Kioskmodus des Internet Explorers

Der Internet Explorer hat einen eingebauten Kioskmodus, siehe [10] .
Allein dieser Modus funktioniert noch nicht als Betriebssystem-Kioskmodus. Deshalb muss man den Kiosk-Modus des Internet-Explorers mit dem des Betriebssystems kombinieren. Ansonsten kann man z.B. einen weiteren Browser installieren oder anderweitig die Sicherheitsvorkehrungen umgehen. In dieser Arbeit wird nicht näher auf den Kioskmodus des Internet Explorers eingegangen, da das zu dem Thema Webseitensperrungen gehört.

4.5. Einzelne Sperrungsbereiche

In den folgenden Unterabschnitten befinden sich einzelne Angelegenheiten, die einschränkt werden müssen, damit Tablet-Nutzer Prüfungen schreiben können.

4.5.1. Zwischenablage in Android

Die Zwischenablage (englisch: Clipboard) kann verwendet werden, um darin Informationen zwischenspeichern zu lassen, z.B. Schulstoff, auf den in einer Prüfung zurückgegriffen werden kann. Wenn ein Poolnutzer diesen Inhalt unerlaubterweise verwendet, dann ist das ein Verstoß. Deshalb muss die Zwischenablage in dieser Arbeit thematisiert werden.

Für Windows ist die Zwischenablage nicht relevant, weil sich Tablet-User neu einlog-

Algorithmus 4.1 Android Zwischenablage leeren

```
1 super.getSystemService(CLIPBOARD_SERVICE).setText(null)
```

Abbildung 4.3.: Autocomplete / Autovervollständigung

gen müssen und auf dem neuen Account haben Benutzer keinen Zugriff mehr auf die Zwischenablage eines anderen Benutzer-Accounts.

Es gibt Geräte mit Android, die mehrere Einträge in der Zwischenablage halten können. Ansonsten ist es so, dass, wenn mehrere Einträge in der Zwischenablage von Android sind, diese zu einer Selektion gehören, die mit einem Mal kopiert werden und eingefügt werden. Die Zwischenablage von Android kann im Normalfall geleert werden, indem ein leerer String in diese hineinkopiert wird, siehe Algorithmus 4.1.

Es gibt Apps, die das Android-Clipboard (Zwischenablage von Android) erweitern, so dass mehrere Einträge in der Zwischenablage vorhanden sein können, also mehrere Selektionen von einem Datum oder mehreren Daten statt, wie bei einer normalen Zwischenablage, einer Selektion von einem Datum oder mehreren Daten. Man kann also in einer erweiterten Zwischenablage mehrmals Text kopieren und dabei werden vorherige Kopien in der Zwischenablage nicht gelöscht, so dass man alle Kopien beliebig geordnet in einem Textfeld einfügen kann.

Solche Sonderfälle behandelt diese Arbeit nicht programmiertechnisch. Erwähnt werden muss das trotzdem, da berücksichtigt werden muss, ob die betreffenden Tablets nicht doch eine modifizierte Zwischenablage besitzen. Ansonsten bergen erweiterte Zwischenablagen ein Betrugsrisiko.

4.5.2. Formularfelder mit Autovervollständigung unter Android

Bild 4.3 zeigt beispielhaft, wie eine Autovervollständigung aussieht.

Unter [11] wird beschrieben, wie die Autovervollständigung (englisch: autocomplete) unter Android implementiert wird.
Wird unter Android in einem Programm eine Klassenarbeit geschrieben oder an einer Gruppenarbeit gearbeitet, dann sind Betrugsversuche möglich, indem vorher Wörter, durch vorheriges Tippen und Übernehmen, in der Autovervollständigung gespeichert werden. Wenn die Prüfung geschrieben wird, kann auf diese vorhandenen Eingaben ggf. zurückgegriffen werden. Administratoren, Lehrer oder andere Zuständige müssen deshalb sicherstellen, dass das Programm z.B. der Browser, mit dem die Klassenarbeit geschrieben wird keine Autovervollständigung anbietet. Ggf. muss auf Basis eines Browsers ein neuer Browser geschrieben werden, ohne aktivierbare Autovervollständigung oder entsprechende temporäre Dateien oder Browserprofildateien müssen, ggf. automatisch, gelöscht werden.

Es gibt heutzutage Apps, die in HTML usw. geschrieben sind. Dies sind keine gewöhnlichen Webseiten für einen Browser, sondern eigenständige Programme. Das ist insofern relevant, weil dafür Formularfelder mit Autovervollständigung automatisch integriert sein können und das vom HTML, CSS oder Javascript abhängt. Das bedeutet, dass zur Untersuchung nicht nur Java-(Byte)-Code untersucht werden darf. Z.B. ist es auch möglich in C++ für Android zu programmieren wobei die Apps trotzdem in einer Sandbox ausgeführt werden.

Es ist möglich, eine ausführbare Datei (*.exe / Java-Bytecode / Shell-Skript / *.msi / etc.) danach analysieren zu lassen, welche Betriebssystemfunktionen sie verwendet. Das geht mit der Methode, mit der Antiviren-Programme Malware finden. Das bedeutet, jedes Antivirenprogramm verwendet diese Methode, z.B. Norton Antivirus, oder Avira Antivir. Die Methode ist das Parsen von DLL-Imports, also die Untersuchung welche Bibliotheks-Funktionen ein Programm (auf welche Art) nutzt. Damit kann herausgefunden werden, ob Formulartexte im Speicher gehalten werden.

Die Programmierung eines Parsers ist nicht Teil dieser Arbeit, weil das den Rahmen sprengt.

4.5.3. Bluetooth unter Android

Bluetooth muss deaktiviert werden, weil es die Möglichkeit bietet, dass Daten von externen Geräten empfangen und zum Tablet des Tablet-Nutzers gesendet werden können, weil die anderen Geräte nicht überwacht werden und weil es aufwändiger ist, den Datenverkehr zu überwachen, als Bluetooth auszuschalten. Aktiviertes Bluetooth gilt somit als Betrugsversuch, wenn es zuvor durch Anti-Cheat-Plus deaktiviert wurde, wie in diesem Unter-Abschnitt beschrieben wird.
Es ist notwendig den Bluetooth-Status, ob aktiviert oder deaktiviert, in Zeitabständen dem

Algorithmus 4.2 Bluetooth unter Android deaktivieren

```
1  BluetoothAdapter mBluetoothAdapter =
2  BluetoothAdapter.getDefaultAdapter();
3  if (mBluetoothAdapter.isEnabled()) {
4    Toast.makeText(getApplicationContext(),
5    "bt isEnabledWillBeDisabled",
6    Toast.LENGTH_SHORT).show();
7    mBluetoothAdapter.disable();
8  } else {
9    Toast.makeText(getApplicationContext(),
10   "bt wasDisabled",
11   Toast.LENGTH_SHORT).show();
12 }
```

Überwachungs-Computer der Aufsichtsperson zu melden, um Betrugsversuche auszuschließen.

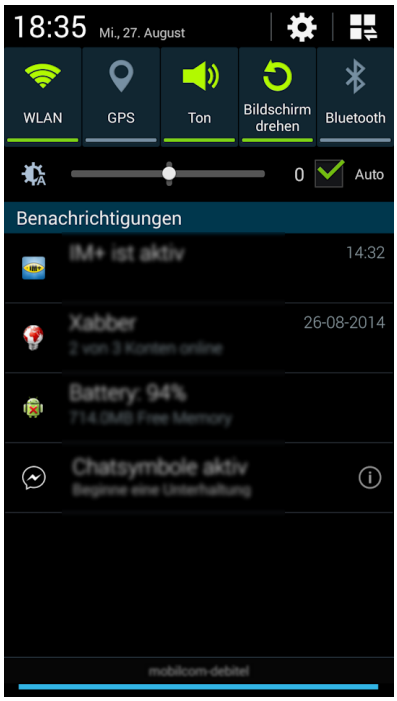
Im Manifest müssen die Rechte "android.permission.BLUETOOTH" und "android.permission.BLUETOOTH_ADMIN" erlaubt werden. Wenn man diesen Code in dem Emulator zum Überprüfen der Funktionstüchtigkeit des Codes ausführt, kommt, wegen einem Null-Zeiger, ein Fehler. In einem Android-Gerät dagegen, das Bluetooth unterstützt, funktioniert das Ganze jedoch.

Der Tablet-Benutzer kann Bluetooth wieder aktivieren. Von daher macht es Sinn, den Status darüber, ob Bluetooth aktiviert ist, dem Lehrer-Computer mit Anti-Cheat-Plus zu melden, in Abständen von z.B. 5 Sekunden. 5 Sekunden reichen nicht für ein manuelles Ausnutzen von Sicherheitslücken zur Informationsbeschaffung aus. Selbst wenn der Tablet-Benutzer ein Skript programmiert hat, weiß er nicht, wann die 5 Sekundenzeit beginnt. Dazu muss er einen Netzwerkscanner im Einsatz haben und diesen in sein Skript einbeziehen. Jedoch wird dem Lehrer so eine Anomalie gemeldet, weil mit Anti-Cheat-Plus gesendet wird, welche Apps aktiv sind. Also werden auch solche verdächtigen Apps wie der Netzwerkscanner gemeldet.

Nicht nur für Bluetooth relevant, aber hier [12] ist beschrieben, wie man abfragt, ob ein Android-Gerät überhaupt GPS oder andere Features unterstützt.

Algorithmus 4.2:
In der ersten bis zweiten Zeile wird die Objektinstanz, für die Verwaltung des Standard-Bluetooth-Adapter, zugewiesen. In der Regel gibt es nur einen Bluetooth-Adapter. In der dritten Zeile wird mit isEnabled() abgefragt, ob der Bluetooth-Adapter aktiviert ist oder nicht. Toast ist ein Objekt zum direkten Ausgeben von Nachrichten. „mBluetoothAdapter.disable();” deaktiviert den Bluetooth-Adapter in Zeile 7.

Abbildung 4.4.: Notification-Drawer



4.5.4. Sensoren unter Android

Unter [13] ist die detaillierte Übersichts-Seite über Sensoren für Entwickler von Android abrufbar. Dort sind alle Sensoren beschrieben, welche Android theoretisch und praktisch unterstützt. 13 Arten von Sensoren sind es zum Zeitpunkt dieser Arbeit (2014).

Stellen Sensoren wie z.B. Gravitations-, Temperatur-, Luftdruck-, Luftfeuchtigkeitssensoren usw. in einer Prüfung eine Sicherheitslücke dar? Die Methoden Informationen über Sensoren in das Tablet zu schleusen, z.B. durch bewusst erzeugte Temperaturschwankungen, sind nicht praktikabel, um als Sicherheitslücke genutzt zu werden. Deshalb müssen Sensoren nicht zwingend deaktiviert werden. Außerdem sind so oder so nur die Applikationen erlaubt, die für eine Klassen- oder Gruppenarbeit benötigt werden und keine für die Verwendung von Sensoren.

4.5.5. Android-Notification-Drawer bzw. Status-Bar

Die Status Leiste (Status Bar) in Android ist eine meist schwarze Leiste oben, die man mit einer Wisch-Finger-Bewegung von oben nach unten öffnet, so dass ein Vollbild-Menü erscheint, den Android-Notification-Drawer, siehe Abbildung 4.4.

Eine deaktivierte Status-Bar bedeutet, dass diese nicht mehr sichtbar ist und deshalb nicht mehr mit oder ohne Absicht als Sicherheitslücke für Leistungsprüfungen ausge-

Algorithmus 4.3 Android-Status-Leiste verstecken [15]

/res/values/themes.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3    <style name="mytheme"
4      parent="@android:style/android:Theme.Holo.Light">
5     <item name="android:windowFullscreen">true</item>
6     <item name="android:windowContentOverlay">@null</item>
7     <item name="android:windowActionBar">false</item>
8     <item name="android:windowNoTitle">true</item>
9    </style>
10 </resources>
```

Verlinkung im Manifest:

