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	09.10.2014
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 Bachelorarbeit 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. Überblick

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 wird weiter technisiert, 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 wird 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. Insbesondere geht es, darum, wie verhindert wird, dass Schüler betrügen (Spicken) in Klassen- und Gruppenarbeiten (oder Testate, Prüfungszulassungen, etc.), die benotet werden können sollen, aber auch Randthemen Geo-Lokalisierung werden behandelt.

Um Betrugsversuche zu unterbinden, wird darauf eingegangen, welche digitalen Möglichkeiten zu Betrügen ein Schüler 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 nicht selbst korrigieren 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 der den leistungsschwächeren Schülern? Was für 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, 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 vor Betrug bei Leistungs-Überprüfungen durch einen Schüler und den zusätzlichen Verwaltungs-Aufwand, den ein Lehrer 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. Der Schutz vor Betrug in digitalen Leistungs-Überprüfungen bedarf 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 Betrug in digitalen Leistungs-Überprüfungen zu überwachen oder diese 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 Konfigurierung.

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 u.a., die Positionserkennung eines Gerätes auf der Erde, der Geo-Lokalisierung. Damit können Tablets eine vorher programmierte automatisierte Reaktion im Gerät ausführen, wenn sich der Schüler 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 6 gemessen, wie schnell die Netzwerkkommunikation vonstatten geht.

Das letzte Kapitel 7 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 Betrugs zulassen. Deshalb besteht das Gesamtkonzept aus mehreren einzelnen Bestandteilen. Die Sicherheitsvorkehrungen von Anti-Cheat-Plus begrenzen den Handlungsspielraum des Schülers (=„innen”), anders als Sicherheitslösungen wie Intrusion-Detection-Systeme und Firewalls, die vor äußeren Bedrohungen schützen. TLS schützt gewöhnlich vor Bedrohungen von außerhalb des benutzen Gerätes, wird in dieser Arbeit jedoch letztendlich zum Schutz vor Klausur-Betrug eingesetzt. Die automatische Erkennung des Klassenraumes, um zeitgesteuert Einschränkungen zu aktivieren und zu deaktivieren, kann nicht garantiert in jedem Fall 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

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

- digitale Betrugsmethoden ohne Einbeziehung von der nicht-technischen Umgebung
  - Von anderer Quelle Informationen gewinnen
    - Webdienste, andere Netzwerkdienste, Dateien, Dateinamen, beliebige Datenträger,
    - Verbindungen zum Tabletgerät
      - 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). 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 der dritten C't von 2011[48], einer deutschen Computerzeitschrift. In den Windowsversionen nach der 7 ist die Vorgehensweise ähnlich. Windows 8 / 8.1 ohne Enterprise im Namen sind Gruppenrichtlinien nicht konfigurierbar.

In [17] 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 Betrug, ist dazu da um in dem Netzwerk einer Schulklasse Nachrichten zu verarbeiten. Der Lehrer muss Befehle veranlassen. Die Tablets der Schüler 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 Schüler geprüft wird.

Der Schüler 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 das 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 Schüler über den Weg des (Funk-)Netzwerkes in Erfahrung bringen können. Wenn es Abweichungen vom Normalzustand bei einem Schüler 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 Schüler, 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 Schüler hat, die die anderen Schüler nicht haben. Das bedeutet, dass der Lehrer 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 Schüler 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 Schüler 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 Schüler heimlich gerootet wurde oder ein Jailbreak (eine Rechteerweiterung bzw. Entsperrung) installiert wurde.

Letztendlich muss der Schüler 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 dem Lehrer Aufgaben und Aufwand ab. Dadurch benötigt der Lehrer kein IT-Expertenwissen mehr wie in Unter-Abschnitt 2.3.1.

Es muss eine Liste von allen Programmen die es für die betreffenden Betriebssysteme gibt geführt und ständig aktuell gehalten werden oder alle Programme, die auf den Tablets installiert sind werden immer automatisiert von einer Organisation dokumentiert und bewertet in regelmäßigen Abständen. 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. Alle ausführbaren Dateien betrifft dies.

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 Maleware und Hackern.
Eine Liste von Programmen für das Anwendungsgebiet dieser Arbeit, muss unterscheiden in Programmen, 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. Programmiertestate im Informatikunterricht an Schulen. Das bedeutet, dass nicht nur Abstufungen denkbar sind zwischen Zulassen und nicht Zulassen, sondern auch um Zulassen und Nichtzulassen nach Art der Leistungsüberprüfung (Klassenarbeit , Gruppenarbeit, etc.).

Um den Lehrer zu entlasten ist es zielführend keine Abstufungen zwischen 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. nur für Klassenarbeiten.

Um automatisiert stattgefundenes Jailbreaking und Rooting und Manipulationen am System feststellen zu können, gibt es Methoden der forensischen Analyse (Einbruchserkennung). Das zu behandeln sprengt jedoch den Rahmen dieser 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-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 Schüler fair bleiben beim Testat (Abschnitt 4.7).

Verschlüsselte,integre und identifizierende Netzwerk-Verbindung:
Für den Netzwerk-Beobachtungs- und den fernbedienbaren Kioskmodus werden SSL-Sockets verwendet.

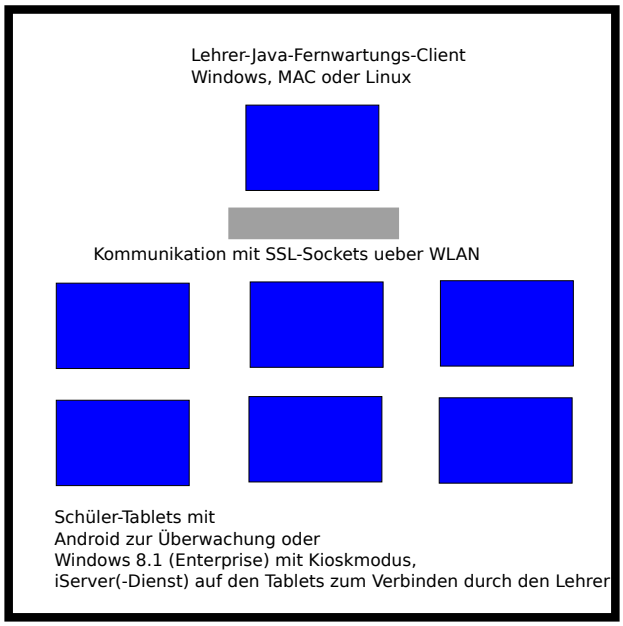
SSL-Sockets ermöglichen die Authentifikation sowohl des Servers- als auch des Clients, womit sichergestellt werden kann, dass nur der berechtigte Lehrer Fernwartungszugriff hat und dass der Lehrer die Tablets zugehörig zu den jeweiligen Schülern identifizieren kann. In Unterabschnitt 4.1.1 und Abschnitt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

Die Schüler können die SSL-Verbindungen zu anderen Schülern im Klassenraum wegen der Verschlüsselung nicht lesen. Außerdem bieten SSL-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 Schüler 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 Vorsorge für die Betrugsversuch-Erkennung und Definierung erlaubter Programme:

Abbildung 3.1.: Netzwerk-Schema im Klassenraum



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 Schüler 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 Schüler:
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:
Der Betrug mit manuell getätigten Verschlüsselungsverfahren, zum Verschleiern der Tat, dauert für eine Klassenarbeit zu lange, so dass die Zeit dafür eher zu schlechteren Resultaten des Schülers führt, als ohne den Einsatz dieser.

Grundlegendes Problem:
Insgesamt darf kein Sicherheitsproblem außer Acht gelassen werden, denn das bedeutet ansonsten, dass die ganze Sicherheitslösung nutzlos ist.