```
1  <application android:label="@string/app_name"
2  android:theme="@style/mytheme">
```

nutzt werden kann.

Unter [14] wird beschrieben, wie man dieses verstecken kann. Man braucht das Recht „android.permission.SYSTEM_ALERT_WINDOW" in der Manifestdatei, um mit einer View (Ein Android-App-Bestandteil) alle anderen Apps zu überdecken. Eine andere Methode ist es, wenn die App zu einer Vollbild-App geändert wird.

Unter [15] wird das auch beschrieben. Im Code-Abschnitt 4.3 wird dessen Methode zitiert, wie die Statusleiste durch die Vollbildmethode versteckt wird.

Algorithmus 4.3:
„android:windowFullscreen" (Zeile 5) erklärt das Theme (siehe Nomenklatur) zu einem Vollbild-Theme. „android:windowContentOverlay" (Zeile 6) entfernt den Schatten unter der Action-Bar durch den Wert @null. "android:windowActionBar" (Zeile 7) deaktiviert die Action-Bar. "android:windowNoTitle" (Zeile 8) deaktiviert die Titelleiste des Themes. Im Manifest (Zeile 1-2) bekommt die App diesen Theme.

Die Status-Bar muss deaktiviert oder versteckt werden, weil damit die Einstellungen für das ganze Android-Gerät erreichbar sind. Das ist eine Sicherheitslücke, wenn Leistungstests ausgeführt werden.

Das Ergebnis einer deaktivierten Status-Bar ist es, dass der Tablet-Nutzer keine Statusmeldungen mehr lesen kann, das Gerät über die Einstellungen nicht mehr manipulieren kann, keine Uhrzeit, keinen Akkustand, keine Bildschirmhelligkeit und kein Datum sieht.

4.5.6. Sperren von ausführbaren Dateien (Windows)

Ein zentraler Bestandteil eines Kioskmodus in Windows ist die Beschränkung darauf, welche ausführbaren Dateien erlaubt sind. Unter Android ist dies nicht möglich. Laut [16] „Hooks im Kernel sollen Android sicherer machen" wird das in einer zukünftigen Version von Android nach 4.4. (Kitkat) jedoch möglich sein.

Damit in Windows verhindert werden kann, dass eine ausführbare Datei gestartet wird, oder dies registriert wird, werden so genannte Hooks benötigt. Es ist nötig einen Kernel-Treiber zu programmieren, der im Kernel-Mode statt im User-Mode ausgeführt wird. Das kann zur Folge haben, dass wenn dieser Programmierfehler hat, das ganze Betriebssystem einfrieren kann, statt nur eines Programmes, so dass das System neu gestartet werden muss. Außerdem ist der Treiber im Kernel-Mode kein Prozess, im Gegensatz zu einem Programm im User-Mode. Unter [17] ist beschrieben, wie solch ein Treiber für Windows XP programmiert werden kann. Darin wird empfohlen die Funktion NtCreate-Section() zu verwenden und es kommt Assembler zur Anwendung. So ein Treiber muss für jede Version von Windows extra neu programmiert werden, weswegen sich die Frage stellt diesen Weg einzuschlagen.

Es ist zwar möglich mit Visual Studio und WQL (SQL for WMI) ohne Treiberprogrammierung ein Event zu registrieren, das „feuert", wenn eine Exe-Datei ausgeführt wurde. Das Ausführen lässt sich jedoch nicht verhindern. Es kommt infrage, für die Funktionalität ausführbare Dateien blockieren zu können, Quellcode von Antivirenherstellern zu lesen. ClamWin kommt dafür infrage, weil dessen Code offen vorliegt. Jedoch kann ClamWin keinen Livescan machen. Das heißt, dass deshalb das Quellcode-lesen nicht zum Ziel führt diese Betriebssystemfunktion implementiert vorzufinden.

Es gibt 4 Programme die diese Funktionalität für Windows-Tablets erfüllen:

- Windows 7 Pro ohne Applocker / 8 Enterprise / 8.1 Enterprise mit dem Applocker in den Gruppenrichtlinien (Funktionsanzahl vollkommen ausreichend für einen Kioskmodus)
- Bit9 Parity Suite (Trial Version verfügbar)
- Lumension Application Control
- McAfee Application Control (mindestens 8 GB RAM erforderlich, zusätzliche Programme als Abhängigkeiten die Ressourcen verbrauchen und Geld kosten)

[18]

Unter [19] werden die Produkte miteinander verglichen. SignaCert Enterprise Trust Services unterstützt kein Whitelisting und Blacklisting und kommt deshalb für die Sperrung

von ausführbaren Dateien für den Kioskmodus nicht infrage. Insgesamt haben die Alternativen zum Applocker von Windows 8 / 8.1 Enterprise mehr Funktionen.

Folgende Funktionalitäten werden für den Kioskmodus gefordert, die die genannten Programme und Betriebssysteme erfüllen:

- Sperren von ausführbaren Dateien, nicht nur z.B. *.exe, *.msi, *.bat, sondern alle (welche siehe Gruppenrichtlinien in Windows)
- (Sperren und Erlauben mittels Pfaden), besser: Herstellern von Exe-Dateien über Signaturen und Hashfunktionen von ausführbaren Dateien
- benutzerspezifisches und gruppenspezifisches Sperren
- Active-Directory-Unterstützung und lokale Benutzer

Unter [1] ist beschrieben, wie vorzugehen ist, beim Sperren für Windows 7 Pro ohne Applocker. in dieser Arbeit wird beschrieben, was in Windows 8.1 anders ist, statt alle Seiten aus [1] zu zitieren und anzupassen. Der Windowsordner und der Standard-Programme-Ordner kann für das Ausführen von ausführbaren Dateien zugelassen werden laut [1]. Es ist sicherer Programme durch Identifizierung mit Hashfunktionen zu erlauben, aber bei einem Update der ausführbaren Datei muss neu konfiguriert werden und der Hashwert neu berechnet werden. Der Vorgang regelmäßig Hashwerte zu Programmen zuzuordnen muss automatisch erfolgen. Die Anforderungen für die Automatisierung dieses Problems sind beschrieben in Unterabschnitt 2.3.2. In [1] wird u.a. angedeutet, wie man Bereiche des Dateisystems unsichtbar und „unbetretbar” für den Kioskbenutzer macht, so dass z.B. der Explorer nicht verboten werden muss. Es gibt einen Unterschied in der Konfiguration von Windows 7 und 8.1, der zur Vollständigkeit hier gezeigt wird: In Windows 8.1 lautet ein Unterabschnitt in den Gruppenrichtlinien nicht „Windows Explorer” [20], sondern „Datei-Explorer”. Also um Datenträger zu verstecken, muss in den Gruppenrichtlinien folgendermaßen navigiert werden: Benutzerkonfiguration \ Administrative Vorlagen \ Windows-Komponenten \ Datei-Explorer . Dort gibt es die Auswahl "Diese angegebenen Datenträger im Fenster Arbeitsplatz ausblenden" und "Zugriff auf Laufwerke vom Arbeitsplatz nicht zulassen".
Des Weiteren muss in den Gruppenrichtlinien noch unterbunden werden, dass externe Datenträger angezeigt werden. In [1] werden mehrere relevante Orte in den Gruppenrichtlinien genannt, wo sich auch diese Einstellung dafür auffindet.

Laut [21] und [22] lässt sich mit den Gruppenrichtlinien die Registry bzw. das Verbot von Funktionalitäten von mehreren Rechnern gleichzeitig ändern.

Es gibt die Möglichkeit ausführbare Dateien nicht zu sperren und weiterhin mit Betriebssystemfunktionen zu arbeiteten. Das ist am Ende aufwändiger, als einen Software-Treiber

zu programmieren, und es gibt die Gefahr, dass nicht alle Sicherheitslücken behandelt werden. Z.B. sind Maus- und Touchgesten unter Windows 8.1. in Registryschlüsseln des Herstellers der jeweiligen Eingabegeräte vermerkt, wodurch das Sperrren dieser für jedes einzelne Hardware-Eingabe-Gerät extra eingerichtet werden muss. Wenn ausführbare Dateien nicht gesperrt werden und man dennoch einen Kioskmodus realisieren will, müssen Funktionen und Anzeigemöglichkeiten der Taskleiste reduziert werden. Dazu muss auf Windows-DLLs wie user32.dll zugegriffen werden.
Die Einträge in den Gruppenrichtlinien sind zu einem Teil Registry-Daten. Um herauszufinden, wie groß dieser Teil ist, oder ob gar alles nur Registry-Daten sind, muss jeder dieser Einstellungen in den Gruppenrichtlinien überprüft werden. Es sind Einträge im Bereich von Dutzenden bis Hunderten vorhanden.

Es ist denkbar, dass anstelle Gruppenrichtlinien zu verwenden, ein spezialisiertes Programm programmiert wird, mit ähnlicher Funktionalität. Dieses Programm kann so spezialisiert schneller und bequemer und mit Automatik Tablets von Tablet-Benutzern beschränken für Klassen- und Gruppenarbeiten, etc.

Da davon ausgegangen wird, dass die Gruppenrichtlinien von Windows 8.1 Enterprise ausreichend sind, wird nicht näher darauf eingegangen, und keine angepasste Version mit der gleichen Funktionalität zu spezifiziert. Das sprengt außerdem den Rahmen dieser Arbeit.

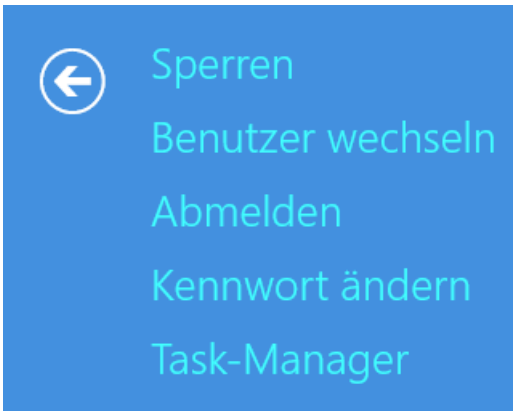
4.5.7. Leeren des Startmenüs (Windows)

In [1] ist beschrieben, wie die Programmeinträge im Startmenü von Windows 7 entfernt werden. In Windows 8.1 sind dazu weitere Schritte nötig. Es werden den Ordnern die Leserechte des Kioskaccounts dafür entzogen. Ggf. wird der Vollzugriff verboten. Um die Rechte zu entziehen, müssen die Ordner im Explorer sichtbar sein. Es müssen also ansonsten ausgeblendete Ordner und Laufwerke angezeigt werden. Der Kiosk-Benutzeraccount nennt sich hier „kiosk”. Beim Ändern der Dateisystem-Rechte muss Rechte-Vererbung aktiviert sein. Die Ordner sind folgende:

- C:\ProgramData\Microsoft\Windows\Start Menu\
- C:\Users\kiosk\AppData\Roaming\Microsoft\Windows\Start Menu\
- C:\Users\kiosk\AppData\Local\Microsoft\Windows\Application Shortcuts\ bzw. Anwendungsverknüpfungen

Der NTFS-Datei-Rechte-Besitzer aller dieser Ordner darf nicht der Kioskuser sein. Denn dann kann dieser Benutzer die Rechte wieder manuell erlangen. Um diese hier gelisteten Ordner herauszufinden, schaut man auf die Eigenschaften einer Programmverknüpfung im Startmenü und sieht in welchem Ordner diese liegt.

Abbildung 4.5.: Alt.-Strg.-Entf.-Menü



Ggf. kann noch das Verzeichnis des Windows-Desktops lese- und schreibgeschützt werden. Dazu zählt derjenige des Benutzers, als auch der öffentliche Ordner, der Dateien enthält, die in der Regel jeder Benutzer lesen und ausführen kann. Der Desktop-Ordner muss einen anderen Besitzer zugewiesen bekommen, als den des ursprünglichen Benutzers des Kioskmodus. Das ist nötig, damit der Kiosk-Benutzer nicht wieder Rechte des Ordners erlangen kann.

4.5.8. Registry-Manipulation, um das Alt-Strg-Entf-Menü zu leeren

Das Alt-Strg-Entf-Menü stellt ein Sicherheitsproblem dar, weil damit z.B. der Taskmanager geladen werden kann und damit beliebige Programme gestartet und beendet werden können.

Diese Einträge in der Registry müssen getätigt werden, damit nicht alle Menüeinträge vorhanden sind. Alle Werte in der Registry dazu sind vom Datentyp DWORD und ihr Variableninhalt muss auf 1 gesetzt werden.

Windows 8.1:
Unter HKEY_CURRENT_USER \ Software \ Microsoft \ Windows \ CurrentVersion \ Policies \ System
müssen die DWORD-Einträge

- DisableTaskMgr
- DisableLockWorkstation
- DisableChangePassword

auf 1 gesetzt werden. Das betrifft dann den eingeloggten Benutzer und ist ohne Administrator-Rechte möglich.

Unter: HKEY_CURRENT_USER \ Software \ Microsoft \ Windows \ CurrentVersion \ Policies \ Explorer
muss der DWORD Wert NoLogoff auf 1 gesetzt werden. (ohne Adminrechte möglich)

Unter: HKEY_LOCAL_MACHINE \ Software \ Microsoft \ Windows \ CurrentVersion \ Policies \ System
muss der DWORD-Wert HideFastUserSwitching auf 1 gesetzt werden. Dies ist nur mit Adminrechten möglich und betrifft alle angemeldeten Benutzer. Unter Windows 8 findet sich der Wert in HKEY_CURRENT_USER \ Software \ Microsoft \ Windows \ CurrentVersion \ Policies \ System und betrifft nur den aktuell eingeloggten Benutzer, der den Wert ändert. D.h. dieser Wert ist auch ohne Administrator-Rechte in Windows 8 änderbar.

4.5.9. Kamera (Android)

Unter [23] ist beschrieben, mit welcher Methode (Camera.lock()), die Kamera davon abgehalten werden kann von anderen Prozessen angesteuert zu werden. Mit „Camera.open()!=null" lässt sich herausfinden, ob die Kamera läuft oder nicht.

4.5.10. Bluetooth und sonstige Geräte unter Windows

Im letzten Text-Abschnitt von Unterabschnitt 4.5.3 wird begründet, warum Bluetooth deaktiviert werden muss. Für sonstige Geräte gilt das gleiche.

Insgesamt ist für das Deaktivieren von Bluetooth unter Windows ein Quelltext nötig, der den von Android für den gleichen Zweck von der Zeilenanzahl um ein Vielfaches übersteigt, und der komplexer ist, als der von Android.

Wenn das Programm, das Bluetooth verwaltet keine Rechte der Ausführbarkeit besitzt, z.B. mittels AppLocker, dann kann der Tablet-User mit dem Kiosk-Account kein Bluetooth verwenden. Das genügt! Im Autostart-Ordner oder in der Registry für Autostart-Programme ist es nicht auffindbar, obwohl es gestartet wird nach dem Einloggen.

Das Aktivieren und Deaktivieren von Geräten durch Programmierung unter Windows sprengt den Rahmen dieser Arbeit. Unter [24] ist ein Ansatz beschrieben, wie Geräte aktiviert und deaktiviert werden können.

4.6. Tasten sperren unter Windows 8.1 und Android

4.6.1. Android

Unter [25] ist die offizielle Dokumentation zur Programmierung von Android von Google zu finden. Darin befindet sich eine Beschreibung welche Klassenmethoden für das

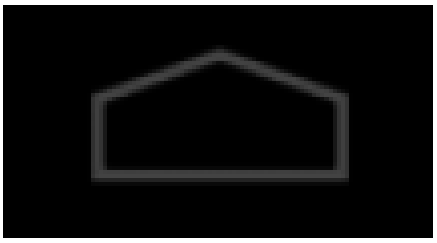
Registrieren von Tasten innerhalb einer App verwendet werden können, mit möglicher Reaktionsänderung auf eine Taste:

- public boolean onKeyMultiple (int keyCode, int count, KeyEvent event)
- public boolean dispatchKeyEvent (KeyEvent event)
- public boolean onKeyUp(int keyCode, KeyEvent event)
- public boolean onKeyDown(int keyCode, KeyEvent event)
- public boolean onKeyLongPress(int keyCode, KeyEvent event)

Diese Methoden müssen in einer Activity (siehe Nomenklatur) verwendet werden und die geerbten gleichnamigen Methoden müssen vor ihrem Methodenkopf @Override davor stehen haben, was für „überschrieben" steht. In den Methoden-Körpern lässt sich abfragen, welche der Tasten gedrückt wurde z.B. mittels
„if (keyCode == KeyEvent.KEYCODE_BACK)", ob die Rücktaste betätigt wurde. Die Methoden werden also bei einem zugehörigen Ereignis automatisch gestartet, wie z.B. dem Loslassen einer Taste bei Verwendung der Methode onKeyUp. Jedoch werden nur Tastenanschläge berücksichtigt, die gedrückt wurden, wenn die App im Vordergrund läuft, wenn die App eine dieser 5 obigen Methoden verwendet. Key-Hooks für globale Tastenanschläge gibt es für Android nicht. Außerdem lässt sich weder abfragen, ob die Recents-Taste betätigt wurde, noch ist es möglich zu verhindern, dass die Funktionalität dieser Taste ausgeführt wird. Die Recentstaste ist diese, die den android-eigenen Taskmanager startet, um zwischen Apps zu wechseln oder diese terminieren zu lassen, indem ein Finger auf die App gelegt wird und dieser zur Seite geschoben wird („wischen"). Dieser Taskmanager bzw. dieses Recentsmenü ist in anderen Geräten erreichbar durch anhaltendes betätigen der Hometaste. Die Hometaste führt, durch einmaliges auslösen quasi, auf den Desktop bzw. den Homescreen (so nennt sich das in Android) wo die individuell abgelegten Icons der Apps und die aktiven Widgets liegen. Die Wirkung der Home-Taste lässt sich durch die Tasten-Ereignisse nicht verändern, aber das ist kein Problem, wenn eine andere App zur Home-App erklärt wird. Wenn am Ende einer der 5 obigen Methoden statt „return super.jeweiligeUebergeordneteMethode()" z.B. „return super.onKeyUp(keyCode,event)" etwas anderes steht, z.B. „return true", dann wird verhindert, dass die Funktionalität ausgeführt wird, die das Betriebssystem normalerweise bereitsstellt. Mit anderen Worten, wird dann die Funktion der Taste blockiert, wenn die Taste in der App gewählt wurde, in der man dieses „return true" programmiert hat. In diesem Beispiel handelt es sich um das Loslassen einer Taste.

Neuere Android-Versionen verwenden Softwaretasten und teilweise und nicht immer, oder andere Tasten, als die Hardwaretasten älterer Versionen. Jedoch ist das von Hersteller zu Hersteller unterschiedlich. Insofern sind alle Varianten zu berücksichtigen, wenn man Software für Android entwicklen will. In manchen Android-Versionen, auch bis 4.4,

Abbildung 4.6.: Home-Taste als Bestandteil im Display



gibt es kein Recents-Menü, weder durch langes Drücken auf Home noch durch eine nicht existierende Recents-Taste, ggf. mit anderen Tastenkombinationen.

4.6.2. Die Hometaste (Android)

Die Hometaste (Bild 4.6) kann auf eine andere Art in ihrer Wirkung verändert werden, als oben in Unterabschnitt 4.6.1 beschrieben ist. Der Homescreen wird durch die Hometaste aufgerufen. Es gibt Launcher-Apps, die diesen als Startzentrale für Apps ersetzen können, oder als Kiosk-Modus-Apps, die es gibt, damit nicht alle Apps gestartet werden können. Der Homescreen selbst ist „nur" eine App. Unter Windows nennt sich der Homescreen „Desktop". Unter Android ist der Homescreen austauschbar.

Die Überwachungs-App dieser Arbeit muss, mittels ihres Quellcodes und der Bedienung dieser, zum Homescreen erklärt werden, damit der Tablet-Benutzer nicht in Versuchung gerät, Apps zu starten, die als Betrugsversuch gezählt werden. Als Benutzer erscheint dann eine Frage, welche App der Homescreen werden soll, ggf. mit 2 Radio-Buttons, ob für immer oder nur einmalig.

Wenn die Überwachungs-App dann der neue Homescreen ist, dann wird diese in den Vordergrund geholt, sobald die Hometaste gedrückt wurde. Das bleibt dauerhaft so, wenn bei der Auswahl des Homescreens „immer" ausgewählt wurde, statt einmalig.

Der Homescreen ist praktisch die App, die als erstes nach dem Booten von Android geladen wird. Jede App kann als Homescreen ausgewählt werden, wenn in Ihrer Manifestdatei das steht, was in Quelltext 4.4 zu lesen ist.

Die Auswahl des Homescreens muss dabei vom Tablet-Nutzer durchgeführt werden, weil das unter Android nicht anders möglich ist. Initiiert werden kann diese Auswahl aus der Ferne über das Netzwerk, z.B. von Seiten des Computers der Aufsichtsperson / des Lehrers.

Dabei ist "android.intent.category.HOME" (Zeile 12) verantwortlich dafür, dass das Androidsystem weiß, dass die App als Homescreen zur Verfügung steht, so dass es bei Gelegenheit fragt, ob man die App als Homescreen verwenden will. Ansonsten lässt sich

Algorithmus 4.4 AndroidManifest.xml

```
1  <?xml ... ?>
2    <manifest .... />
3    <application ..... >
4      <activity    ..... >
5        <intent-filter>
6          <action android:name="android.intent.action.MAIN" />
7          <category
8            android:name="android.intent.category.LAUNCHER" />
9          <category
10           android:name="android.intent.category.DEFAULT" />
11         <category
12           android:name="android.intent.category.HOME" />
13         </intent-filter>
14       </activity>
15     </application>
16   </manifest>
```

dies auch durch explizites Festlegen in den Androideinstellungen ermöglichen , die von Hersteller zu Hersteller anders strukturiert sind. Im Standard-Android 4.4 ist diese Einstellung in den Settings zu finden unter: Device / Home.

<intent-filter> ... </intent-filter> legt fest, auf welche Intents (Intents siehe Nomenklatur) die App eine Reaktion zeigen soll. Ohne so eine Angabe wird eine programmierte Intent-Reaktion nicht reagieren. In einem Teil der Fälle muss zusätzlich die Funktionalität in Java programmiert werden, die beim Ankommen eines Intents geschehen soll. Mit “<action ...” wird eine Kategorie gewählt, wie auf Intents reagiert wird. Mit “<category ...” wird der vordefinierte erlaubte Intent zugelassen, auf den die App reagieren können soll. Ausführliche Dokumentation zu Intents finden sich im Netz.[26] ist ein Buch zum Thema Intents.

“android.intent.category.DEFAULT” (Zeile 10) ist die Standardkategorie für eine App. Damit wird die betreffende Activity zur Standard-Activity erklärt, was bedeutet, dass sie im Zweifel, welche Activity zu wählen ist, für eine Aktion, diese Activity verwendet wird, weshalb die Kategorie Default heißt.
"android.intent.category.LAUNCHER" (Zeile 8) bedeutet, dass die Activity (Activity siehe Nomenklatur) die Initial-Activity ist, mit der die App startet. "android.intent.action.MAIN" (Zeile 6) bedeutet, dass die betreffende Activity der Haupteingangspunkt ist, wobei keine Daten empfangen werden müssen.

Unter [27] ist beschrieben, wie es mittels eines Workarounds erreichbar ist, dass, beim Betätigen der Hometaste, nicht nur die Frage kommt, welche App zum Homescreen erklärt wird, sondern dass auswählbar ist, dass dies nicht nur einmalig gilt. D.h. die App

ist dauerhaft der Homescreen. Die Simulation des Tastendrucks auf Home ist unter [28] erklärt. Wenn nun in einem Menü ausgewählt wird, welcher der nächste Homescreen sein wird, dann müssen die XML-Dateien, wie unter [27] beschrieben, geändert werden und der Java-Code von [27] bei Menüauswahl muss ausgeführt werden. Darauf folgt schließlich die Simulation des Drückens auf die Hometaste, beschrieben unter [28]. Wurde vom Benutzer ausgewählt, dass die neue Home-App für immer die alte ersetzt, und dann noch ein Mal die Hometaste betätigt, dann erscheint die neue Home-App. Ansonsten wird ein weiteres Mal abgefragt, welche die neue Home-App sein soll, von denen die einen entsprechenden Eintrag im Manifest haben, dass sie zum neuen Homescreen werden können.

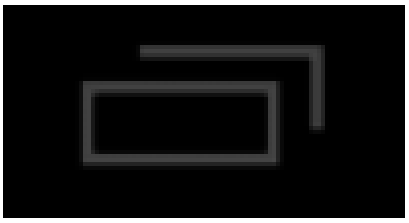
4.6.3. Tasten sperren in Windows 8.1

Eine Übersicht aller Tastenkombinationen unter Windows findet sich hier: [29].

Unter [30] findet sich eine Beschreibung, wie Tastenanschläge global mit C# registriert werden und in ihrer Wirkung verändert werden können. Bei dieser Methode spielt es keine Rolle, welche Anwendung den Focus hat (im Gegensatz zu Android und Windows RT), d.h. welche Anwendung gerade die aktive Anwendung ist und entsprechend in der obigen Leiste gewöhnlich eine andere Farbe hat als die anderen Fenster mit gleichfarbiger Titelleiste. Dabei ist es möglich, dass die Tastenanschläge nicht an Windows oder andere Programme weitergeleitet werden, d.h. andere Tastenkürzel anderer Programme können quasi überschrieben werden oder windows-eigene Tastenkombinationen funktionieren nicht mehr, wenn man das so beabsichtigt hat. Betriebssysteme bieten in der Regel Hooks an, zu Deutsch “Haken”. Diese sind in der Regel mit dem Observer-Entwurfsmuster programmiert, wie z.B. auch Addons und Plugins. Das Observer-Pattern gehört zu den Standard-Entwurfsmustern des Informatikstudiums. Diese Hooks ermöglichen es zu verschiedenen Ereignissen individuell zu reagieren oder vorgegebene Reaktionen zu Ereignissen vom Betriebssystem zu blockieren. Man kann diese Vergleichen mit Events und Event-Handlern von z.B. Java.

Unter Windows 7 bis 8.1 lässt sich jedoch nicht die Tastenkombination Alternate(wechseln) + Steuerung + Entfernen bzw. Alt + Strg. + Entf. in ihrer Reaktion von Windows blockieren. Wenn das geschriebene Programm alle Tasten blockt, funktioniert letztgenannte Tastenkombination trotzdem. Das Problem dabei ist, dass z.B. damit der Taskmanager gestartet werden kann, der das Ausführen und Beenden von beliebigen Programmen ermöglicht und deshalb ein Sicherheitsrisiko für einen Kioskmodus darstellt. Unter Windows 8 und 8.1 lässt sich das Menü, in dem u.a. der Taskmanager auswählbar ist, das bei dieser beschriebenen Tastenkombination erscheint, verändern. Mit Registryeinträgen ist das möglich. Teilweise werden dazu Administratorrechte benötigt, aber nur für den Registryeintrag, um das Wechseln des Benutzers zu einem anderen Account zu unterbinden.

Abbildung 4.7.: Recents-Taste als Bestandteil im Display



Um den Aufruf des Taskmanagers zu verhindern, sind keine Administratorrechte notwendig, wenn die Registry manuell oder mittels eines Programms entsprechend bearbeitet wird.

4.6.4. Blockieren der Recents-Taste unter Android

Wie bereits beschrieben, ist die Recentstaste in ihrer Wirkung nicht blockierbar. Wenn die App, die für den android-eigenen Recents-Taskmanager mit dem Kill-Kommando beendet wird, funktioniert sie trotzdem. Entweder unterbindet das System das Beenden mit Kill oder die App wird automatisch wieder neu gestartet. Der Quellcode von Android ist offen verfügbar. In dem betreffenden Code des Recents-Taskmanagers sind einige typisierte Eingänge für Interprozessnachrichten vorhanden (Intents). Dabei gibt es Unterschiede zwischen den Versionen von Android. Des Weiteren ist nur ein Teil dieser Intents (Intents siehe Nomenklatur) in der XML-Manifestdatei, die jedes Androidprogramm haben muss, registriert , so dass auch nur dieser Teil extern von einer anderen App Daten und Anweisungen empfangen kann. Alle anderen Intentemfpänger sind nur innerhalb der Recents-App erreichbar. Es gibt die Möglichkeit, das Recentsmenü mittels Intents nach seinem Entstehen wieder zu schließen[31] . Das genügt nicht.

Ansonsten gibt es die Möglichkeit, die zum Recentsmenü zugehörige System-App mit dem Kill-Kommando zu beenden, aber dazu werden Administratorrechte benötigt. Ggf. muss verhindert werden, dass die Recentsapp neu gestartet wird, wenn sie beendet wird mit Kill. Dazu kann ein zwischenzeitliches Umbenennen dieser App in der Zeit unterhalb einer Sekunde auf dem Festspeicher helfen. Es gibt noch die Möglichkeit direkt den Androidquellcode von Google zu ändern und zu compilieren. Das ist dann eine Modifikation, wie z.B. der Cyanogenmod. Das hat den Nachteil dass man dieses Android auf die jeweiligen Geräte portieren muss.

4.6.5. Gesten, Shortcuts und programmierbare Tasten in Windows 8.1

In diesen Quellen sind die Tastenkombinationen von Windows 8.1 / 8 beschrieben:

- wichtige: [32]
- alle: [33]

Gesten werden hier erklärt:

- [34]

Weitere Eingabemethoden werden hier erklärt:

- [35]

Hier sei aber erwähnt, dass wenn Tablet-Nutzer sich neu einloggen müssen, mit einem Kiosk-Account, mit beschränkten Rechten, das Problem dieses Risikos gelöst ist, dass mit Eingabemethoden unerlaubt Informationen beschaffen werden können. Das liegt daran, weil damit vorher gespeicherte Daten in der Zwischenablage, etc. nicht mehr verfügbar sind. Vorher gestartete Programme sind beim Neu-Einloggen nicht mehr offen.

Um Gesten global zu deaktivieren, ist es abhängig vom Eingabe-Gerät-Produkt nötig, dies für jedes Hardwareprodukt einzeln tun zu müssen. So sind z.B. die Registryeinträge für Gesten des Synaptic-Touchpad, von einem Dritthersteller, an dem jeweiligen Ort in der Registry, den die Firma ausgesucht hat. Es gibt Tastaturen mit programmierbaren Tasten. Diese Tasten lassen sich je nach Hersteller an einem anderen Ort abstellen (Orte können sein GUI-Konfigurationsfelder / Dateien / Registry-Einträge etc.). Insofern macht es Sinn, dass ein Tablet-Benutzer sich in den Kiosk-Benutzer-Account neu anmelden muss.

Bildungseinrichtungen, wie Schulen, dürfen also keine Tastaturen einsetzen mit programmierbaren Tasten, wenn Klassenarbeiten mit den Tablets durchgeführt werden.

Zunächst ist es für einen Kioskmodus wünschenswert, wenn Gesten, Shortcuts und programmierbare Tasten keine Gefahr zum Umgehen des Kioskmodus werden können. In der Spezifikation des Schul-Kiosk-Modus ist die Gefahr hingegen niedrig, da das Ausführen von Exe-Dateien usw. beschränkt wird, und damit Gesten, Shortcuts und programmierbare Tasten in ihrer Wirkung so beschränkt werden können, dass Benutzer keine Rechte erlangen können und an keine Daten kommen, die sie nicht haben dürfen.
Es gibt das Modell der schichten-basierten Sicherheit, in dem nicht nur eine Sicherheitsschicht vorliegt für höhere Sicherheit, als wenn es nur eine Sicherheitsschicht gibt. Deshalb ist die Betrachtung dieses Abschnitts relevant.

Es wurde in diesem Unterabschnitt 4.6.5 festgestellt, dass die Sperrung von Gesten, Shortcuts und programmierbaren Tasten als Sicherheitsschicht nicht vollständig möglich ist, wenn dies für alle existierenden Eingabegeräte eingestellt werden muss. Sind die Eingabegeräte der Tablets bekannt, dann können speziell für diese die möglichen Gesten, Shortcuts und programmierbare Tasten mit einer Programmierung deaktiviert werden.
Wegen der Sperrung von ausführbaren Dateien müssen Gesten, Shortcuts und programmierbare Tasten nicht zwingend deaktiviert werden. Das bietet jedoch zusätzlichen Schutz.

4.7. Beobachten statt Einschränken (statt Kioskmodus)

Zu Windows wird in dieser Arbeit beschrieben, wie ein Kioskmodus realisiert wird, um das allgemeine Sicherheitsproblem dieser Arbeit zu lösen. Es ist jedoch ausreichend den Pool-Nutzer insofern zu überwachen, dass das Ausnutzen von Sicherheitslücken der Aufsichtsperson gemeldet wird. Für Android wird daher die andere Methode beschrieben, wie Betrugsversuche erkannt werden können, statt das Ausnutzen von Sicherheitslücken mit einem Kioskmodus zu verhindern.

4.7.1. aktive Prozesse und Dienste auflisten in Android

Es ist eine Sicherheitslücke, wenn verbotene Programme geöffnet werden können. Ein Verletzen dieses Verbots muss dem zuständigen Lehrercomputer über das Netzwerk gemeldet werden, damit in Klassenarbeiten faire Bedingungen herrschen. Ein Betrugsversuch kann auch vorliegen, wenn ein Programmbestandteil eines Programms geöffnet wird. Jedoch sprengt die Behandlung dieses Themas diese Arbeit und es ist einfacher direkt nur Programme zu erlauben und zu verbieten. Damit der Tablet-Benutzer nicht ohne Absicht, z.B. wegen der Gewohnheit, nicht zugelassene Apps startet, muss es einen extra Launcher geben, in dem nur erlaubte Programme gestartet werden können, so dass es, mit dem Verlassen des Launchers, umständlich wird verbotene Apps zu starten. Über eine Netzwerk-Verbindung muss alle 5 Sekunden dem Rechner der Aufsichtsperson gesendet werden, welche App gerade aktiv ist, also diejenige die der Tablet-User sieht. Ggf. wird auch gesendet, welche anderen Apps und Dienste insgesamt gestartet sind.

In [36] wird erklärt, wie abgefragt wird, welche App gerade im Vordergrund ist, die der Tablet-Benutzer in dem Augenblick sehen kann. Im Algorithmus 4.5 wurde der Code dazu insoweit vom Autor der Arbeit geändert, dass alle App-Namen, von Apps die gestartet sind, in einer Schleife ausgegeben werden, außerdem alle aktiven Dienste und Android-Tasks.
Das ist hilfreich, wenn programmiertechnisch gemeldet werden muss, welche Apps / Dienste / Tasks laufen und welche die aktive App ist.

Android-Tasks sind etwas anderes als Windows Tasks. Sie beinhalten Activitys (Activity siehe Nomenclature) aus Apps und arbeiten mit dem Stapel aktiver Apps, siehe: [37] .
Um auf Tasks zugreifen zu können, wird das Recht
android.permission.GET_TASKS benötigt, das in der Manifestdatei anzugeben ist.

Erklärung zu Algorithmus 4.5:

In Zeile 1-2 wird der ActivityManager besorgt, der dazu da ist mit allen laufenden Activitys von Android zu interagieren. In Zeile 3-4 werden aus diesem die aktuell laufenden

Algorithmus 4.5 Auflisten der aktiven Prozesse und Dienste