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. SSL-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.

3.1. Windows - Kiosk-Modus gegen Betrug

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 SSL/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 4.5.7. Deshalb ergänzt diese Arbeit was in Windows 8 anderes und weiteres zu tun ist.

3.2. Android - Tabletüberwachung gegen Betrug

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 Schüler 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 Betrug (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 zum Spicken verwendet werden können. In Abschnitt 4.5.1 , 4.5.2 werden Methoden beschrieben mit denen Schüler 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 Autovervollständigungsfunktionalität 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 Schülers vor unbeabsichtigtem Betrügen wegen seinen Gewohnheiten:
Abschnitt 4.6.1, 4.6.2, 4.6.4 behandeln das Unterbinden von Tasteneingaben, was den Schüler 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 Schüler nicht mit wenigen Fingerbewegungen die Einstellungen erreicht, was dem Lehrer als Betrugsversuch gemeldet wird.

Zusätzliche, nicht notwendige Blockade vor Betrugsversuchen:
Abschnitt 4.5.3, 4.5.5 und 4.5.10 behandelt das Deaktivieren von Bluetooth, der Kamera und Sensoren. Diese 3 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 Schüler 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 vom 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 Laufwer-

ke ausgeblendet, werden 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 Antispywareprogrammewerden 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
- Sound-Funktionalität für Ein- und Ausgang
- 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 von den Schülern 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 Schüler 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 Schüler 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 Schüler 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 Schüler nur Leserechte besitzt, aber der Lehrer zusätzlich Schreibrechte.

Anstelle eines Kioskmodus kann auch registriert werden, ob der Schüler 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 Schülers und der Client des Lehrers sowieso zueinander authentifizieren mittels TLS- / SSL-Sockets. Der Server muss bei sich TLS generell immer beim Client authentifizieren. Je nach Umsetzung, der Verwendung von Methoden der SSL-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 Schüler 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. SSL / TLS

Für dieses Kapitel sind Grundkenntnisse in Kryptologie bzw. Kryptographie erforderlich. SSL ist ein Verfahren, um über ein Netzwerk mittels Sockets sicher zu Kommunizieren. Ab Version 3.1 wurde SSL in TLS umbenannt, wobei die Bezeichnung SSL nicht obsolet geworden ist. 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 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 Schüler nur vom Lehrer-Client überwacht werden können, ist daher die programmiertechnisch optionale Authentifizierung des Clients in der Spezifikation von Anti-Cheat-Plus verpflichtend.

Der Schüler 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 Schüler mit diesem nicht betrügt. Jedoch schreibt der Schüler die Klassenarbeit mit dem anderen virtuellen oder realen Tablet und betrügt dabei, ohne dass der Lehrer dies auf elektronischem Weg bemerkt. Zwischen beiden Geräten herrscht eine Verbindung, so dass der Schüler 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 SSL / TLS die Authentifizierung automatisch mit Zertifikaten und Schlüsseln durchgeführt.

SSL / TLS wird für die Spezifikation in das Überwachungsprogramm, mit Zugriff auf Bibliotheksfunktionen, eingebaut. TLS / SSL 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 SSL / TLS als die Sicherheitsschicht von HTTPS, OpenVPN, einem Teil der Android-Apps. Als TLS- / SSL-Programmbibliothek, die dazu dient damit andere Programme SSL- / TLS-Sockets nutzen können, ist Openssl der bekannteste Vertreter und Keytool für Java von Oracle.

4.2.1. TLS / SSL mit Java

Für Java gibt es freie Implementierungen, um Kryptographie zu gewährleisten z.B. „Bouncy Castle" [1], u.a. für TLS bzw. SSL. Jedoch bietet auch die grundlegende API von Java Möglichkeiten TLS / SSL zu realisieren. Dabei liefert Oracle das Kommandozeilenprogramm keytool mit als Pendand zu openssl, jedoch mit geringerem Funktionsumfang und proprietären Formaten. Die Formate von openssl und keytool lassen sich ineinander umwandeln. Unter [32] 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

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

das Webseiten-Serverzertifikat von einer CA unterschreiben zu lassen, weil diese bei Ausklammerung der Problematik mit den Geheimdiensten eine vertrauliche Quelle sind. Für das Unterschreiben durch die CA 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.

4.2.2. TLS / SSL 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 [49] hat beim Erstellen dieser Arbeit zum Erfolg geführt. Es ist aber auch möglich allein mit der API-Referenz von Oracle TLS- / SSL-Sockets verwendbar zu machen [4].

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[19]. 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 / SSL 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

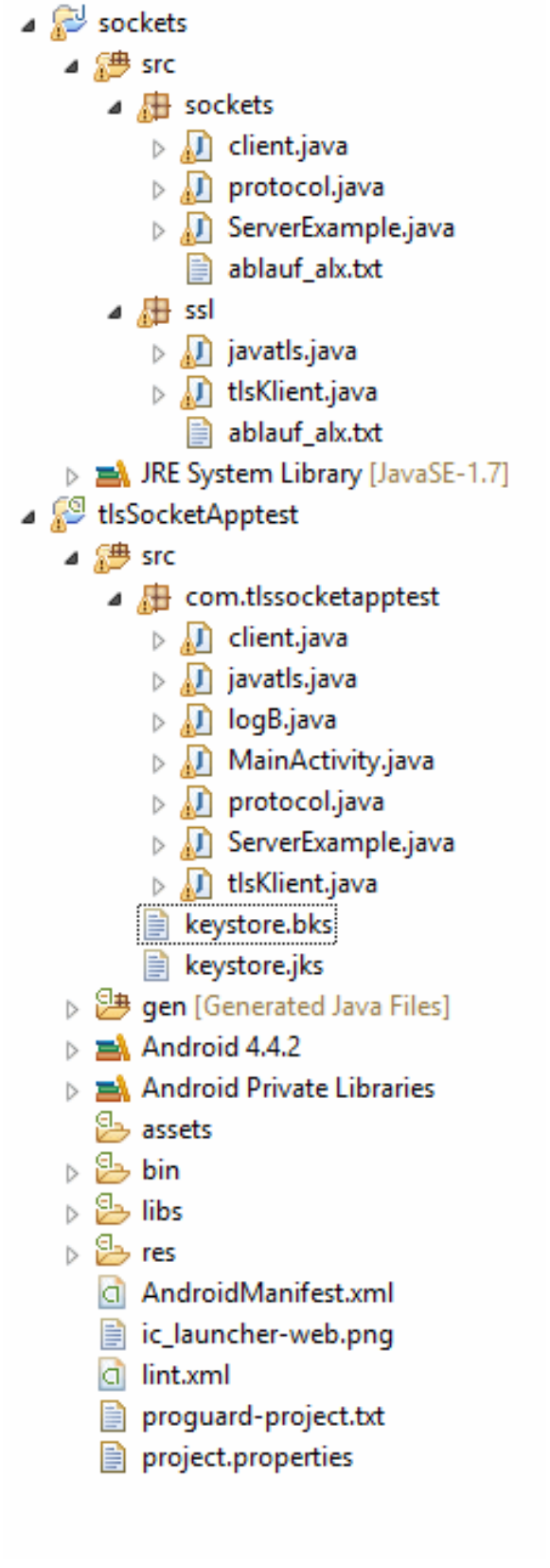


Tabelle 4.1.: SSL / TLS Dateiformate

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 [2] : 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 [32] und [35] finden sich verschiedene Formate, die für SSL 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 [32] 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 / SSL Handshake wird auf einer Seite von Microsoft ausführlich beschrieben: [35]
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 SSL- / 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 Contai-

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

nerdateien vorgesehen sind, und auf welcher Seite, Server oder Client. Im Quellcode 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 [20]. 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.