```
1  ActivityManager am = (ActivityManager)
2  getApplicationContext().getSystemService( Context.ACTIVITY_SERVICE);
3  List<RunningAppProcessInfo> RunningApps =
4  am.getRunningAppProcesses();
5  List<RunningServiceInfo> runningServices =am.getRunningServices(1);
6  List<RunningTaskInfo> runningTasks=am.getRunningTasks(1);
   //List<RunningTaskInfo> rtis = am.getRunningTasks(1);
7  for (RunningAppProcessInfo ra : RunningApps) {
8    Log.i("RunningApps", ra.processName);
9  }
10 for (RunningServiceInfo rs : runningServices) {
11   Log.i("runningServices", rs.process);
12 }
13 for (RunningTaskInfo rt : runningTasks) {
14   Log.i("runningTasks", rt.toString());
15 }
```

Programme in einem Objekt verlinkt, in Zeile 5 die Dienste und in Zeile 6 die Android-Tasks, die in [38] näher erklärt werden. Der Parameter 1, der 3 mal vorkommt, bestimmt die Maximalanzahl, die in die jeweilige Liste eingefügt wird. Nummer 1 der Apps ist die aktuell angezeigte App. Tasks bestehen aus Activitys. Activitys sind Bestandteile von Apps. Mit den 3 For-Schleifen ab Zeile 8 werden alle Apps, Dienste und Tasks ausgegeben, die in den jeweiligen Listen zuvor gespeichert wurden.

Erklärung zu Algorithmus 4.6:

In Zeile 10 wird die Activity besorgt, die auf dem Gerät für den Benutzer aktuell sichtbar ist.
In Zeile 12-13 wird untersucht, ob die Activity vom Namen her die gleiche ist, wie diejenige von vor 5 Sekunden. Die 5 Sekunden stehen in Zeile 30 (sleep(5000)).

4.7.2. extra Launcher für Android

In [39] wird beschrieben, wie man die installierten Apps mittels Quellcode gelistet bekommt.

In [40] wird beschrieben, wie man eine App aus einer anderen App starten lässt. Das ist relevant, wenn aus dem Launcher andere Apps gestartet werden.

Damit der neue programmierte Launcher den alten Standardlauncher ersetzt, muss dieser zum Homescreen gemacht werden. Dazu muss im Manifest das Recht

Algorithmus 4.6 Senden der aktuell sichtbar-aktiven App

```
1  @SuppressWarnings("hiding")
2  class OneShotTask implements Runnable {
3    public ActivityManager am;
4    List<RunningTaskInfo> tasks;
5    ComponentName LasttopActivity=null;
6
7    private boolean AppInForegroundChanged() {
8      tasks = am.getRunningTasks(1);
9      if (!tasks.isEmpty()) {
10       ComponentName topActivity = tasks.get(0).topActivity;
11       if (LasttopActivity!=null) {
12         if (!topActivity.getPackageName().equals(
13           LasttopActivity.getPackageName())) {
14           return true;
15         } else {
16           LasttopActivity=topActivity;
17         }
18       } else {
19         LasttopActivity=topActivity;
20       }
21     }
22     return false;
23   }
24   @Override
25   public void run() {
26     am = (ActivityManager) getSystemService(
27     Context.ACTIVITY_SERVICE);
28     while (true) {
29       try {
30         Thread.sleep(5000);
31       } catch (InterruptedException e) {
32         e.printStackTrace();
33       }
34       AppInForegroundChanged();
35     }
36   }
37 }
38 Thread t1 = new Thread( new OneShotTask() );
39 t1.start();
```

Algorithmus 4.7 Starten einer App aus einer anderen, ein Beispiel

```
1 String app = "com.android.settings";
2 PackageManager manager = getPackageManager();
3 Intent i = manager.getLaunchIntentForPackage(app);
4 if (i == null)
5   throw new PackageManager.NameNotFoundException();
6 i.addCategory(Intent.CATEGORY_LAUNCHER);
7 startActivity(i);
```

„android.intent.category.HOME" angefordert werden, womit Android erkennt, dass die App als potentieller Homescreen-Ersatz fungieren kann.
„android.intent.category.DEFAULT" spielt nur eine Rolle, wenn die App per Intent aufgerufen wird, ohne explizite Angabe einer Kategorie.

Im Unterabschnitt 4.6.2 wird am Ende beschrieben, wie jede App zu einem Homescreen werden kann. Das ist nötig, damit Tablet-Benutzer nicht den Standardlauncher nehmen, mit dem sie ohne Absicht Apps starten können, die als Betrugsversuch zählen, was dem Lehrercomputer gemeldet wird.

Es wurde vom Autor dieser Arbeit programmiertechnisch überprüft, ob das alles in den Links funktioniert, Nachweise siehe Anhang A.1 und A.2, in denen gezeigt und beschrieben wird, was die Ausgaben sind. Es ist aber anzumerken, dass einzelne Antworten auf den Seiten der Quellen bewertet werden, so dass dies ein Hinweis ist, ob die Lösung funktioniert.

Dem Lehrer muss nicht gemeldet werden, ob der Launcher durch den Tablet-Benutzer wirklich gewechselt wurde, allerdings würde die Information darüber nicht schaden. Für den Tablet-Nutzer ist es mit dem neuen Launcher nur einfacher, nicht ohne Absicht eine App zu starten, die im Moment der Klausur nicht erlaubt ist.

In Zeile 1 des Algorithmus 4.7 steht in Klammern der Name des Pakets das zur App gehört, wodurch Android weiß, welche App gestartet wird. Mit „startActivity" in Zeile 7 wird die App der Android-Einstellungen gestartet. Welche App das ist wurde in Zeile 3 übergeben.

4.8. Geo-Lokalisierung

Wenn Lehrer oder Schüler den Klassenraum betreten, und dies technisch von einer Software erkannt wird, dass Geräte, wie z.B. Tablets den Ort gewechselt haben, dann können damit programmatisch computergestützte Ereignisse ausgelöst werden: Schutzmechanismen, ein Kioskmodus, die Lockerung von Schutzmechanismen, Aktivierungen /

Deaktivierungen von Überwachung wegen Klassenarbeiten können damit in Verbindung stehen.

Es gibt mehrere Methoden zur Lokalisierung von Computern auf unserem Planeten Erde: über GPS, IP-Adressen, Datenbanken, Kombinationen dieser Methoden.

4.8.1. Verwendung von Geo-Lokalisierung

Geo-Lokalisierung kann verwendet werden, wenn Tablet-Nutzer in die Nähe das Klassenraumes mit dem Tablet kommen oder auch wenn das Tablet im Klassenraum gestartet wird, um dort den Port für den Überwachungs-Client des Lehrers zu öffnen. Auf diese Weise ist dieser offene Port außerhalb des Klassenraumes als geschlossener kein Sicherheitsrisiko für die Tablet-User.

Eine weitere Möglichkeit ist es, dass das Tablet automatisch abfragt, was denn der Status in dem Klassenraum ist, den der Lehrer schon festlegen kann, bevor die Tablet-Benutzer den Raum betreten. Ein Status kann sein: Normaler Unterricht, Gruppenarbeit, Klassenarbeit, Testat, Übung. Für jede dieser Möglichkeiten können Profile existieren, die den Schülern entsprechende Rechte und Verbote zuweisen.

Der Lehrer kann, neben dem vorherigen Festlegen wie oben beschrieben, auch Zustände planen, wenn Tablet-User regelmäßig zu Zeitpunkten die gleichen Tätigkeiten ausführen, in den gleichen Räumlichkeiten.

Für Sicherheitsfeatures, wie Überwachung oder dem Einschränken von Rechten, ist Geo-Lokalisierung bedingt geeignet, weil z.B. das Tablet ummantelt werden kann, oder der Tablet-Benutzer schaltet das Netzwerk oder GPS aus, oder es gibt unvorhergesehene technische Defekte.

Sobald ein Defekt der Geo-Lokalisierung eines Tablets vorliegt, und deshalb ein Tablet-Nutzer nicht überwacht werden kann, kann dieser zumindest mit diesem Gerät nicht an der Klassenarbeit mitschreiben. U.a. deshalb muss es weitere Tablets im Vorrat geben.

4.8.2. GPS mit Android

Für Windows lag für diese Arbeit kein Gerät vor, mit dem GPS angsteuert werden kann, aber für Android.

Der Code stammt aus verschiedenen Tutorials und wurde vom Autor dieser Arbeit in einem Prototypen umgesetzt und dann auf einem Android-Gerät als Programm in seiner Funktionalität überprüft. Die Ausgabe auf dem Android-Gerät ist in Abbildung 4.9 sichtbar. Darin ist die erste Zahl die Latitude und die zweite die Longitude von dem Ort, an

Abbildung 4.8.: GPS an / aus im Android-Notification-Drawer

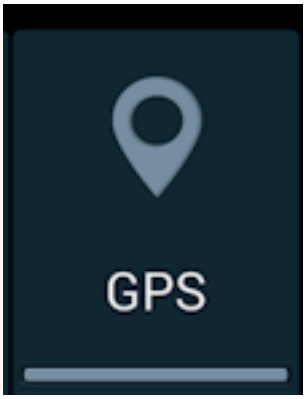
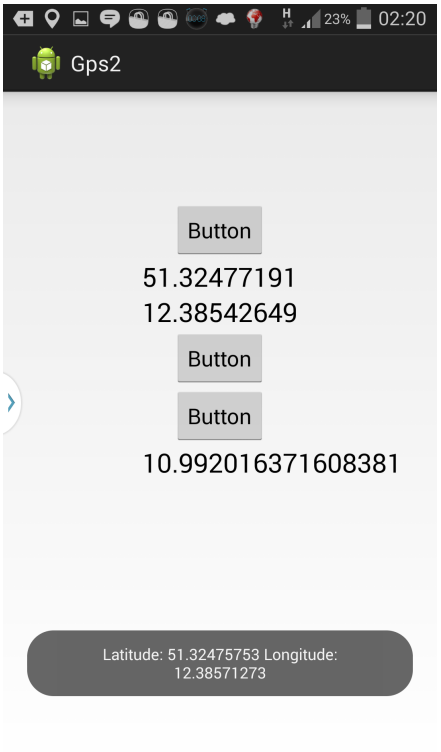


Abbildung 4.9.: GPS-Prototyp-Bildschirmausgabe



Algorithmus 4.8 LocationManager

```
1 LocationManager locationManager =
2 (LocationManager) getSystemService(Context.LOCATION_SERVICE);
```

Algorithmus 4.9 Rechte zum Lokalisieren in der Manifestdatei

```
1 <uses-permission
2 android:name="android.permission.ACCESS_FINE_LOCATION" />
3 <uses-permission
4 android:name="android.permission.INTERNET" />
```

dem mit GPS gemessen wurde. Die dritte Zahl ist das Ergebnis einer Messung von einer Distanz in Metern von den ersten beiden Werten bis zu den beiden Werten unten auf dem Bild im abgerundeten Rechteck. Diese letzten beiden Werte entsprechen dem zweiten Messpunkt. Die Distanzbestimmung in Metern behandelt Unterabschnitt 4.8.5.

In einer von der Klasse „Activity" (Begriff siehe Nomenklatur) abgeleiten Instanz bekommt man mit dem Algorithmus 4.8 Zeile 1-2 das Objekt, in dem die Geo-Lokalisierungsdaten verwaltet werden. Die Activityklasse muss die Java-Schnittstelle LocationListener implementieren.

Die App muss die Rechte bekommen, um Lokalisierung durchführen zu dürfen.

Im Algorithmus 4.10 in Zeile 7-8 werden die Koordinaten Latitude und Longitude besorgt. Die geerbt-überschreibende Methode onLocationChanged wird immer dann automatisch aufgerufen, wenn das Androidsystem eine neue automatische GPS-Messung durchführt.

Des Weiteren gibt es noch andere Tutorials im Internet, die das gleiche Problem auf unterschiedliche aber ähnliche Art lösen mit teilweise anderen Befehlen der teilweise gleichen Klassen.

Algorithmus 4.10 Koordinaten abrufen

```
1 doulbe latitude;
2 double longitude;
3
4 @Override
5 public void onLocationChanged(
6 Location location) {
7   latitude = location.getLatitude();
8   longitude = location.getLongitude();
9 }
```

Das Problem an GPS ist, dass es zwar unter freiem Himmel funktioniert, es aber unter Überdachung u.a. davon abhängt, wie dick die Überdachung ist, und welcher Winkel am Fenster zu den Satelliten vorliegt. Deshalb hat es einen Vorteil in Abschnitt 4.8.3 und 4.8.4 beschriebene alternative Möglichkeiten zum Bestimmen des Standortes zu verwenden. Dass GPS bei Überdachung bedingt funktioniert, ist von Bedeutung, weil die Lokalisation für automatische Reaktionen in Anti-Cheat-Plus verwendet werden kann.

4.8.3. IP-Geo-Lokalisierung

Über die IP-Adresse ist der Internetanbieter / ISP / Provider lokalisierbar. Weiterhin können verschiedene Informationen über den Internetanbieter abgerufen werden, wozu auch die Koordinaten auf der Erde gehören. Es gibt verschiedene Anbieter, die all diese Informationen zur Verfügung stellen.

Hier [41] wird erklärt, wie man mit Java IP-Geo-Lokalisierung umsetzt. Dabei wird die GeoLite-Datenbank verwendet.

4.8.4. Lokalisierung über Daten von Datenbanken und Kombinationen

Ohne GPS kann ggf. die Ortung über verschiedene Daten in Datenbanken erfolgen. Dazu gibt es die W3C Geolocation API. Es werden die Daten kombiniert und daraus Schlussfolgerungen gezogen. Dazu gehört: IP-Adresse, MAC-Adresse, WLAN-Hotspots, Bluetooth-MAC, Radiofrequenz, auch GPS (jetzt oder früher), die GSM / CDMA Zelle im Gebiet. Entsprechende Webseiten können Lokalisationen anfragen, so dass der Benutzer im Browser bestätigen kann, ob die Webseite die Daten zur Geo-Lokalisierung bekommen darf. Quelle: [41]

In [42] ist eine Java-Programmbibliothek, mit der Lokalisierungen umgesetzt werden können.

In [25] wird erklärt, wie mit Javascript der Standort herausgefunden werden kann.

4.8.5. Distanzen in Meter umwandeln, aus Erdkoordinaten

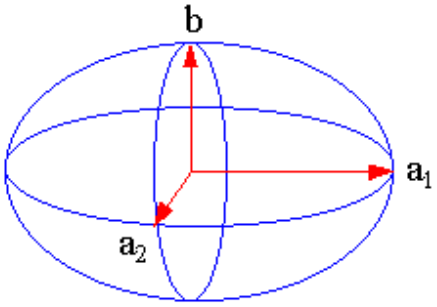
Distanzen-Berechnung wird benötigt, damit Schul-Räumlichkeiten lokalisiert werden können.

Um beliebige Distanzen aus Deltas von Latitude und Longitude (Erd-Koordinaten) zu berechnen, ist es sinnvoll auf Programmbibliotheken [43] zurückzugreifen.

Die Java-Quellcode-Dateien der Bibliothek[3] [43], kann man in sein Projekt ziehen, an-

[3] http://www.gavaghan.org/blog/free-source-code/geodesy-library-vincentys-formula-java/

Abbildung 4.10.: ein Ellipsoid



Algorithmus 4.11 Distanz in Meter, aus Weltkoordinaten

```
1  public static Double distanceInMeters(double latitude,
2  double longitude, double userLat, double userLon) {
3    GeodeticCalculator geoCalc = new GeodeticCalculator();
4    Ellipsoid reference = Ellipsoid.WGS84;
5    GlobalPosition pointA = new GlobalPosition(latitude,
6    longitude, 0.0);
7    GlobalPosition userPos = new GlobalPosition(userLat,
8    userLon, 0.0);
9    return geoCalc.calculateGeodeticCurve(reference,
10   userPos, pointA).getEllipsoidalDistance();
11 }
```

stelle sie als *.jar einbinden zu lassen.

Algorithmus 4.11:

In Zeile 4 wird der Typ des Ellipsoid für die Erde gewählt. In Zeile 5 wird der erste Messpunkt bestimmt aus 2 Latitude und Longitude. In der Zeile 7 der zweite, der bei einer weiteren Messung an einem anderen Ort zu einer anderen Zeit gemessen wurde. In Zeile 9-10 wird schließlich zwischen beiden Messpunkten die Distanz in Metern berechnet.

5. Sicherheits-Analyse mit Lösungen

5.1. Reverse-Engineering

Wenn für Android Java verwendet wird, ist es für einen Angreifer leichter die App mit Reverse Engineering zu verstehen, anstelle die App von außen als „Blackbox" zu analysieren. In Java geschriebene Programme können u.a. in Bytecode vorliegen. Es ist möglich Bytecode, wie auch Assembler, zu decompilieren. In decompiliertem Bytecode sind Methodennamen der App zwar nicht mehr rekonstruierbar, wohl aber Methodennamen von Aufrufen zu Java-Programmbibliotheken.

Warum ist Reverse Engineering ein Problem? Ein Tablet-Nutzer kann die App / das Programm stückweit ändern und so den Lehrer darüber täuschen, ob er überwacht wird, insbesondere wenn der Quellcode offen ist. Das ermöglicht dem Tablet-Benutzer Informationsquellen zu nutzen, die in einer Klassenarbeit nicht zugelassen sind.

Lösung des Problems: Es wird verhindert, dass der Code und diese Arbeit für alle Menschen zugänglich ist mittels eines Sperrvermerks.

5.2. Layered Security

Layered Security bedeutet: Wurde eine Sicherheitsschicht von einem Angreifer überwunden, gibt es weitere, z.B. Schichten von verschiedenen Verschlüsselungsalgorithmen und / oder verschiedenen Verschlüsselungs-Programmbibliotheken. Es ist möglich 2 Sichten durch ein Backdoor zu überwinden, z.B. kann HTTPS und OpenVPN die gleiche Sicherheits-Programmbibliothek verwenden, und zwar OpenSSL. Hat diese Bibliothek eine Schwachstelle ist eine Layered Security aus 2 Schichten nicht ausreichend oder eine Schicht muss modifiziert werden, zum Beispiel durch Verwendung einer alternativen TLS-Bibliothek.

Neben der TLS-Schicht empfiehlt sich als weiteren Schutz die Verwendung eines VPNs oder anderen Sicherheitsschichten inklusive Authentifikation. Mit einem VPN können die Tablet-Nutzer nicht untereinander Daten austauschen, denn das lässt sich unterbinden! Außerdem wird den Tablet-Benutzern damit eine weitere Hürde auferlegt den Sicherheits-Schutz zu untersuchen, weil die Datenpakete für einen Netzwerkscanner nicht im Klartext vorliegen. Allerdings ist es möglich, dass der Tablet-Benutzer auf Zertifikate und Schlüs-

Algorithmus 5.1 Android, Überprüfung des Besitzes von Zugriffs-Rechten des Manifests
Quelle: [44]

```
1 private boolean checkWriteExternalPermission() {
2   String permission =
3     "android.permission.WRITE_EXTERNAL_STORAGE";
4   int res = getContext().
5   checkCallingOrSelfPermission(permission);
6   return (res == PackageManager.PERMISSION_GRANTED);
7 }
```

sel zugreifen kann, wenn er dafür Leserechte besitzt. Diese Leserechte darf er nicht haben, sonst kann er Identitäten fälschen! Dieser Lesezugriff ist eine Hintertür, hinter mehrere Netzwerk-Sicherheitsschichten. Unter 4.1.1 bzgl. „Honeypots" ist ein Sicherheitsproblem beschrieben, was eintreten kann, wenn der Tablet-Nutzer Zugriff auf die Zertifikate seines Tablets hat.

5.3. Nachträglicher Entzug von Berechtigungen einer Applikation (Android)

Ab Version 4.3 unterstützt Android das nachträgliche Entfernen von Rechten die Apps bei der Installation erlaubt wurden. Das sind die Rechte in der Manifestdatei, bzw. die bei der Installation einer App gelistet werden. Außerdem gibt es die App SRT AppGuard zum Entziehen von Rechten, die unter [1] heruntergeladen werden kann und nicht aus dem Google Play Store installierbar ist.

Es gibt folgendes Problem: Wenn Anti-Cheat-Plus Rechte entzogen werden, kann es nicht mehr korrekt das Umgehen von Sicherheitslücken überwachen. Man braucht keine Administrator-Rechte, um Programmen in Android Rechte zu entziehen. Deshalb muss in der App jedes einzelne Recht überprüft werden. In [44] wird beschrieben, wie man das einzeln überprüfen kann. Bis jetzt haben 42 Personen, sichtbar auf der Webseite [44], bestätigt, dass diese Lösung funktioniert. Das Rechte-Überprüfen muss dem Rechner der Aufsichtsperson gemeldet werden und diese entscheidet, ob das als Betrugsversuch gewertet wird.

Erklärung zu Algorithmus 5.1:

in Zeile 2-3 wird die Zeichenkette gespeichert, die zu 100% der Bezeichnung der Android-Berechtigung einer App in Android entspricht. Dabei spielt hierbei keine Rolle, welche Berechtigung das genau ist. in Zeile 5 wird die Information über die Berechtigung beschaffen und in Zeile 6 mit einem Vergleich überprüft, ob die Berechtigung vorliegt.

[1] http://www.srt-appguard.com/de/

5.4. Erzwungenes Terminieren der Überwachungs-Applikation (Windows und Android)

Ein Programm terminiert erzwungen, wenn es mit dem Kill-Kommando beendet wird oder wenn es sich wegen Programmierfehlern für den Benutzer unbeabsichtigt beendet.

Zur Austauschbarkeit des Überwachungs-Programms kommt hinzu, dass im laufenden Betrieb der Prozess und oder Dienst auf Seiten des Schülergerätes mit einem Kill-Kommando „gewaltsam" beendet werden kann. Wenn Anti-Cheat-Plus jedoch in Abständen von z.B. 5 Sekunden Signale zum Rechner der Aufsichtsperson schickt, wird dies bemerkt und das zählt dann als Betrugsversuch.

Das Beenden mit dem Kill-Kommando wird unmöglich, wenn das Programm unter Android ein „Systemprogramm" ist, das mit Root-Rechten installiert wurde, aber die Verwendbarkeit von Administratorrechten danach wieder entfernt wurden. Man muss in dem Fall dann noch überprüfen, dass der Tablet-Nutzer sich nicht heimlich Administrator-Rechte beschafft, aber auch so ein Schutz kann umgangen werden.

Unter Windows lässt sich das Programm nicht mit dem Kill-Kommando ohne Administrator-Rechte beenden, wenn es selbst mit Administrator-Rechten läuft oder es ein Treiber (Kernelmodul) ist, der nicht als Prozess auftaucht.

Was ist aber, wenn sich das Programm wegen einem Programmierfehler unbeabsichtigt beendet? Es ist nicht möglich vollkommen abdeckend mit Try-Catch-Finally-Blöcken Abstürze zu behandeln, da damit ggf. in verwendeten Programmbibliotheken Ereignisse nicht behandelt werden.

Alle Activitys (Begriff siehe Nomenklatur) von Android werden in einem Stapel (Stack) gespeichert und die sichtbare Activity ist oben. Wenn zu wenig RAM zur Verfügung steht, beendet Android die unterste Activity gewaltsam mit dem Kill-Kommando. Unter Android ist es möglich, statt einer App einen Service zu programmieren, bzw. beides zu kombinieren. Wenn der Service zu einem Vordergrund-Service erklärt wird [45], wird er nicht mehr vom Android-System mit dem Kill-Kommando beendet, wenn zu wenig RAM verfügbar ist. In dem Fall ist es nicht mehr nötig einzustellen, dass der Dienst nach dem „gewaltsamen" Beenden wieder automatisch durch das Android-System neu gestartet wird. Wie das dennoch eingestellt wird sieht man unter der offiziellen Google-Dokumentation, auch[45].
Der Tablet-User braucht für das Kill-Kommando einen Dienst-Kill-Taskmanger. Wenn solche Apps nicht erlaubt werden, bzw. nur digital festgelegt ist, welche Apps erlaubt sind, dann wird das Öffnen von solch einem verbotenen Programm entsprechende Konse-

quenzen haben.

5.5. Manipulation von außen (Windows)

In Windows 8.1. lässt sich, wie zu Anfang dieser Arbeit beschrieben, der Kioskmodus hauptsächlich mit Mitteln des Betriebssystems realisieren. Zur Automatisierung, Verwaltung und Überwachung wird jedoch Anti-Cheat-Plus benötigt. Eine zentrale Rolle spielt dabei das Erlauben und Verbieten von ausführbaren Dateien. Dabei gibt es nicht nur die anfangs beschriebenen Sicherheitsprobleme. Wenn ganze Verzeichnisse erlaubt werden, dann kann der Tablet-Nutzer, mit verschiedenen Möglichkeiten das Schreibverbot von „Außen" zu umgehen, ausführbare Dateien in den Ordner kopieren und sie zur Zeit der Klassenarbeit ausführen, und auf diese Weise betrügen. Die Möglichkeiten sind das Ausbauen der Festplatte, das Entfernen des Akkus, der das BIOS Passwort aufrechterhält und dann das Starten eines alternativen Systems, das auf die Festplatte vollen Zugriff hat. Deshalb führt so kein Weg daran vorbei ausführbare Dateien für jede Datei einzeln mit Hashwerten zu sperren und nach Windowsupdates diese Hashwerte im Sperr-System zu aktualisieren für den Lehrer-Überwachungs-Computer.

Die Alternative ist es mit asymmetrischer Verschlüsselung zu arbeiten, so dass der Tablet-Benutzer Leserechte für ausgewählte Bereiche hat (mittels des öffentlichen Schlüssels), in denen der Lehrer von der Ferne Schreibrechte hat (privater Schlüssel), oder es wird mit ROMs gearbeitet, z.B. CD-ROMs, die der Tablet-Benutzer nicht manipulieren kann, weil sie nur einmal beschreibbar sind. Ein Austausch der CD-ROM ist möglich, aber es kann mit digitalen Unterschriften gearbeitet werden, die die Authentizität nachweisen kann (UEFI mit Secure Boot). Wird mit asymmetrischer Verschlüsselung gearbeitet, gibt es das Problem, dass dies den Prozessor belastet und unter Umständen zu Wartezeiten führen kann. Asymmetrische Verschlüsselung langsamer als symmetrische Verschlüsselung. Deshalb ist die Geschwindigkeit für Dateigrößen unter 25 Megabyte (z.B. E-Mails unter 25 MB mit PGP / GPG) vertretbar. Z.B. hat das symmetrische obsolete DES eine Zeitkomplexität von O(Log n) und das asymmetrische RSA $O(n^3)$ [46].

Eine weitere Alternative ist es spezielle Geräte zu betreiben, die Tablet-Benutzer nicht mit nach Hause nehmen können, und die physisch vor Manipulation geschützt sind. In denen sind keine weiteren Administrator- oder andere Benutzer-Konten, die für den Tablet-Nutzer zugreifbar sind, sondern nur ein Kioskbenutzer-Account.

Insgesamt kann dieses Thema nicht in dieser Arbeit vollständig behandelt werden, da das den Rahmen sprengt. Es ist möglich, dass das Problem generell prinzipiell nicht lösbar ist.

Mit UEFI + Secure-Boot existiert eine Technologie, in Kombination mit Windows 8 ab

2012 (das erste Betriebssystem mit Secure-Boot-Unterstützung), mit der die Authentizität eines Betriebssystemherstellers oder des Bootloaders überprüft werden kann. So wird verhindert, dass Schadsoftware in dem Moment des Systemstarts vor dem Starten des Betriebssystems geladen wird. Z.B. muss in Linux der kompilierte Kernel signiert sein, wenn er mit aktiviertem Secure Boot geladen wird, damit das Betriebssystem GNU/Linux bzw. Linux starten kann. Mit Secure Boot lässt sich überprüfen, dass der Tablet-User das Bootmedium nicht ausgetauscht hat, wenn Secure Boot noch weitergehend Überprüfungen durchführt, wenn es diese Funktion hat. Secure Boot hat aber nicht diese Funktionalität! Es ist nur sichergestellt, dass vor dem Betriebssystemstart keine Schadsoftware geladen wurde und dass die Startkomponenten des Betriebssystems orginal sind.

5.6. Manipulationsüberwachung (Android + Windows)

Es gibt die Möglichkeit, dass das Serverprogramm und das Clientprogramm überwacht wird, nicht manipuliert worden zu sein. Solche Techniken verwendet Steam und ein Teil der Antivirenprogramme.

Zum Beispiel können Android-Apps signiert werden. Eine Unterschrift gehört zu einem Entwickler. Damit kann eine App zu einem Entwickler zugeordnet werden. Das Android-Betriebssystem verlangt, dass Apps digital signiert sind. Wenn nun ein Tablet-Nutzer die App reengineered hat, dann kann er sie nicht mit der digitalen Unterschrift des ursprünglichen Autors unterschreiben. Nun muss nur noch das System den Autor überprüfen und dies dem Lehrer über das Netzwerk melden. Diese Nachricht kann manipuliert worden sein und so kann vorgetäuscht werden, dass die App nicht vom Tablet-User manipuliert wurde. Mit einer digitalen Unterschrift ist lediglich ein Urheber bestätigt. Die Überprüfung ist sicher, wenn sie direkt am Schülertablet mit Boardmitteln durchgeführt wurde, statt über Netzwerk, und das Gerät nicht vom Tablet-Nutzer gerootet ist oder gerootet worden war.

Denkbar ist, dass der Lehrer ein zufälligen oder kryptographischen Code (z.B. ein Hash) zum Schülertablet schickt, und das Tablet mit diesem Code und seiner Signatur etwas berechnet (z.B. ein Hash / eine Unterschrift), das dem Lehrercomputer geschickt wird. Das Ganze hat das Ziel die Integrität der Überwachungssoftware zu überprüfen, aber da die Berechnung auf der Schülertabletseite liegt, hat der Tablet-Benutzer freie Hand mittels Reverse Engineering den Lehrercomputer zu täuschen. Schließlich kann er den Algorithmus der Berechnung verwenden und anpassen und er hat lesenden Zugriff auf die Signatur der App.

Unter [47] ist beschrieben, dass das Signierungssystem umgangen werden kann, wenn es eine Lücke in Android gibt, die in dem Artikel thematisiert wird. Deshalb ist es wichtig, dass die Androidversion akuell gehalten werden muss.

Unter Visual Studio gibt es auch die Möglichkeit des Signierens und zwar von Assemblys. Eine Assembly besteht aus dem Programm oder der Programmbibliothek, ggf. Schlüsselpaaren und ggf. Zertifikaten, der Manifestdatei und Angaben, wie z.B. der Versionsnummer.

Dann gibt es noch die Möglichkeit Manipulationen zu erschweren, statt sie ganz zu verhindern, aber darauf wird nicht näher eingegangen, da das nicht genügt!

Unter [48] wird das Problem beschrieben, dass ein Teil der Antiviren-Programme durch Schadsoftware verändert werden kann und es von den Herstellern dagegen zu wenig Schutz gibt. Die Antivirenprogramme sind damit selbst ein Sicherheitsrisiko. Das ist insofern relevant, weil die Überwachungsapp selbst ein Einfallstor darstellen kann.

Diese Arbeit vertieft dieses Thema nicht weiter, weil das den Rahmen sprengt.

5.7. Der Steam-Manipulationsschutz

Das Spieleplattform Steam verwendet einen Raubkopier-Schutz, den sie ansatzweise erklärt [49]. Dabei werden *.exe-Dateien für jeden Software-Käufer einzeln gefertigt. Das bedeutet, dass das gleiche Programm gleich funktioniert, sich jedoch im Binärcode für den einzelnen Käufer unterscheidet. Das ist zum Beispiel möglich, indem eine Funktion oder Methode in der Binärdatei woanders positioniert wird und die zugehörigen Zeiger dazu angepasst werden. Auf diese Art lässt sich Software quasi „permutieren”. Ein Cracken kann damit nicht verhindert werden. Jedoch muss die ganze Steam-Plattform gecrackt werden, wenn ein Spiel gecrackt werden soll, was es unmöglich macht, weiterhin mit Steam Spiele käuflich zu erwerben. Aus der „Rechnung” der Käufer zwischen Aufwand und Nutzen, entscheiden sich deshalb Menschen dazu, weiterhin Spiele zu kaufen und nicht zu cracken. Steam muss, in Abständen von Tagen, Updates erhalten und benötigt eine Onlineverbindung. Das gehört mit in das Konzept gegen das Cracken von Software bzw. Spielen.

Was für eine Rolle spielt diese Schutz-Funktionalität von Steam für diese Arbeit? Anti-Cheat-Plus kann vom Tablet-Nutzer gecrackt worden sein. Damit das detektiert werden kann, ist daher die Anti-Raubkopier-Methode, die die Programmierer von Steam implementiert haben, eine Option Schutz vor Manipulation zu haben. Softwarepatente und ungenügendes unvollständiges Wissen über die Methode, mit der Steam vor Raubkopiererei schützt, können dabei ein Problem darstellen. Es gibt aber ggf. die Option Geld an Valve zu bezahlen, um die Methode von Steam nutzen zu können.

6. Programmierungen von Prototypen

Der Quellcode befindet sich auf der zur Arbeit gehörigen CD. Programme für Windows lassen sich mit Visual Studio 2013 für c# öffnen. Programme für Android lassen sich mit Eclipse Luna in Kombination mit dem Android SDK öffnen.

6.1. KioskService - Windows

Im Ordner „KioskService_wichtig” liegt ein C# - Prototyp.
In [50] findet sich eine Klasse zur Administrierung von Windows-Benutzern und Gruppen, auf die für diesen Prototypen zurückgegriffen wird (class LocalMachine).

Laut Spezifikation darf der Tablet-Nutzer nie Administratorrechte besitzen, wenn er das selbe Gerät für computergestützte Prüfungen verwendet. Dennoch müssen auch Administrator-Konten deaktiviert und aktiviert werden können, z.B. wenn die Aufsichtsperson auf dem Tablet solch einen hinterlegt hat, oder wenn nur noch deaktivierte Administrator-Konten vorliegen, die zu aktivieren sind.

Relevantes zur Klasse „users”:

1. Zum Deaktivieren eines Windows-Account-Benutzers wird mit „LocalMachine.GetGroupMembers("Administratoren", out members)” erfragt welcher Benutzeraccount zur Gruppe der Administratoren gehört.

2. Wenn der zu deaktivierende Benutzer dazu gehört dann wird mit „LocalMachine.RemoveUserFromGroup(user, "Administratoren")” dieser aus dieser Gruppe entfernt, damit der Benutzer deaktiviert werden kann.

3. mit „LocalMachine.EnableDisableUser(user, false)” wird er schließlich deaktiviert

4. und er erhält seine Admin-Rechte wieder mit „LocalMachine.AddUserToGroup(user, "Administratoren")”.

Zum Aktivieren genügt
„LocalMachine.EnableDisableUser(user, true)”

Klasse „AsynchronousClient” und „networking” ist die grundsätzliche Implementierung

eines asynchronen Clients und Servers nach Anleitung [51, 52].

Klasse „protocol" ist eine prototypische unvollständige Umsetzung, die die Verarbeitung des Netzwerkprotokolls darstellt. Mit einer Switch-Case-Anweisung wird überprüft, welcher der Netzwerk-Befehle übermittelt wurde und dazu wird die passende Funktion gewählt und deren Rückgabewert wird ggf. weiter verarbeitet.

Klasse „Service1" ist schließlich die Umsetzung eines Windows-Dienstes nach Anleitung [53].

6.1.1. Benutzer-Konten auflisten

Algorithmus 6.1 ist selbsterklärend. Dem GUI-Element, eine Listbox, werden Listenelementen zugewiesen vom Typ String mit dem Inhalt eines Benutzers, so dass alle lokalen Benutzer aufgelistet werden.

Algorithmus 6.1 Benutzer-Konten auflisten