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 Schüler 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 können, da es sich nicht um einen Standard handelt und es keine weiteren Programme gibt, die dieses Protokoll einsetzen werden. ASCII wurde verwendet mit dem Ziel dass das Protokoll menschen-lesbar ist.

Damit das Protokoll sicher ist, müssen alle Eingabedaten 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 und oder verboten ist zur Abwehr von Exploits. Verschiedene Pufferüberläufe müssen abgefangen werden (z.B. Datentyp-Unter und Überläufe), die durch Socket-Eingabedaten hervorgerufen werden. Es muss überprüft werden, ob Eingabedaten be-

stimmte Systemfunktionen bedienen bzw. direkt oder über Umwege mit der Programmlogik erreichen können.

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. Asynchrone Sockets senden ein Event zu einem Eventhandler und können weiter Daten empfangen oder senden. Synchrone Sockets senden oder empfangen Daten und blockieren so lange, bis der Prozess zuendegeführt 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 [50] findet sich eine funktionierende Anleitung zum programmieren von Windows-Diensten. Ein Dienst muss kein 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.

Windowsdienste können mit noch höheren Rechten als Administratorrechten unter Windows ausgestattet sein, und zwar mit den Rechten des Benutzers „System”.

Ein Dienst bietet sich für Anti-Cheat-Plus an, weil Schüler 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 ein 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 Schüler 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 Schüler 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

ACID steht für Atomicity, Consistency, Isolation und Durability. Das sind 4 Eigenschaften (Atomizität, Konsistenz, Isolation , Dauerhaftigkeit), die relationale SQL-Datenbanken haben. Insgesamt nützen diese Eigenschaften der Verlässlichkeit. Es gibt Anwendungsbereiche ohne Datenbanken, in denen ACID-Eigenschaften wünschenswert sein können.

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. 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 Reperatur automatisch durchgeführt werden, damit Atomizität sichergestellt wird.

Atomare 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, denn 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.

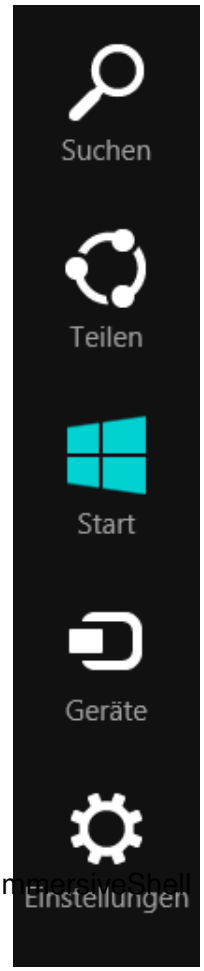
Versionen der Classic Shell[2] vor 3.9.0 sind quelloffen. Die Classic Shell ersetzt den Start-Button von Windows 8 / 8.1 mit dem Startmenü von Windows 7, das komplett nachgebildet wurde, und nicht versteckt in Windows 8 / 8.1 vorhanden ist. Dazu muss auf die API der Taskleiste zugegriffen werden und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

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

4.4.1. Deaktivieren der Charmsleiste (Windows)

Abbildung 4.2.: Charmsleiste



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 Schüler von da aus die Konfigurationen von Windows ändern , nach Dateien suchen und Dateien mit anderen teilen kann, und somit in Klassenarbeiten betrügen 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 beim Verändern von Registryeinträgen und dem Eintreten deren Wirkung.

4.4.2. Kioskmodus des Internet Explorers

Der Internet Explorer hat einen eingebauten Kioskmodus, siehe [50] .
Allein dieser Modus funktioniert noch nicht als Betriebssystem-Kioskmodus. Deshalb muss man den Kiosk-Modus des Internet-Explorers kombinieren mit dem des Betriebssystems. Ansonsten kann man z.B. einen weiteren Brow-

ser installieren oder anderweitig die Sicherheitsvorkehrungen umgehen. In dieser Arbeit wird nicht näher auf den Kioskmodus des Internet Explorers eingegangen, da das zu dem Thema über Webseitensperrungen gehört.

4.5. Einzelne Sperrungsbereiche

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 Betrug. Deshalb muss die Zwischenablage in dieser Arbeit thematisiert werden.

Für Windows ist die Zwischenablage nicht relevant, weil sich Schüler neu einloggen 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.

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

Algorithmus 4.1 Android Zwischenablage leeren

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

nem 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 ist so eine erweiterte Zwischenablage ein Sicherheitsrisiko, mit der Schüler in Klassenarbeiten betrügen können.

4.5.2. Formularfelder mit Autovervollständigung unter Android

Abbildung 4.3.: Autocomplete

Unter [25] wird beschrieben, wie die Autovervollständigung (englisch autocomplete) unter Android implementiert wird.

Wird in einem Programm unter Android eine Klassenarbeit geschrieben oder an einer Gruppenarbeit gearbeitet, dann ist es möglich, dass Schüler schummeln, indem Sie vorher Wörter in der Autovervollständigung speichern, durch vorheriges tippen und übernehmen, und wenn die Prüfung geschrieben wird, auf diese vorhandenen Eingaben zurückgreifen. Administratoren, Lehrer und 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 gelöscht werden.

Es ist heutzutage so, dass es Apps gibt, die in HTML usw. geschrieben sind und trotzdem wie andere Apps aufgerufen werden mit dem Programmstart. 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 alleine Java-(Byte)-Code untersucht werden darf.

Es ist möglich, eine ausführbare Datei unterschiedlicher Art (*.exe / Bytecode / Skript) analysieren zu lassen, welche Betriebssystemfunktionen sie verwendet. Das geht mit der Methode, mit der Antivirenprogramme Malware finden. Das bedeutet, jedes Antivirenpro-