```
SelectQuery query = new SelectQuery("Win32_UserAccount");
ManagementObjectSearcher searcher =
  new ManagementObjectSearcher(query);
foreach (ManagementObject envVar in searcher.Get())
{
   this.listBox1.Items.Add(envVar["Name"]);
}
```

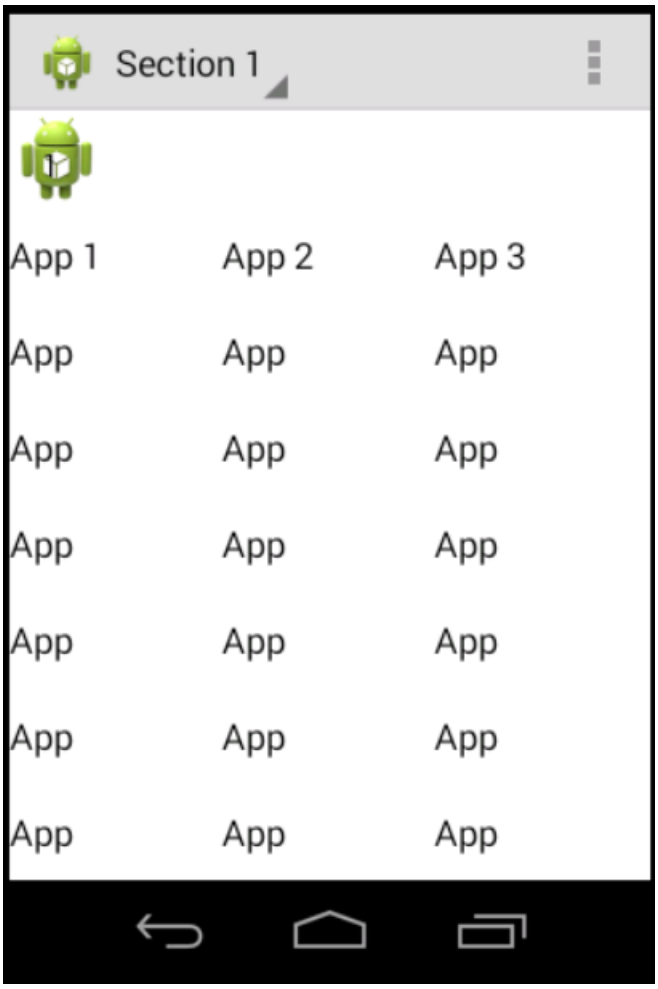
6.2. Anti-Cheat-Plus-Prototyp - Android

Für Android wurde mit einem Eclipse-Projekt ein Anti-Cheat-Plus-Prototyp erstellt. Als Layout für startbare Apps wurde eine GridView gewählt, siehe activity_main.xml auf der CD. Diese wird als Tabelle dargestellt wie in Bild 6.1.

Diese App ist im Code zu einem Homescreen erklärt worden. Wie das geht, was das bedeutet, und wozu das sinnvoll ist wurde in Unterabschnitt 4.6.2 erklärt.

Die ersten 3 Tabellenzellen in Leserichtung „App1", „App2" und „App3" bewirken in folgenden Unterabschnitten beschriebene Reaktionen. Alle weiteren sind Attrappen zum Aufruf von festzulegenden Apps, die in einer Klassenarbeit zugelassen sind. Die Aufsichtsperson muss diese per Fernwartung festlegen. In der Praxis sind die ersten 3 Tabellenzellen für die Funktion des Aufrufs von Apps festzulegen.

Abbildung 6.1.: Anti-Cheat-Plus-Prototyp - Android



6.2.1. Apps aus Apps starten, Bluetooth deaktivieren, aktive Apps auflisten

Mit dem Tippen auf „App 1" wird im Java-Code folgende Methode ausgeführt: „RunApp("com.android.settings");" bei der Algorithmus 4.7 ausgeführt wird. Das bedeutet, dass die Einstellungs-App von Android gestartet wird, die den Einstellungen der Windows-Systemsteuerung entspricht. Hier sollte für diese Arbeit überprüft werden, wie Apps aus anderen Apps gestartet werden können, siehe Algorithmus 4.7.

Ein Tippen auf „App 2" deaktiviert Bluetooth und führt zu einem Fehler, wenn das Gerät kein Bluetooth unterstützt, z.B. wenn die Funktion im Emulator ausgeführt wird. In Unterabschnitt 4.5.3 wurde näher darauf eingegangen.

Der Aufruf von „App 3" bewirkt eine darauf folgende Ausgabe in Logdateien. Wie in Unterabschnitt 4.7.1 beschrieben, erfolgt hier die Ausgabe aller aktiven Apps, Dienste und Tasks in der Logdatei.

6.2.2. Protokollierung des Wechsels der sichtbaren App

Die Methode „AppInForegroundChanged()" bewirkt, dass in der Logdatei ausgegeben wird, wann die für den Nutzer sichtbare App wechselt, so dass eine andere sichtbar ist. Insgesamt wird das mit diesem Vergleich bewerkstelligt:
„(! topActivity.getPackageName().equals(LasttopActivity.getPackageName()))". LasttopActivity ist eine Variable, die der „Activity" entspricht, die beim letzten Aufruf der Methode „AppInForegroundChanged()" für den Tablet-Nutzer sichtbar war. In Algorithmus 6.2 ist Code, der im Android-Projekt verteilt vorliegt, jedoch hier zusammengehörig der Veranschaulichung dient. Dieser Code ist nötig, damit abgerufen werden kann, was die aktuelle Top-Activity ist, also der App-Bestandteil einer App aus allen aktiven Apps, der in diesem Moment für den Nutzer sichtbar ist.

Algorithmus 6.2 vorderste sichtbare Activity in einer Activity-abgeleiteten-Klasse abrufen

```
ActivityManager am = (ActivityManager) getApplicationContext().
  getSystemService( Context.ACTIVITY_SERVICE);
tasks = am.getRunningTasks(1);
ComponentName topActivity = tasks.get(0).topActivity;
```

6.2.3. Informationen über alle installierten Apps gewinnen (Name, Version, etc.)

Algorithmus 6.3 zeigt, wie alle installierten Apps ausgelesen werden können, wobei hier z.B. verzichtet wurde die Klasse PInfo zu zeigen, denn diese dient nur der Ausgabe der Informationen, siehe mitgelieferte CD (selbsterklärend).

6.2.4. Standard-Homescreen wechseln

Die Methode „onOptionsItemSelected", in der Klasse MainActivity, wird ausgeführt, wenn das Menü oben Rechts vom Prototypen aufgerufen und ein Menüpunkt angewählt wurde. Wenn der einzige Menüpunkt in diesem Menü das erste Mal gewählt wird, dann folgt das Auswählen des Homescreens, z.B. zu „Launcher" (der Standardhomescreen) oder dem Anti-Cheat-Plus-Prototypen als Einstellung in Android. Beim zweiten Mal funktioniert das wieder genauso, aber beim dritten Mal passiert nichts. Die Funktionalität muss aber immer ausgeführt werden können und darf nicht unerwartet versagen.

In Unter-Abschnitt 4.6.2 wurde u.a. auf [27, 28] verwiesen, wo beschrieben wird, wie

Algorithmus 6.3 Informationen über alle installierten Apps gewinnen