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 }
```

gramm 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, mit denen betrogen werden kann.

Die Programmierung eines Parsers ist nicht Teil dieser Arbeit, weil das den Rahmen sprengt. Es wird darauf verwiesen, dass Compiler- und Interpreterbau Grundlagen zum Parsen enthält, weil das ein Bestandteil von beidem ist.

4.5.3. Bluetooth unter Android

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

Der Schüler kann Bluetooth wieder aktivieren, von daher macht es Sinn, den Status darüber, ob Bluetooth aktiviert ist, dem Lehrer-Computer mit dem Überwachungsprogramm zu melden, in Abständen von z.B. 5 Sekunden. 5 Sekunden reichen nicht für einen Betrug aus. Selbst wenn der Schüler etwas gescriptet 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 dem Überwachungsprogramm dieser Arbeit gesendet wird, welche Apps aktiv sind.

Nicht nur für Bluetooth relevant, aber hier [28] ist beschrieben, wie man abfragt, ob ein Android-Gerät überhaupt GPS oder andere Features unterstützt.

Algorithmus 4.2 auf der vorherigen Seite „Bluetooth unter Android”:
In der ersten bis zweiten Zeile wird das Objekt besorgt, das den Bluetooth-Adapter verwaltet. In der dritten Zeile wird mit isEnabled() abgefragt, ob der Bluetooth-Adapter aktiviert ist oder nicht. Toast ist ein Objekt zum Ausgeben von Nachrichten. „mBluetoothAdapter.disable();” deaktiviert den Bluetooth-Adapter in Zeile 7.

Bluetooth muss deaktiviert werden, weil es die Möglichkeit bietet, dass Daten von externen Geräten empfangen und zum Tablet des Schülers 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 die Überwachungs-App die diese Arbeit behandelt deaktiviert wurde, wie in diesem Abschnitt beschrieben.
Es ist nützlich den Bluetooth-Status, ob aktiviert oder deaktiviert, in Zeitabständen dem Lehrer-Überwachungs-Computer zu melden, um Betrugsversuche auszuschließen.

4.5.4. Websperren

Webseitensperren sind nicht das Thema dieser wissenschaftlichen Arbeit, da dies zu weit führt. Eine wissenschaftliche Hausarbeit kann sich damit beschäftigen. Es ist davon auszugehen, dass das Thema schon historisch gesehen ausgeschöpft ist.

Jedenfalls werden Websperren benötigt sobald der Schüler einen Browser verwenden darf, jedoch nicht alle Webseiten ansteuern darf. Auch für den regulären Unterrichtsbetrieb machen Websperren Sinn, da ein Tablet ansonsten kontraproduktiv für den Unterricht ist, weil ein Teil der Schüler abgelenkt wird vom Unterricht. Selbst ein Teil der erwachsenen Menschen und Lehrer wird durch Computer zu einem Teil abgelenkt von der Arbeit und brauchen deshalb bewusste Selbstbeherrschung.

Insofern haben Websperren im Kontext dieser Arbeit zumindest eine Bedeutung!

4.5.5. Sensoren unter Android

Unter [5] ist die detaillierte Übersichtsseite über Sensoren für Entwickler von Android. Dort ist beschrieben, was Android alles für Sensoren theoretisch und praktisch unterstützt. 13 Typen sind es zum Zeitpunkt dieser Arbeit (2014).

Wie kann ein Schüler z.B. mit einem Gravitations-, Temperatur-, Luftdruck-, Luftfeuchtigkeitssensor usw. in einer Prüfung betrügen? Die Methoden Informationen über Sensoren in das Tablet zu schleusen, z.B. durch bewusst erzeugte Temperaturschwankungen, ist Unsinn, und deshalb ist es nicht nötig Sensoren zu deaktivieren, um Betrugsversuche zu unterbinden. Außerdem sind sowieso nur die Applikationen erlaubt, die für eine Klassen-

Algorithmus 4.3 Android-Status-Leiste verstecken [47]

/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>
```

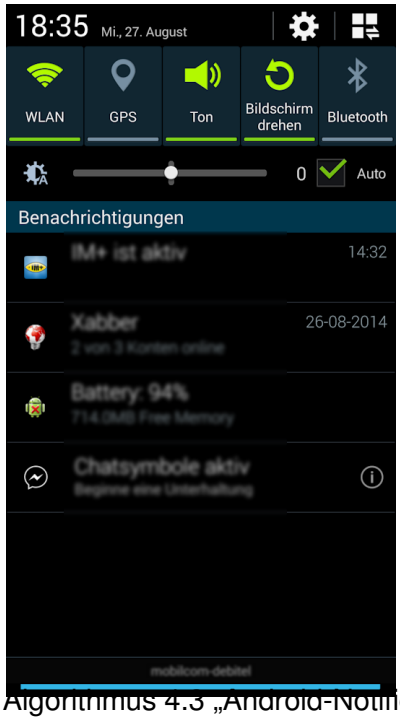
Verlinkung im Manifest:

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

oder Gruppenarbeit benötigt werden und keine für die Verwendung von Sensoren.

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

Abbildung 4.4.: Notification-Drawer



Die Status Leiste (Status Bar) in Android ist eine meist schwarze Leiste oben, die man mit einem Wisch von oben nach unten öffnet, so dass ein Vollbild-Menü erscheint.

Unter [31] 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 [47] 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-Notification-Drawer bzw. Status-Bar":
„android:windowFullscreen" (Zeile 5) erklärt das Theme 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. "andro-

id: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 erreichbar sind. Damit kann der Schüler das Gerät konfigurieren was in einer Klassenarbeit zu Betrugsversuchen führen kann. Es soll nicht möglich sein, dass der Schüler auch ohne Absicht einen Betrugsversuch begeht.

4.5.7. 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 [27] „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, was zur Folge haben kann, 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 kein Prozess im Kernel-Mode, im Gegensatz zu einem Programm im User-Mode. Unter [30] ist beschrieben, wie so ein Treiber für Windows XP programmiert werden kann. Darin wird empfohlen die Funktion NtCreateSection() 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. Für jede Version von Windows muss ein Teil der Treiberprogrammierung neu erlernt werden. Ein technischer Informatiker mit der Spezialisierung Treiberprogrammierung für Windows und ggf. Assemblerprogrammierung kann in der Lage sein so einen Treiber zu entwickeln.

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, jedoch lässt sich das Ausführen nicht verhindern. Es kommt infrage Quellcode von Antivirenherstellern zu lesen für die Funktionalität ausführbare Dateien blocken zu können. ClamWin kommt dafür infrage. Jedoch kann ClamWin keinen Livescan machen . Das muss ein technischer Informatiker mit Spezialisierung machen.

Da die Aufgabenstellung der Arbeit dazu führt, dass ein Kioskmodus entwickelt werden muss, liegt es nahe dass alle Bestandteile vom Autor entwickelt und / oder spezifiziert werden, es sei denn Teile sind frei verfügbar. In dem Fall des beschriebenen Treibers ist das jedoch nicht möglich, weil das den Rahmen der Masterarbeit sprengt und es ansons-

ten nur proprietäre Lösungen für die Funktionalität gibt.

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 (Funktionsreichtum 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)

[45]
Unter [10] werden die Produkte miteinander verglichen. SignaCert Enterprise Trust Services unterstützt kein Whitelisting und Blacklistung und kommt damit nicht infrage. CoreTrace Bouncer gibt es nicht mehr, da die Firma von Lumension aufgekauft wurde. Ingesamt sind die Alternativen zum Applocker von Windows 8 / 8.1 Enterprise funktionsreicher.

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

- Sperren von ausführbaren Dateien, nicht nur exe, sondern auch .bat .msi usw.
- Sperren und Erlauben mittels Pfaden, Herstellern von Exe-Dateien über Signaturen und Hashfunktionen von ausführbaren Dateien
- benutzerspezifisches und gruppenspezifisches Sperren
- Active-Directory-Unterstützung und lokale Benutzer

In der in der dritten C't von 2011 ist beschrieben, wie vorzugehen ist beim Sperren für Windows 7 Pro ohne Applocker. Der Windowsordner und der Programmeordner ist empfehlenswert zu erlauben. Es ist zwar sicherer Programme über Hashfunktionen zu erlauben, aber bei einem Update der ausführbaren Datei muss neu konfiguriert werden und der Hash neu eingelesen werden, was im Schulbetrieb nicht zumutbar ist für den Lehrer. Ein Administrator muss sich also darum kümmern, ggf. lässt sich das teilweise oder vollständig automatisieren. In dieser C't wird auch angedeutet wie man Bereiche des Dateisystems unsichtbar und unbetretbar macht für den Kioskbenutzer, so dass z.B. der Explorer nicht verboten werden muss. Zur Vollständigkeit wird das hier gezeigt: [36] Bei Windows 8.1 lautet ein Unterabschnitt jedoch nicht Windows Explorer sondern Datei-Explorer in den Gruppenrichtlinien. Also um Datenträger zu verstecken: Gruppenrichtlinien: Benutzerkonfiguration \ Administrative Vorlagen \ Windows-Komponenten \ Datei-Explorer

"Diese angegebenen Datenträger im Fenster "Arbeitsplatz" ausblenden"
"Zugriff auf Laufwerke vom Arbeitsplatz nicht zulassen"
Des Weiteren muss in den Gruppenrichtlinien noch unterbunden werden, dass externe Datenträger angezeigt werden. In der dritten C't von 2011 werden mehrere relevante Orte in den Gruppenrichtlinien genannt, wo sich auch diese Einstellung dafür auffindet.