```
private ArrayList<PInfo> getInstalledApps(
boolean getSysPackages) {
  ArrayList<PInfo> res = new ArrayList<PInfo>();
  List<PackageInfo> packs = getPackageManager().
    getInstalledPackages(0);
  for (int i = 0; i < packs.size(); i++) {
    PackageInfo p = packs.get(i);
    if ((!getSysPackages) && (p.versionName == null)) {
      continue;
    }
    PInfo newInfo = new PInfo();
    newInfo.appname = p.applicationInfo.loadLabel(
    getPackageManager()).toString();
    newInfo.pname = p.packageName;
    newInfo.versionName = p.versionName;
    newInfo.versionCode = p.versionCode;
    newInfo.icon = p.applicationInfo.loadIcon(
      getPackageManager());
    res.add(newInfo);
  }
  return res;
}
```

mit einem Workaround dauerhaft der Homescreen umgestellt werden kann. Dies wurde im Prototyp umgesetzt und ausgetestet, u.a. in der Methode „onOptionsItemSelected".

Dass aber beim dritten Mal nicht wieder, wie in diesem Unterabschnitt beschrieben, keine Auswahl mehr erfolgt, kann u.a. damit gelöst werden, dass die Android-Einstellungen aufgerufen werden können. Denn dort lässt sich der Standard-Homescreen wechseln.

Nun müssen die Einstellungen verborgen werden, damit diese nicht als Sicherheitslücke in Testaten ausgenutzt werden können. Mit der globalen Einstellung dass „jetzt" ein Testat geschrieben wird oder nicht, lässt sich jedoch der direkte Aufruf der Android-Einstellungen aus dem zusätzlichen App-Launcher (Anti-Cheat-Plus) erlauben oder verbieten. Launcher heißt sowohl der Standardhomescreen, als auch die Kategorie der Apps, die als Homescreen-App zum Starten von Apps eingesetzt werden können.

6.3. GPS mit Android

Im Abschnitt 4.8 bzw. 4.8.2 und 4.8.5 ist u.a. beschrieben, wie auch die GPS-Detektion im GPS-Prototypen umgesetzt wurde. Bild 4.9 zeigt, wie der Prototyp aussieht.

Dieser Prototyp funktioniert nur in Geräten mit Bluetooth-Unterstützung, also nicht im Emulator zu der Eclipse-Entwicklungsumgebung.

Bedienung:
Es gibt 3 Buttons. Die Bedienung des ersten gibt die GPS-Koordinaten direkt unter dem Button aus, wenn diese beschaffen werden können. Ggf. muss gewartet werden. Des Weiteren wird in Abständen von 3 Sekunden die GPS-Verbindung überprüft und in diesen Abständen die Erd-Koordinaten als Android-„Toast"-Nachricht (Feld mit Text auf dunkelgrauem Untergrund mit abgerundeten Ecken) jeweils ausgegeben.

Durch Berühren des mittleren Buttons speichert der Prototyp die darüber angezeigten Koordinaten, für den Benutzer zunächst nicht sichtbar, in weiteren Variablen. Wenn man sich dann 20 Meter von dem aktuellen Ort weg bewegt und den untersten Button betätigt, dann wird der Abstand in Meter darunter ausgegeben.

Auf [54] wird dargelegt, dass GPS eine Genauigkeit von 5 bis 20 Metern besitzt. Mit Korrektursignalen ist eine Verbesserung auf 1-3 Metern möglich.

6.4. Sockets - zwischen Java für Android und Java für Windows / Linux

Auf Zitate aus den Quelltexten wird in diesem Unterabschnitt wegen der notwendigen Menge zum Verständnis deshalb ganz verzichtet, Quelltext siehe CD.

In Java für Windows / Linux wurde eine Client programmiert. Für Android wurde ein Server programmiert.
Bevor der PC-Client mit dem Android-Server kommunizierte, wurde getestet, ob das Android-Programm mit sich und das PC-Programm mit sich kommunizieren kann, als stufenweise Problemnäherung. Deshalb existiert zusätzlich Code, der nicht ausgeführt wird, aber referenziert werden kann, so dass es möglich ist, das PC-Programm als Server und das Androidprogramm als Client zu verwenden. Im Code ist jedoch das PC-Programm der Client und die Android-App der Server.

Damit der Code als Programm funktionstüchtig auf einem anderen System reproduziert werden kann, ist es notwendig, die TLS-Containerdateien zum Dateisystem richtig zu referenzieren (z.B. richtige Pfadangabe als Übergabeparameter) oder in der App oder der Jar-Datei (Dateien: keystore.jks (für PCs) und keystore.bks (für Android)). Ggf. können statt einer Datei, als Keystore und Truststore, aus Sicherheitsgründen 2 sich im Inhalt unterscheidende Dateien verwendet werden (siehe Unterabschnitt 4.2.2), d.h. je eine. Ggf. muss ein Keystore und Truststore mit dem Oracle Keytool erstellt werden, siehe auch

Unterabschnitt 4.2.2.

7. Geschwindigkeits-Messungen

Wenn die Lösung, die diese Arbeit beschreibt zu viel Rechenzeit in Anspruch nimmt, so dass es nicht zu einer digitalen Klausur kommt bzw. zu wenig Zeit für diese übrig bleibt, dann ist sie de facto nicht nutzbar. Es gibt gewisse Performance-Gesichtspunkte, die zu untersuchen sind, auch als Flaschenhals bezeichnet. Dazu zählen:

- Sslsocket initialisierung mit Handshake
- Senden von TLS-Socket-Paketen
- Empfangen von TLS-Socket-Paketen

Der Quellcode der Messung befindet sich auf der zugehörigen CD zu dieser Arbeit.

7.1. Messaufbau

Gemessen wurde mit einem Linux-Laptop mit einem Core i5-4200U-Prozessor, 8 GB RAM, der 30 Clientverbindungen aufnimmt zu einem Tablet „Acer Iconiatab A211" mit Android 4.1.1, NVIDIA-Tegra-3-Prozessor mit 4x1,2GHz und 1GB RAM, das mit einem Serverprogramm diese 30 Verbindungen aufnimmt.

Es stand eine WLAN-n-Verbindung zur Verfügung mit 65 Megabit pro Sekunde, als die Zeitabstände gemessen wurden. Zum heutigen Zeitpunkt gibt es WLAN-Geschwindigkeiten bis 1300 Megabit, jedoch sind für den TLS-Handshake CPU-Last und Netzwerklatenzen relevant.

Wenn mit einem Tablet gezeigt wird, dass die Geschwindigkeiten von 30 Verbindungen den Ablauf im Klassenraum nicht behindern, dann ist gezeigt dass die Lösung mit 30 Tablets mit je einer Verbindung funktioniert, wenn eine Sterntopologie-Kabelnetzwerkverbindung genutzt wird. 30 Tablets absolvieren die Aufgaben $\leq$30 mal schneller, wenn in allen Testgeräten mit einem Kern gearbeitet wird. Wenn 30 Tablets mit dem WLAN-Accesspoint verbunden sind, kann das jedoch auch zur Folge haben, dass die WLAN-Verbindungen der einzelnen Tablets langsamer sind, als wenn nur ein Tablet verbunden ist. Allerdings sind für den TLS-Handshake Netzwerk-Latenzen und Prozessorleistung der teilnehmenden Geräte relevant.

Es werden also parallel zwischen 2 Geräten 30 Verbindungen hergestellt mit einmal parallelem bzw. gleichzeitigem Senden und Empfangen.
Gemessen wird für die TLS-Initialierung von den ersten Befehlen dafür bis zum Abschluss der letzten TLS-Initialisierungen. Zur Messung von „allem" ist der Messendpunkt dagegen der letzte der 2 Messpunkte, das Ende aller Datenaustauschbefehle zwischen Client und Server. Der erste Messpunkt von beiden Messungen (Dauer von „allem" oder nur die Handshakedauer) ist der gleiche.

Die Summe der 30 Sende- und Empfangszeiten ist addiert, d.h. dass nicht vom ersten Senden bis zum letzten Ende des parallelen Sendens und Empfangens gemessen wurde.
.
Gesendet und Empfangen werden 300 Java-Char-Elemente (UTF-16).

7.2. Messergebnisse in Tabellenform

Von der Linux-Clientseite aus wurde gemessen.

$\bar{t}$ - ist der Durchschnitt
σ - ist hier die „empirische Standardabweichung" aus so genannter „korrigierter Stichprobenvarianz"
MSE - ist die mittlere Quadratische Abweichung

Tabelle 7.1.: Geschwindigkeits-Messungen in ms, 1. Durchlauf

in ms	TLS-Initialierung	ΣSenden	ΣEmpfangen	alles
$\bar{t}$	12819,40	21,90	512,80	12819,90
Median	12614,00	17,00	477,50	12614,50
σ	1073,70	15,72	127,19	1073,66
MSE	1037548	222	14560	1037477

Tabelle 7.2.: Geschwindigkeits-Messungen in ms, 2. Durchlauf

in ms	TLS-Initialierung	ΣSenden	ΣEmpfangen	alles
$\bar{t}$	11720,90	6,60	757,70	11721,00
Median	11832,50	5,50	742,00	11832,50
σ	917,17	6,11	143,87	916,95
MSE	757074	34	18630	756714

7.3. Auswertung der Messergebnisse

Die benötigte Zeit liegt insgesamt bei unter 13 Sekunden. Damit bleibt genug Zeit übrig für die Leistungsüberprüfung der Tablet-Nutzer. Wenn jedoch 30 Geräte je 30 WLAN-Verbindungen verwenden, beeinträchtigt das die Geschwindigkeit des gesamten Netzwerkes. Dies konnte jedoch nicht gemessen werden, da dazu 30 Geräte benötigt werden.

8. Schluss

8.1. Zusammenfassung

Die Aufgabenstellung dieser Arbeit war es, dass die sicherheits-technischen Rahmenbedingungen, zum Durchführen von Leistungsüberprüfungen mit Tablets, geschaffen werden. In Leistungsüberprüfungen soll vermieden werden, dass verbotene Informationsquellen bezogen werden.
Als Leistungsüberprüfung zählt neben Klausuren alles andere, was mit Tablets machbar ist. Es wurde gezeigt, dass dies unter mindestens 2 Betriebssystemen realisierbar ist und mit 2 Methoden, der Überwachung (inklusiv-)oder der Einschränkung des Handlungsspielraums. Per Netzwerkfernwartung wird die Leistungsüberprüfung eingeleitet und beendet.

Dazu wurde aufgelistet und beschrieben, welche Einschränkungen umgesetzt werden müssen und was schon vom Betriebssystem mitgeliefert wird und noch konfiguriert werden muss, nämlich die Blockade von ausführbaren Dateien und die Einschränkfunktionen der Gruppenrichtlinien unter Windows, z.B. um Laufwerke zu verstecken anstelle den Windows-Explorer zu deaktivieren.

Es wurde ein Netzwerkprotokoll entworfen, Betriebssystemfunktionen recherchiert, in ihrer Wirkung technisch überprüft und der Beweis dieser Überprüfung dazu in der Arbeit hinterlegt.

Damit die verantwortliche Beaufsichtigungs-Person der Leistungsüberprüfung keine Lehrkraft aus dem IT-Bereich sein muss, wurde beschrieben wie das Melden und Auswerten von Betrugsversuchen automatisiert werden kann. Damit Leistungsüberprüfungens-Arten hinzugenommen werden können, wurde beschrieben, wie mit Profilen neue Sicherheitskonfigurationen angelegt werden können.

Es wurde festgestellt, dass eine Sicherheitslücke dazu führen kann, dass die gesamte Schutzvorrichtung nutzlos wird. Dazu zählt auch ein Programmierfehler, der zum Absturz inklusive dem Beenden dieser Schutzvorrichtung führt.

Nicht nur die Umgebung des Tablets muss vor dem Ausnutzen von Sicherheitslücken geschützt werden, sondern auch die Netzwerkverbindung. Die Sicherstellung von Inte-

grität, Authentizität und dem Schutz vor dem Ausspähen der Netzwerkdaten ermöglichen TLS-Sockets. Es wurde für Android und Java für Windows und Linux deshalb prototypisch eine TLS-Socketverbindung als Realisierung überprüft, Laufzeiten gemessen, beschrieben wie mit Hilfe von verschiedenen zu kombinierenden Quellen TLS-Sockets realisiert werden können und der sicherheitstechnische Zusammenhang ausformuliert.
Zusätzlich gibt es Standards für Netzwerkauthentifikations-Methoden, auf die in der Arbeit kurz eingegangen wurde. Auch der Computer der Beaufsichtigungsperson muss per TLS-Socketverbindung seine Authentizität nachweisen.

In dem Kapitel 6 wurde dargelegt, was für Prototypen programmiert wurden, und beschrieben wo in der Arbeit die Nachweise deren Funktionstüchtigkeit hinterlegt wurden. Des Weiteren werden zu dieser Arbeit die Quelltexte mitgeliefert.

Es konnte mit dieser Arbeit gezeigt werden, dass es möglich ist, mit Rechnern beliebige Leistungskontrollen durchzuführen. Es hat sich herausgestellt, dass es zu bewältigende Tücken gibt und mehr Erfahrungen gesammelt werden müssen, denn es gibt ein Potential für nicht auf den ersten Blick erkennbare Fehlerquellen. Des Weiteren sind solche Fehler nicht akzeptabel, da Leistungsüberprüfungen, sowohl für Dozenten / Lehrer / Professoren, als auch für Schüler und Studenten, Reibungspunkte im Leben darstellen.

8.2. Ausblick

Wenn mit Anti-Cheat-Plus der sicherheitstechnische Rahmen geschaffen worden ist, dass z.B. Klausuren mit Tablets geschrieben werden können, kommen neue zwischenmenschliche Fragen auf. Ein Problem ist es, dass damit Aufsichtspersonen wie z.B. Lehrer eine neue Form der Verfügungsgewalt in die Hand bekommen. Denn wenn die Aufsichtsperson einen Betrugsversuch proklamiert, dann ist unter Umständen für andere Tabletnutzer nicht nachvollziehbar, ob das gerechtfertigt war. Das liegt daran, dass gewöhnlich nur jeder den Inhalt seines Tablets / PCs sieht.

Es ist denkbar, dass das Melden von dem Umgehen von Sicherheitsbarrieren nicht gerechtfertigt ist, wenn es kein tatsächlicher Versuch des Tablet-Nutzers war, verbotene Informationsquellen zu beziehen. Das Problem ist nämlich, dass Software in der Regel Fehler verschiedener Kategorien enthält (z.B. Kategorie: logischer Fehler). Erst durch ausgiebiges Testen (Unit-Test, Betatest, etc.) und Überprüfen der Software, lassen sich solche Fehlerquellen ergründen.

Die Verwendung von Papier und Stift bereitet nicht die speziellen Risiken, die soeben beschriebenen wurden.

Die Prioritäten der Entwicklung von Anti-Cheat-Plus müssen in der Zukunft vor allem bei

der Programm-Fehlerbeseitigung liegen, weil der Schaden fälschlicher Beschuldigungen wegen Betrugs größer ist, als wenn ein Tablet-Nutzer unerkannt eine Sicherheitslücke ausnutzt, die ihn nur bedingt zu einer besseren Note verhilft. Außerdem können Programmfehler auch dazu führen, dass unerkannt Sicherheitslücken ausgenutzt werden können.

Absolute Sicherheit gibt es nur in Ausnahmefällen und mit Eingrenzungen, z.B. mit Einschränkungen bei Verwendung der Verschlüsselung One-Time-Pad. Deshalb kann es zukünftig ggf. eine Weiterentwicklung der Sicherheitsmaßnahmen von Anti-Cheat-Plus geben, wenn Probleme erkannt worden sind.

Gewöhnlich werden Displays bevorzugt, die bei seitlicher Betrachtung nicht dunkler erscheinen, als bei frontaler Betrachtung. In Leistungskontrollen mit Tablets können Displays die seitlich abdunkeln jedoch den Vorteil bieten, dass benachbarte Tablet-Nutzer nicht abschauen können.
Es wurde in der Arbeit festgestellt, dass es noch sicherer ist für Leistungsüberprüfungen extra Tablets in der Schule zurückzuhalten, die nur für diesen Einsatzzweck gedacht sind. Wenn das so gehandhabt wird, dann ist es von Vorteil, beim Kauf der Tablets, die nur für Leistungstests gedacht sind, auf ein Display zu achten, dass seitlich maximal abdunkelt.

Bei entsprechender Stückzahl, ggf. weltweit, ist es denkbar, dass speziell Tablets zur Produktion für Schulen in Auftrag gegeben werden, deren Display bei seitlicher Betrachtung schlecht gelesen werden kann. Das hat den Vorteil, dass Tablet-Nutzer näher nebeneinander sitzen können und deshalb weniger Räumlichkeiten benötigt werden, als wenn eine Klausur mit Papier und Stift geschrieben wird. Das wiederum erlaubt mehr Freiheiten in der Planung der Bildungs-Einrichtung. Außerdem ist die Verwendung solcher Klausur-Tablets sicherer, weil die Tablet-Nutzer die Geräte nicht zuhause manipulieren können, denn diese bleiben in der Schule und werden nur bei Prüfungen herausgegeben.