Laut [46] und [39] 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, aber dann muss trotzdem mit Betriebssystemfunktionen gearbeitet werden. Das ist am Ende aufwändiger als einen Software-Treiber zu programmieren und es gibt die Gefahr, dass nicht alle Wege gesperrt sind. Z.B. sind Maus- und Touchgesten unter Windows 8.1. in Registryschlüsseln des Herstellers vermerkt, wodurch das Sperrren dieser für jedes einzelne Hardware-Eingabe-Gerät extra betrachtet werden muss. Wenn ausführbare Dateien nicht gesperrt werden und man dennoch einen Kioskmodus realisieren will, muss die Taskleiste in ihrer Funktionalität reduziert werden. Dazu muss auf Windows-DLLs wie user32.dll zugegriffen werden.
Die Einträge in den Gruppenrichtlinien sind zu einem Teil Registry-Einträge. Um herauszufinden, wie groß dieser Teil ist, oder ob gar alles nur Registryeinträge 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 Schülern 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.

4.5.8. Leeren des Startmenüs (Windows)

In der in der dritten C't von 2011 [48] ist beschrieben wie man das Startmenü von Windows 7 leer macht. In Windows 8.1 gab es Änderungen, weshalb dieser Vorgang aufwändiger ist als unter Windows 7. Das Prinzip ist es, den Ordnern, die das Startmenü verwendet die Rechte zu entziehen, die von dem Kioskbenutzer gelesen werden können, bzw. den Vollzugriff zu blockieren. Dazu müssen ausgeblendete Ordner und Laufwerke angezeigt werden. In diesem Fall nennt sich der Kiosk-Benutzeraccount „kiosk". Beim Ändern der Dateisystem-Rechte muss Vererbung aktiviert sein. Die Ordner sind folgen-

de:

- 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 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.

Ggf. kann noch das Verzeichnis des Windows-Desktops lese- und schreibgeschützt werden. Dazu zählt derjenige des Bentutzers als auch der öffentliche Ordner, der Dateien enthält die in der Regel jeder Benutzer lesen und ausführen kann. Der Datei-Rechte-Besitzer des Desktop-Ordners muss geändert werden, weg vom Kioskuser zu dem dieser Ordner gehört, damit der Kiosk-Benutzer nicht wieder die Rechte erlangen kann, durch Änderung der Rechtevergabe, den Desktop-Order zu lesen und zu beschreiben.

4.5.9. Registry-Manipulation, um Menü von Alt-Stgr-Entf zu leeren

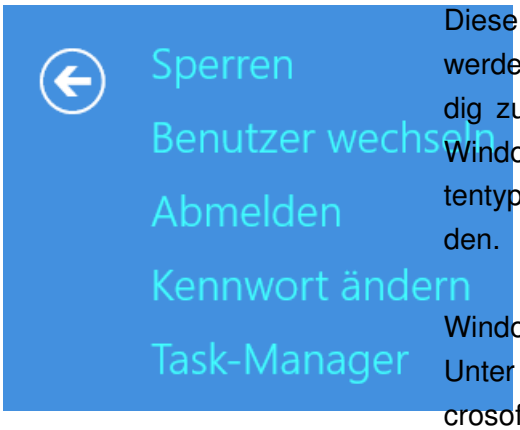


Abbildung 4.5.: Alt.-Strg.-Entf. Menü

Diese Einträge in der Registry müssen getätigt werden, um das aufkommende Menü vollständig zu entleeren. Es gibt Abweichungen von Windows 8 zu Windows 8.1. Alle sind vom Datentyp DWORD und müssen 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 Adminrechte 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 Adminrechte in Windows 8 änderbar.

4.5.10. Kamera (Android)

Unter [26] 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.

4.5.11. Bluetooth und sonstige Geräte unter Windows

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

Das programmiertechnische Aktivieren und Deaktivieren von Geräten unter Windows fällt ansonsten in den Bereich der technischen Informatik. Deshalb wird das hier nicht vertieft. Unter [44] ist ein Ansatz beschrieben, wie Geräte aktiviert und deaktiviert werden können. Der gleiche Quelltext kann aber unter unterschiedlichen Systemen funktionieren oder nicht funktionieren. Dabei ist zu beachten, dass beim Aktivieren von Gerätschaften weitere Geräte auftauchen, die wiederum aktiviert werden können, weil sie von den vorangegangenen abhängen. Es gibt mindesten 2 verschiedene Arten mit denen Geräte bezeichnet werden in Windows, einmal menschen-verständlich und einmal kryptisch. Je nach Hersteller variieren die Bezeichnungen. Deshalb muss mit Textsuche oder regu-

lären Ausdrücken gearbeitet werden und z.B. nach der Groß- und Kleinschreibung des Wortes „Bluetooth" gesucht werden oder es wird tablet-gerät-spezifisch, wenn es nur ein Rechnertyp (Tablettyp) gibt, nach dem richtigen Namen des Bluetooth-Geräts gesucht.

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. Ein technischer Informatiker mit Spezialisierung auf Windows-Geräte (bzw. Bestandteilen von Computern) kann dieses Problem lösen.

Im letzten Text-Abschnitt von Unterabschnitt 4.5.3 auf Seite 39 „Bluetooth unter Android" wird begründet, warum Bluetooth deaktiviert werden muss. Für sonstige Geräte gilt das gleiche.

4.6. Tasten sperren unter Windows 8.1 und Android

4.6.1. Android

Unter [12] ist die offizielle Dokumentation zur Android-Programmierung 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 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 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, gibt es kein Recents-Menü, weder durch langes Drücken auf Home noch durch eine nicht existierende Recents-Taste. Wenn alles andere nicht hilft, ist es möglich den Android-Quellcode von Google zu besorgen und diesen zu verändern. Damit ist es möglich Android vollständig zu verändern, oder zu lesen wie Google Android umgesetzt hat.

4.6.2. Die Hometaste (Android)

Die Hometaste kann auf eine andere Art in ihrer Wirkung verändert werden, als oben in Un-

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>
```

terabschnitt 4.6.1 auf Seite 47 beschrieben ist. Der Homescreen wird aufgerufen durch die Hometaste. Es gibt Launcher-Apps, die diesen Ersetzen können als Startzentrale für Apps oder 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 Schüler 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.

Abbildung 4.6.: Home-Taste

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 dies auch ermöglichen durch explizites Festlegen in den Androideinstellungen, 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 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. Intents sind Verfahren zur Interprozesskommunikation innerhalb von Android. Dazu zählen auch Broadcasts in Android. 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.[43] ist ein Buch zum Thema Intents.

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

Unter [23] ist beschrieben, wie es mittels eine 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 [34] erklärt. Wenn nun in einem Menü ausgewählt wird, welcher der nächste Homescreen sein wird, dann müssen die XML Dateien, wie unter [23] beschrieben, geändert werden und der Java-Code von [23] bei Menüauswahl ausgeführt werden. Darauf folgt schließlich die Simulation des Drückens auf die Hometaste, beschrieben unter [34]. 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: [7].

Unter [37] 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 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 windowseigene 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 Standardentwurfsmustern 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+Ent
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. 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 nicht blockierbar. Wenn die App, die für den androideigenen 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. 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 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 nach seinem Entstehen wieder zu schließen[21] mittels Intents. 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.

Auf den Algorithmus 4.5 auf der nächsten Seite wird nicht näher eingegangen, da das Problem auf diese Art ungenügend gelöst wird.

4.6.5. Gesten, Shortcuts und programmierbare Tasten in Windows 8.1

Abbildung 4.7.: Recents-Taste

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

- wichtige: [24]
- alle: [8]

Gesten werden hier erklärt:

- [9]

Weitere Eingabemethoden werden hier erklärt:

- [11]

Alle Quellenangaben in diesem Abschnitt hier über Eingabemethoden sind relevant, weil mit Ihnen das Risiko verringert werden kann, dass Schüler Funktionalitäten des Betriebssystems zum Schummeln nutzen. Hier sei aber erwähnt, dass wenn Schüler sich neu einloggen müssen mit einem Kiosk-Account mit beschränkten Rechten das Problem dieses Risikos gelöst ist. 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.

Algorithmus 4.5 Recents-Taskmanager „toggeln”