Nomenclature

Activity Eine Activity entspricht genau einem sichtbaren Benutzerinterace, z.B. ein Bildschirm mit Menüs und den zugehörigen Aktionen. Eine Applikation kann mehrere Activitys besitzen, z.B. öffnet sich durch Betätigen des Menüs eine weitere Activity und die vormals sichtbare Activity bewegt sich im Activitystapel aller gestarteten Apps um 1 nach hinten.

Intent Intents sind eine abstrakte Angabe für Operationen die ausgeführt werden, z.B. als Interprozesskommunikation mit dem Starten einer Aktivity / App, zum Senden von so genannten Broadcasts, starten von Diensten, etc.

Theme Ein Theme ist ein Stil, der für eine ganze Activity oder Applikation angewendet wird. Der Theme liegt als XML-Datei vor.

Ein Stil ist eine Zusammenstellung von Eigenschaften, die das Aussehen und die Formatierung für eine View oder ein Fenster spezifizieren.

Die Klasse View bietet die grundsätzlichen Funktionen und Eigenschaften für Benutzerinterfacekomponenten.

A. Anhang

A.1. Anzeige der installierten Apps

Das ist die Ausgabe des Programmes aus [39] im Android-Emulator, die mit dem Zeichen „|” versehen wurde statt Tabulator:

Sound Recorder| com.android.soundrecorder| 4.4.2-999428| 19
com.android.sdksetup| com.android.sdksetup| 4.4.2-999428| 19
Launcher| com.android.launcher| 4.4.2-999428| 19
Package Access Helper| com.android.defcontainer| 4.4.2-999428| 19
com.android.smoketest| com.android.smoketest| 4.4.2-999428| 19
Search| com.android.quicksearchbox| 4.4.2-999428| 19
Contacts| com.android.contacts| 4.4.2-999428| 19
Android Keyboard (AOSP)| com.android.inputmethod.latin| 4.4.2-999428| 19
Phone| com.android.phone| 4.4.2-999428| 19
Calculator| com.android.calculator2| 4.4.2-999428| 19
ProxyHandler| com.android.proxyhandler| 4.4.2-999428| 19
HTML Viewer| com.android.htmlviewer| 4.4.2-999428| 19
Connectivity Test| com.android.emulator.connectivity.test| 1.0| 1
dritterVersuch| com.example.dritterversuch| 1.0| 1
Calendar Storage| com.android.providers.calendar| 4.4.2-999428| 19
Input Devices| com.android.inputdevices| 4.4.2-999428| 19
Custom Locale| com.android.customlocale2| 1.0| 1
Calendar| com.android.calendar| 4.4.2-999428| 19
Browser| com.android.browser| 4.4.2-999428| 19
Music| com.android.music| 4.4.2-999428| 19
NetSpeed| com.android.netspeed| 4.4.2-999428| 19
Widget Preview| com.android.widgetpreview| 4.4.2-999428| 19
Example Wallpapers| com.example.android.livecubes| 4.4.2-999428| 19
Downloads| com.android.providers.downloads.ui| 4.4.2-999428| 19
User Dictionary| com.android.providers.userdictionary| 4.4.2-999428| 19
Documents| com.android.documentsui| 4.4.2-999428| 19
com.android.sharedstoragebackup| com.android.sharedstoragebackup| 4.4.2-999428| 19
VpnDialogs| com.android.vpndialogs| 4.4.2-999428| 19

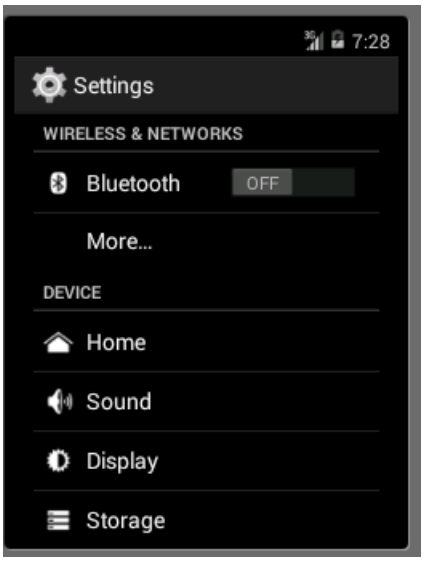
Messaging| com.android.mms| 4.4.2-999428| 19
PacProcessor| com.android.pacprocessor| 4.4.2-999428| 19
Media Storage| com.android.providers.media| 4.4.2-999428| 700
Certificate Installer| com.android.certinstaller| 4.4.2-999428| 19
API Demos| com.example.android.apis| 4.4.2-999428| 19
Print Spooler| com.android.printspooler| 1| 1
Fallback| com.android.fallback| 4.4.2-999428| 19
com.android.gesture.builder| com.android.gesture.builder| 4.4.2-999428| 19
Camera| com.android.gallery| 4.4.2-999428| 19
Android System| android| 4.4.2-999428| 19
Settings| com.android.settings| 4.4.2-999428| 19
Contacts Storage| com.android.providers.contacts| 4.4.2-999428| 19
Home screen tips| com.android.protips| 1.0| 1
External Storage| com.android.externalstorage| 4.4.2-999428| 19
Basic Daydreams| com.android.dreams.basic| 4.4.2-999428| 19
Development Settings| com.android.development_settings| 1.0| 1
Sample Soft Keyboard| com.example.android.softkeyboard| 4.4.2-999428| 19
Exchange Services| com.android.exchange| 4.4.2-999428| 500060
System UI| com.android.systemui| 4.4.2-999428| 19
Live Wallpaper Picker| com.android.wallpaper.livepicker| 4.4.2-999428| 19
Speech Recorder| com.android.speechrecorder| 4.4.2-999428| 19
Key Chain| com.android.keychain| 4.4.2-999428| 19
GPS Location Test| com.android.emulator.gps.test| 1.0| 1
Package installer| com.android.packageinstaller| 4.4.2-999428| 19
Dev Tools| com.android.development| 1.0| 1
com.android.smoketest.tests| com.android.smoketest.tests| 4.4.2-999428| 19
Phone/Messaging Storage| com.android.providers.telephony| 4.4.2-999428| 19
Pico TTS| com.svox.pico| 1.0| 1
Camera| com.android.camera| 1| 1
OpenWnn| jp.co.omronsoft.openwnn| 4.4.2-999428| 19
Email| com.android.email| 4.4.2-999428| 500060
Dialer| com.android.dialer| 4.4.2-999428| 19
Clock| com.android.deskclock| 3.0.0| 301
Fused Location| com.android.location.fused| 4.4.2-999428| 19
com.android.backupconfirm| com.android.backupconfirm| 4.4.2-999428| 19
Settings Storage| com.android.providers.settings| 4.4.2-999428| 19
com.android.keyguard| com.android.keyguard| 4.4.2-999428| 19
Shell| com.android.shell| 4.4.2-999428| 19
Download Manager| com.android.providers.downloads| 4.4.2-999428| 19

Abbildung A.1.: Prototyp von Anti-Cheat-Plus

A.2. App aus anderer App starten

Im Prototypen von Anti-Cheat-Plus wird die Auswahl „A” getroffen und daraufhin starten die Android-Einstellungen, die selbst eine App sind. Dies wurde mit Hilfe der Android-Virtuellen-Maschine überprüft. Ursprung ist Bild A.1 und Ergebnis ist Bild A.2.

Abbildung A.2.: Android Einstellungs-App



B. Abbildungsverzeichnis

C. Tabellenverzeichnis

D. Algorithmenverzeichnis

E. Literaturverzeichnis

[1] VAHLDIEK, Axel: Der öffentliche PC Windows narrensicher konfigurieren. In: *Heft 3* 1 (2011), Februar, Nr. 1, S. 8. – 1

[2] ; Microsoft (Veranst.): *Gruppenrichtlinien für Anfänger*. http://technet.microsoft.com/de-de/library/hh147307%28v=ws.10%29.aspx. Version: April 2011

[3] DITTBERNER, Jan: *Keytool, OpenSSL, und Co. Was nehme ich wofür und Warum?* de.slideshare.net/jandd/keytool-openssl-und-co-wofr-nehme-ich-was-und-warum. Version: Mai 2011

[4] OOSTEN, Erik van: *Securing connections with TLS*. http://blog.trifork.com/2009/11/10/securing-connections-with-tls/. Version: November 2009

[5] ; Oracle (Veranst.): *Java TM Secure Socket Extension (JSSE) Reference Guide*. http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html

[6] *Difference between trustStore and keyStore in Java - SSL*. http://javarevisited.eu01.aws.af.cm/2012/09/difference-between-truststore-vs-keyStore-Java-SSL.html. Version: September 2012

[7] ; Microsoft (Veranst.): *Chapter 6 - Digital Certificates*. http://technet.microsoft.com/en-us/library/dd361898.aspx

[8] ; Microsoft (Veranst.): *Description of the Secure Sockets Layer (SSL) Handshake*. http://support.microsoft.com/kb/257591/en-us. Version: Juli 2008

[9] *SHA-3 WINNER*. http://csrc.nist.gov/groups/ST/hash/sha-3/winner_sha-3.html. Version: Oktober 2012

[10] *Verwendung des Kioskmodus im Microsoft Internet Explorer*. http://support.microsoft.com/kb/154780/de. Version: Januar 2004

[11] ; Google (Veranst.): *AutoCompleteTextView*. http://developer.android.com/reference/android/widget/AutoCompleteTextView.html. Version: August 2014

[12] ; Google (Veranst.): *PackageManager*. http://developer.android.com/reference/android/content/pm/PackageManager.html. Version: August 2014

[13] ; Google (Veranst.): *Sensors Overview*. http://developer.android.com/guide/topics/sensors/sensors_overview.html

[14] CHAKRABORTY, Arnab: *Developing Kiosk Mode Applications in Android*. http://arnab.ch/blog/2013/11/developing-kiosk-mode-applications-in-android/. Version: unbekannt unbekannt

[15] STAFF: *How-To Create Kiosk Mode on the Nexus 7*. http://www.gokiosk.net/kiosk/gokiosk/2013/04/how-to-create-kiosk-mode-on-the-nexus-7.html. Version: April 2013

[16] ; Golem.de (Veranst.): *Hooks im Kernel sollen Android sicherer machen*. http://www.golem.de/news/intel-hooks-im-kernel-sollen-android-sicherer-machen-1403-105054.html. Version: Maerz 2014

[17] BASSOV, Anton: *Hooking the native API and controlling process creation on a system-wide basis*. http://www.codeproject.com/Articles/11985/Hooking-the-native-API-and-controlling-process-cre. Version: Oktober 2005

[18] SOMMERGUT, Wolfgang: *Windows 7 Enterprise: Alternativen zu AppLocker*. http://www.windowspro.de/wolfgang-sommergut/windows-7-enterprise-alternativen-zu-applocker. Version: Maerz 2010

[19] ; Infoworld.com (Veranst.): *Whitelisting security solutions by the features*. http://www.infoworld.com/node/98873

[20] HEITBRINK, Mark: *Gruppenrichtlinien Laufwerke im Explorer ausblenden und den Zugriff darauf verhindern*. http://www.gruppenrichtlinien.de/artikel/laufwerke-im-explorer-ausblenden-und-den-zugriff-darauf-verhindern/. Version: Januar 2013

[21] SOMMERGUT, Wolfgang: *Registry-Schlüssel mit Group Policy Preferences erstellen und löschen*. www.windowspro.de/wolfgang-sommergut/registry-schluessel-group-policy-preferences-erstellen-loeschen. Version: September 2013

[22] JOOS, Thomas: *So ändern Sie Registry-Einstellungen über Gruppenrichtlinien*. http://www.ip-insider.de/themenbereiche/administration/client-server-administration/articles/431445. Version: Januar 2014

[23] ; Google (Veranst.): *Camera*. http://developer.android.com/reference/android/hardware/Camera.html#lock%28%29. Version: August 2014

[24] RACIS, Frank: *How do I disable a system device programatically?* http://stackoverflow.com/questions/4097000/how-do-i-disable-a-system-device-programatically. Version: Juli 2011

[25] ; Google (Veranst.): *Package Index*. http://developer.android.com/reference/packages.html. Version: August 2014

[26] AFTAB, Wajahat K. b.: *Learning Android Intents Explore and apply the power of intents in Android application development*. Packt Publishing, 2014. – ISBN 978-1-78328-963-9

[27] *How to reset default launcher/home screen replacement?* http://stackoverflow.com/questions/15537522/how-to-reset-default-launcher-home-screen-replacement. Version: Maerz 2013

[28] GUY, Romain: *Android Simulate Home click*. http://stackoverflow.com/questions/2752319/android-simulate-home-click. Version: Mai 2010

[29] ; Microsoft (Veranst.): *Tastenkombinationen*. http://windows.microsoft.com/de-at/windows/keyboard-shortcuts#keyboard-shortcuts=windows-8

[30] HERMANN, Udo: *C Sharp Globale Hotkeys, welche wirklich global sind*. http://physudo.blogspot.de/2013/09/c-globale-hotkeys-welche-wirklich.html. Version: September 2013

[31] *android intercept recent apps button*. http://stackoverflow.com/questions/17769367/android-intercept-recent-apps-button. Version: Juli 2013

[32] ; Chip.de (Veranst.): *Windows 8 Shortcuts: Immer den richtigen Drücker*. http://www.chip.de/news/Windows-8-Shortcuts-Immer-den-richtigen-Druecker_58119153.html. Version: Juni 2013

[33] ; Microsoft (Veranst.): *Tastenkombinationen Windows 8.1 / Windows RT 8.1*. http://windows.microsoft.com/de-de/windows/keyboard-shortcuts#keyboard-shortcuts=windows-8

[34] ; Microsoft (Veranst.): *Verwenden von Gesten*. http://windows.microsoft.com/de-de/windows7/using-touch-gestures

[35] ; Hewlett-Packard (Veranst.): *Windows 8-Berührungssteuerung und -Tastaturbefehle*. www8.hp.com/de/de/support-topics/windows8-support/touch-gestures-keystrokes.html

[36] *How to detect when an Android app goes to the background and come back to the foreground*. http://stackoverflow.com/questions/4414171/how-to-detect-when-an-android-app-goes-to-the-background-and-come-back-to-the-fo. Version: - 2010 - 2012

[37] ; Google (Veranst.): *Tasks and Back Stack*. http://developer.android.com/guide/components/tasks-and-back-stack.html

[38] ; Google (Veranst.): *Tasks and Back Stack*. http://developer.android.com/guide/components/tasks-and-back-stack.html

[39] *Get installed Applications with Name, Package Name, Version and Icon*. http://www.androidsnippets.com/get-installed-applications-with-name-package-name-version-and-icon. Version: 2009

[40] *Open another application from your own (intent)*. http://stackoverflow.com/questions/2780102/open-another-application-from-your-own-intent. Version: Mai 2010

[41] ; Geolocation.com (Veranst.): *What is Geolocation? What are different type of Geolocation?* http://www.geolocation.com/. Version: - 2012-2014

[42] ; Google Project Hosting (Veranst.): *Java API for Google geocoder v3*. https://code.google.com/p/geocoder-java/

[43] GAVAGHAN, Mike: *Java Geodesy Library for GPS Vincentys Formulae*. http://www.gavaghan.org/blog/free-source-code/geodesy-library-vincentys-formula-java/. Version: April 2008

[44] *How permission can be checked at runtime without throwing SecurityException?* http://stackoverflow.com/questions/7203668/how-permission-can-be-checked-at-runtime-without-throwing-securityexception. Version: 2011-2013

[45] *Services*. http://developer.android.com/guide/components/services.html, 2014

[46] YOGESH KUMAR, Harsh S. Rajiv Munjal M. Rajiv Munjal: *Comparison of Symmetric and Asymmetric Cryptography with Existing Vulnerabilities and Countermeasures*. http://www.ijcsms.com/journals/Special%20Issue%20of%20Volume%2011_Comparison%20of%20Symmetric%20and%20Asymmetric%20Cryptography%20With%20Existing%20Vulnerabilities%20and%20Countermeasures.pdf. Version: Oktober 2011

[47] ; Heise Zeitschriften Verlag GmbH & Co. KG (Veranst.): *Androids Code-Signatur lässt sich umgehen*. http://www.heise.de/security/meldung/Androids-Code-Signatur-laesst-sich-umgehen-1911077.html. Version: Juli 2013

[48] ; Heise Zeitschriften Verlag GmbH & Co. KG (Veranst.): *Schutzlose Wächter - Antiviren-Software als Sicherheitslücke*. http://www.heise.de/security/meldung/Schutzlose-Waechter-Antiviren-Software-als-Sicherheitsluecke-2277782.html. Version: Juli 2014

[49] ; Valve (Veranst.): *Steamworks 2009*. http://www.steampowered.com/steamworks/SteamWorksBrochure2009.pdf. Version: 2009

[50] *C Sharp - lokale User und Gruppen administrieren*. http://dotnet-snippets.de/snippet/lokale-user-und-gruppen-administrieren/702. Version: November 2011

[51] ; Microsoft (Veranst.): *Asynchrones Client-Socket-Beispiel*. http://msdn.microsoft.com/de-de/library/bew39x2a%28v=vs.110%29.aspx

[52] ; Microsoft (Veranst.): *Asynchrones Server-Socket-Beispiel*. http://msdn.microsoft.com/de-de/library/fx6588te%28v=vs.110%29.aspx

[53] "WABILITYDE": *C Sharp | Dienst Programmieren und Installieren | Tutorial*. Video. https://www.youtube.com/watch?v=5v21fBSoL-Q. Version: August 2012

[54] NATHANSEN, Martin: *GPS-Genauigkeit und Einschränkungen*. http://gpso.de/technik/gpsgenau.html. Version: 2012