```
1  @Override
2  public void onWindowFocusChanged(boolean hasFocus) {
3    super.onWindowFocusChanged(hasFocus);
4    if (!hasFocus) {
5      windowCloseHandler.postDelayed(windowCloserRunnable, 0);
6  } }
7  private void toggleRecents() {
8    Intent closeRecents =
9     new Intent("com.android.systemui.recent.action.TOGGLE_RECENTS");
10   closeRecents.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK
11     | Intent.FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS);
12   ComponentName recents =
13     new ComponentName("com.android.systemui",
14       "com.android.systemui.recent.RecentsActivity");
15   closeRecents.setComponent(recents);
16   this.startActivity(closeRecents);
17 }
18 private Handler windowCloseHandler = new Handler();
19 private Runnable windowCloserRunnable = new Runnable() {
20   @Override      public void run() {
21     ActivityManager am = (ActivityManager)
22       getApplicationContext().getSystemService(
23         Context.ACTIVITY_SERVICE);
24     ComponentName cn = am.getRunningTasks(1).get(0).topActivity;
25     if (cn != null &&
26  cn.getClassName().equals(
27   "com.android.systemui.recent.RecentsActivity"))
28     {
29       toggleRecents();
30     }
31   }
32 }
```

Um Gesten global zu deaktivieren, ist es manchmal 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 Schüler sich neu anmelden muss, in den Kiosk-Benutzer-Account.

Schulen sollten 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 sollen.
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 festgestellt, dass die Sperrung von Gesten, Shortcuts und programmierbaren Tasten als Sicherheitsschicht nicht vollständig möglich ist, das jedoch wegen der Sperrung von ausführbaren Dateien kein Problem darstellt.

4.7. Beobachten statt Sperren (statt Kioskmodus)

Zu Windows wird in dieser Arbeit beschrieben, wie ein Kioskmodus realisiert wird, damit Schüler mit dem Gerät nicht bei einer Klassenarbeit betrügen können. Es ist jedoch ausreichend den Schüler insofern zu überwachen, dass gemeldet wird, wenn er betrügt. Für Android wird daher die andere Methode beschrieben, wie Betrugsversuche erkannt werden können, statt rein gerätebasiertes Betrügen zu unterbinden.

4.7.1. aktive Prozesse und Dienste auflisten in Android

Ein Schüler kann u.a. damit betrügen ein anderes Programm zu öffnen, das er nicht öffnen soll. Dieser Be-

Algorithmus 4.6 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 }
```

trugsversuch muss dem Lehrercomputer über das Netzwerk gemeldet werden, damit in Klassenarbeiten faire Bedingungen herrschen. Ein Betrugsversuch kann auch vorligen wenn ein Programmbestandteil eines Programms geöffnet wird, jedoch muss eine weitere wissenschaftliche Arbeit geschrieben werden damit das festgestellt werden kann und es macht mehr Sinn direkt nur Programme zu erlauben und zu verbieten. Damit der Schüler nicht unbeabsichtigt in Versuchung gerät andere Apps zu starten, gibt es einen extra Launcher (Home-App), in dem nur erlaubte Programme gestartet werden können, so dass es lediglich umständlich wird verbotene Apps zu starten. Über eine Socket-Verbindung wird alle 5 Sekunden dem Lehrer gesendet, welche App gerade aktiv ist, also diejenige die der Schüler sieht. Ggf. wird auch gesendet, welche anderen Apps und Dienste insgesamt gestartet sind.

Hier [16] wird erklärt, wie abgefragt wird, welche App gerade im Vordergrund ist, die der Schüler in dem Augenblick sehen kann. Im Quelltext zu dieser Arbeit gab es dazu Änderungen, deshalb siehe Algorithmus 4.6!

Android-Tasks sind etwas anderes als Windows Tasks. Sie beinhalten Activitys (Deutscher Plural des Wortes) aus Apps und arbeiten mit dem Stapel aktiver Apps, siehe: [6] . Um auf Tasks zugreifen zu können, wird das Recht android.permission.GET_TASKS benötigt, das in der Manifestdatei anzugeben ist.

Erklärung zu Algorithmus 4.6 „aktive Prozesse und

Dienste auflisten in Android”:

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 Programme in einem Objekt verlinkt, in Zeile 5 die Dienste und in Zeile 6 die Android Tasks, die hier [40] 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 Activities. Activities sind Bestandteile von Apps. In 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 4.7 auf der nächsten Seite „aktive Prozesse und Dienste auflisten in Android”:

In Zeile 10 wird die Activity besorgt, die für den Benutzer aktuell sichtbar ist auf dem Gerät.
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

Hier [13] wird beschrieben wie man die installierten Apps gelistet bekommt.

Hier [15] 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 „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 „Die Hometaste (Android)” wird am Ende beschrieben, wie jede App zu einem Homescreen werden kann. Das ist nötig, damit Schüler nicht den Standard-

Algorithmus 4.7 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 4.8 Starten einer App aus einer anderen, ein Beispiel

```
1 RunApp("com.android.settings");
```

launcher nehmen, mit dem sie ohne Absicht Apps starten können, die als Betrugsversuch zählen, was dem Lehrercomputer gemeldet wird.

Es wurde programmiertechnisch überprüft, ob das alles in den Links funktioniert, weil die Quellen nicht „hochwertig" sind. Es ist aber anzumerken, dass einzelne Antworten auf den Seiten bewertet werden, so dass dies ein Hinweis ist, ob die Lösung funktioniert.

Dem Lehrer muss nicht gemeldet werden, ob der Launcher wirklich gewechselt wurde durch den Schüler. Für den Schüler 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 4.8 „extra Launcher für Android" steht in Klammern der Name des Pakets das zur App gehört, wodurch Android weiß, welche App gestartet wird, mit dem Aufruf RunApp.

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 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 dem Erd-Ellipsoid, dem intrasolaren Planet Erde.

4.8.1. 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 erprobt auf einem Android-Gerät und einem Eclipse-

Algorithmus 4.9 LocationManager

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

Algorithmus 4.10 Rechte zum Lokalisieren in der Manifestdatei

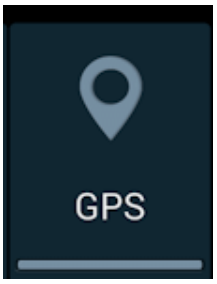
```
1 <uses-permission
2 android:name="android.permission.ACCESS_FINE_LOCATION" />
3 <uses-permission
4 android:name="android.permission.INTERNET" />
```

Projekt.

In einer Activity bekommt man mit dem Algorithmus 4.9 „GPS mit Android” 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.

Abbildung 4.8.: GPS an / aus



Im Algorithmus 4.11 „GPS mit Android” in Zeile 7-8 werden die Koordinaten Latitude und Longitude besorgt. Die geerbt-überschreibende Methode onLocationChanged wird immer dann automatisch aufgerufen, wenn das Andr̨oidsystem 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.

Das Problem an GPS ist, dass es zwar an Fenstern und auf der Straße in Sekunden

Algorithmus 4.11 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 }
```

den Standort anzeigt, aber bei Überdachung nicht, oder nach Minuten. Deshalb hat es einen Vorteil auf hier beschriebene alternative Möglichkeiten zurückzugreifen, den Standort zu bestimmen.

4.8.2. IP-Geo-Lokalisierung

Über die IP-Adresse ist der Internetanbieter / ISP / Provider lokalisierbar, der in mehr als 50% der Fälle im selben Land liegt, wie der Internet-Nutzer, aber nicht immer. 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 [42] wird erklärt, wie man mit Java IP-Geo-Lokalisierung umsetzt. Dabei wird die GeoLite-Datenbank verwendet.

4.8.3. über Daten von Datenbanken und Kombinationen

Ohne GPS kann ggf. auch 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: [42]

Hier [3] ist eine Java-Programmbibliothek, mit der Lokalisierungen umgesetzt werden können.

Hier [12] wird erklärt, wie Javascript der Standort herausgefunden werden kann.

4.8.4. Distanzen in Meter umwandeln, aus Erdkoordinaten

Um beliebige Distanzen aus Deltas von Latitude und Longitude (Erd-Koordinaten) zu berechnen, ist es sinnvoll auf Programmbibliotheken [33] zurückzugreifen.

Die Java-Quellcode-Dateien der Bibliothek[3] [33], kann man in sein Projekt ziehen, anstelle sie als *.jar einbinden zu lassen.

Algorithmus 4.12 auf der nächsten Seite „Distanzen in Meter umwandeln, aus Erdkoordinaten”:

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

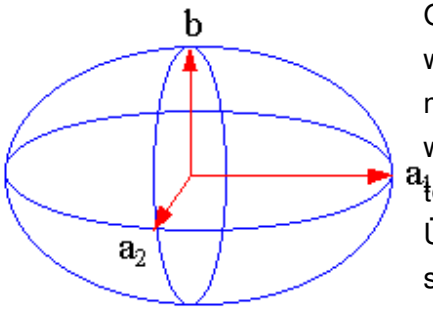
Algorithmus 4.12 Distanz in Meter, aus Weltkoordinaten

```
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 }
```

In Zeile 4 wird der Typ des Ellipsiod 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 weiten Messung an einem anderen Ort zu einer anderen Zeit gemessen wurde. In Zeile 9-10 wird schließlich die Distanz in Metern berechnet, zwischen beiden Messpunkten.

Distanzen-Berechnung wird benötigt, damit Schul-Räumlichkeiten lokalisiert werden können. Dabei muss der Raum mit einem Android-Gerät per GPS ausgemessen werden, so dass danach Tablets schon kurz vor dem Raum feststellen können, dass dieser Raum in der Nähe ist.

Abbildung 4.9.: Ellipsoid



4.8.5. Verwendung von Geo-Lokalisierung

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

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 Schüler 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 Schüler 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 Schüler schaltet das Netzwerk oder GPS aus, oder es gibt unvorhergesehene technische Defekte.

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

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 und aus dem gesammelten Wissen ein Programm zu entwickeln. In Java geschriebene Programme liegen in der Regel in der App in Bytecode vor. Bytecode ist leichter menschen-lesbar als Assembler. Es ist möglich, Bytecode zu decompilieren. Decompilierter Bytecode in Java ist leichter lesbar als decompilierter Assemblercode in z.B. C. In decompiliertem Bytecode in Java sind Methodennamen zwar nicht mehr rekonstruierbar, wohl aber Methodenaufrufe zu Java-Programmbibliotheken.

Warum ist Reverse Engineering ein Problem? Ein Schüler 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 Schüler Informationsquellen zu nutzen, die in einer Klassenarbeit nicht zugelassen sind.

Deshalb darf der Quellcode der spezifizierten Applikation(-en) dieser Arbeit nicht in die Hände eines Schülers geraten. Insofern macht es Sinn, den Code so zu lizenzieren, dass er entweder generell nicht einsehbar ist oder nur für einen speziellen Personenkreis z.B. für Personen die eine Graduierungsarbeit verfassen. Die entgeltliche oder unentgeltliche Verfügbarkeit dieser Applikation(-en) zu dieser Arbeit spielt hierbei keine Rolle.

Wenn Schüler diese Arbeit einsehen können, wird es für sie damit leichter die Programm(-e) zu dieser Arbeit zu verstehen, als ohne dieses Wissen. Das kann schließlich auch dazu führen, dass sie in digitalen Klausuren unbemerkt betrügen können. Aus diesem Grund macht es Sinn diese Arbeit nicht für die Bibliotheken und Buchhändler einsehbar und verfügbar zu machen.

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. 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 SSL-Bibliothek.

Neben der TLS- / SSL-Schicht empfiehlt sich als weiteren Schutz die Verwendung eines VPNs oder anderen Sicherheitsschichten inklusive Authentifikation. Mit einem VPN können die Schüler nicht untereinander Daten austauschen, denn das lässt sich unterbinden! Außerdem wird den Schülern damit eine weitere Hürde auferlegt den Betrugsversuchsschutz zu untersuchen, weil die Datenpakete für einen Netzwerkscanner nicht im Klartext vorliegen. Allerdings ist es möglich, dass der Schüler auf Zertifikate und Schlüssel 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. „Möglichkeiten zum Umgehen und Durchbrechen von Rechte- und Funktionsentzug" bzgl. „Honeypots" ist eine Methode beschrieben, wie ein Schüler in Klausuren betrügen kann, wenn er u.a. Zugriff auf Zertifikate hat.

5.3. Nachträglicher Entzug von Berechtigungen einer Applikation (Android)

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

Es gibt folgendes Problem. Wenn der Überwachungs-App zu dieser Arbeit, Rechte entzogen werden, kann sie nicht mehr korrekt überwachen, ob der Schüler betrügt. Man braucht keine Administrator-Rechte, um Programmen Rechte zu entziehen. Deshalb muss in der App nocheinmal jedes einzelne Recht überprüft werden. Hier [18] wird beschrieben, wie man das überprüfen kann. Bis jetzt haben 42 Personen, sichtbar auf der Webseite der Quelle, bestätigt, dass diese Lösung funktioniert. Das Rechte-Überprüfen muss dem Rechner des Lehrers gemeldet werden und dieser entscheidet, ob das als Betrugsversuch gewertet wird.

Erklärung zu Algorithmus 5.1 auf der nächsten Seite:

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

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

Algorithmus 5.1 Android, Überprüfung des Besitzes von Zugriffs-Rechten des Manifests
Quelle: [18]

```
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 }
```

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.

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 unbeabsichtigt beendet.

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

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

Unter Windows lässt sich das Programm nicht killen ohne Administrator-Rechte, wenn es selbst mit Administrator-Rechten läuft oder ein Treiber ist, der nicht als Prozess auftaucht.

Was ist aber, wenn sich das Programm wegen einem Programmierfehler unbeabsichtigt beendet? Es ist möglich vollkommen abdeckend mit Try-Catch-Finally-Blöcken Abstürze zu behandeln, so dass in dem Fall der Lehrer informiert wird, dass der Schüler nicht mutwillig das Programm mit dem Kill-Kommando eines Taskmanagers beendet hat. Jedoch kann die Nutzung von Try-Catch z.B. in Schleifen zu Performance-Einbußen führen. Außerdem kostet es Zeit und Konzentration zu überprüfen, dass wirklich an jeder Stelle Try-Catch verwendet wurde. Dabei können sich Fehler einschleichen. Dazu müssen so

genannte Audits durchgeführt werden.

Alle Activitys von Android werden in einem Stapel (Stack) gespeichert und die sichtbare Activity ist oben. Wenn zu wenig RAM zu Verfügung steht, beendet Android die unterste Activity gewaltsam mit dem Kill-Kommando. Mit einem Dienst kann dies nicht geschehen. Deshalb muss ein Dienst in Kombination mit Acitivitys programmiert werden.

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 das Programm benötigt, das diese Arbeit thematisiert. 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 Schüler, 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 Schüler Leserechte für ausgewählte Bereiche hat, in denen der Lehrer von der Ferne Schreibrechte hat, oder es wird mit ROMs gearbeitet, z.B. CD-ROMs, die der Schüler nicht manipulieren kann, weil sie nur eintmal 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 Kryptographie. Deshalb ist die Geschwindigkeit für E-Mails vertretbar. Z.B. hat das symmetrische obsolete DES eine Zeitkomplexität von O(Log n) und das asymmetrische RSA $O(n^3)$
[51].

Eine weitere Alternative ist es spezielle Geräte zu betreiben, die Schüler 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 Schüler 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-Unterstüztung), mit der die Authentizität eines Betriebssystemsherstellers 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 Schüler 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 Schüler 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 Schüler manipuliert wurde. Die Überprüfung ist sicher, wenn sie direkt am Schülertablet mit Boardmitteln durchgeführt wurde, statt über Netzwerk, und das Gerät nicht vom Schüler 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 Schüler 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 [22] ist beschrieben, dass das Signierungssystem umgangen werden kann, wenn es eine Lücke gibt in Android, 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 [29] 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.

6. 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 SSL-Socket-Paketen
- Empfangen von SSL-Socket-Paketen

Gemessen wurde mit einem 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 müsste eigentlich mit 30 Tablets getestet werden, wenn jedoch mit einem Tablet gezeigt wird, dass die Geschwindigkeiten den Ablauf im Klassenraum nicht behindern, dann ist gezeigt dass die Lösung funktioniert. 30 Tablets würden die Aufgaben 30 mal schneller absolvieren, wenn mit einem Kern gearbeitet wird.

Es werden also parallel zwischen 2 Geräten 30 Verbindungen hergestellt mit einmal Senden und Empfangen parallel.
Gemessen wird der Beginn der ersten Initialisierung bis zum Abschluss der letzten.
Die Summe der 30 Sende- und Empfangszeiten ist addiert, d.h. dass nicht vom ersten Senden bis zum letzten Ende des parallelen Sendens gemessen wurde.
Gesendet und Empfangen werden 300 Java-Char-Elementen (UTF-16).
Die letzte Messung „alles" betrachtet den Vorgang von der Initialisierung bis zum Senden und Empfangen, bis der letzte fertig ist, parallel.
Der Quellcode zum Messen ist auf der zugehörigen CD zu dieser Arbeit.

Tabelle 6.1.: Geschwindigkeits-Messungen

in ms	SSL-Initialierung	ΣSenden	ΣEmpfangen	alles
$\bar{t}$				
Median				
E(T)				
σ				

7. Schluss

7.1. Zusammenfassung

7.2. Ausblick

Problem Lehrer neue Macht, thema für Arbeiten der Pädagogik

A. Abbildungsverzeichnis

B. Tabellenverzeichnis

C. Algorithmenverzeichnis

D. Literaturverzeichnis

[1]

[2] Chapter 6 - digital certificates.

[3] Java api for google geocoder v3.

[4] Java tm secure socket extension (jsse) reference guide.

[5] Sensors overview.

[6] Tasks and back stack.

[7] Tastenkombinationen.

[8] Tastenkombinationen windows 8.1 / windows rt 8.1.

[9] Verwenden von gesten.

[10] Whitelisting security solutions by the features.

[11] Windows 8-berührungssteuerung und -tastaturbefehle.

[12] Your current location, what is geolocation? what is html 5 geolocation technology?

[13] Get installed applications with name, package name, version and icon, 2009.

[14] Android - simulate home click, Mai 2010.

[15] Open another application from your own (intent), Mai 2010.

[16] How to detect when an android app goes to the background and come back to the foreground, - 2010 - 2012.

[17] Gruppenrichtlinien für anfänger, April 2011.

[18] How permission can be checked at runtime without throwing securityexception?, 2011-2013.

[19] Difference between truststore and keystore in java - ssl, September 2012.

[20] Sha-3 winner, Oktober 2012.

[21] android intercept recent apps button, Juli 2013.

[22] Androids code-signatur lässt sich umgehen, Juli 2013.

[23] How to reset default launcher/home screen replacement?, Maerz 2013.

[24] Windows 8 shortcuts: Immer den richtigen drücker, Juni 2013.

[25] Autocompletetextview, August 2014.

[26] Camera, August 2014.

[27] Hooks im kernel sollen android sicherer machen, Maerz 2014.

[28] Packagemanager, August 2014.

[29] Schutzlose wächter - antiviren-software als sicherheitslücke, Juli 2014.

[30] Anton Bassov. Hooking the native api and controlling process creation on a system-wide basis, Oktober 2005.

[31] Arnab Chakraborty. Developing kiosk mode applications in android, unbekannt unbekannt.

[32] Jan Dittberner. Keytool, openssl, und co. was nehme ich wofür und warum?, Mai 2011.

[33] Mike Gavaghan. Java geodesy library for gps vincentys formulae, April 2008.

[34] Romain Guy. Android simulate home click, Mai 2010.

[35] Mike Harvey. Der vs. crt vs. cer vs. pem certificates and how to convert them, Oktober 2011.

[36] Mark Heitbrink. Gruppenrichtlinien laufwerke im explorer ausblenden und den zugriff darauf verhindern, Januar 2013.

[37] Udo Hermann. C sharp globale hotkeys, welche wirklich global sind, September 2013.

[38] Gerard J. Holzmann. Design and validation of computer protocols, 1991.

[39] Thomas Joos. So ändern sie registry-einstellungen über gruppenrichtlinien, Januar 2014.

[40] Prof. Dr. Kreitz. Kryptographie und komplexit ?at.

[41] Tyler McHenry. Designing painless protocols, Dezember 2009.

[42] mkyong. Java find location using ip address, Oktober 2013.

[43] Wajahat Karim Muhammad Usama bin Aftab. *Learning Android Intents Explore and apply the power of intents in Android application development*. Packt Publishing, Januar 2014. ISBN 978-1-78328-963-9.

[44] Frank Racis. How do i disable a system device programatically?, Juli 2011.

[45] Wolfgang Sommergut. Windows 7 enterprise: Alternativen zu applocker, Maerz 2010.

[46] Wolfgang Sommergut. Registry-schlüssel mit group policy preferences erstellen und löschen, September 2013.

[47] Staff. How-to create kiosk mode on the nexus 7, April 2013.

[48] Axel Vahldiek. Der öffentliche pc windows narrensicher konfigurieren. *Heft 3*, 1(1):8, Februar 2011. 1.

[49] Erik van Oosten. Securing connections with tls, November 2009.

[50] "WabilityDE". C sharp | dienst programmieren und installieren | tutorial. Video, August 2012.

[51] Harsh Sharma Yogesh Kumar, Rajiv Munjal. Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures, Oktober 2011.