Instant Message Whispering via Covert Channels

Qualitätssicherungsdokument

Gruppe 35:	Jan Simon Bunten <jan_simon.bunten@stud.tu-darmstadt.de> Simon Kadel <simon.kadel@stud.tu-darmstadt.de> Martin Sven Oehler <martin_sven.oehler@stud.tu-darmstadt.de> Arne Sven Stühlmeier <arne_sven.stuehlmeier@stud.tu-darmstadt.de>
Teamleiter:	Philipp Plöhn <philipp.ploen@stud.tu-darmstadt.de>
Auftraggeber:	Carlos Garcia <carlos.garcia@cased.de> FG Telekooperation FB 20 - Informatik
Abgabedatum:	15.2.2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Bachelor-Praktikum WS 2013/2014
Fachbereich Informatik

Inhaltsverzeichnis

1 Einleitung

Ziel des Projekts ist es, eine Bibliothek zu entwickeln, die es ermöglicht, unentdeckt Kommunikationskanäle zu einem oder mehreren anderen Teilnehmern zu öffnen. Um die Kommunikation vor Dritten zu verstecken, werden sogennante Covert Channels verwendet.

Covert Channels

Covert Channels sind Kommunikationskanäle, die von Außen nicht als solche erkennbar sind. In der Literatur sind viele unterschiedliche Covert Channels bekannt.

Im Unterschied zur Kryptographie, die nur die Daten eines Kanals verbirgt, wird der ganze Kanal verborgen. Dadurch wird es Dritten erschwert, die Verkehrsdaten der Verbindung (Zeitpunkt, Dauer) auszuwerten. Ist es möglich die Pakete einer bestehenden Verbindung anderer Teilnehmer des Netzwerks zu verändern, kann damit auch die Identität der Nutzer verborgen werden.

Wie bei offenen Kanälen ist es auch bei Covert Channels von großer Bedeutung, wie groß der Datendurchsatz ist und wie zuverlässig die Informationen übertragen werden. Vor allem der Datendurchsatz ist bei Covert Channels üblicherweise stark beschränkt.

Implementierung

Das Hauptziel ist, ein Framework zu implementieren, das notwendige Funktionen für die Covert Channels bereit stellt. Dazu gehören das Öffnen und Schließen von Covert Channels, das Senden von selbst erstellten Paketen, das Empfangen von Paketen und das Anzeigen von Statistiken der geöffneten Kanäle. Die eigentlichen Covert Channels können als Plugins hinzugefügt werden. So soll sichergestellt werden, dass die Bibliothek für unterschiedliche Covert Channels genutzt werden kann.

Neben dem Framework werden wir im Rahmen des Projekts drei unterschiedliche Covert Channels implementieren. Zuerst ein einfacher Channel, der darauf beruht Informationen im Header eines TCP oder UDP Pakets zu verstecken. Als zweites einen komplizierterer Covert Channel aus der Literatur. Wenn möglich soll dann noch ein dritter von uns entwickelter Covert Channel implementiert werden.

Das Projekt soll als Open-Source veröffentlich werden, um es anderen zu ermöglichen die Bibliothek in ihren Projekten zu verwenden oder die Bibliothek weiterzuentwinkeln. Wir werden eine Beispielanwedung in der Art eines Instant Messagers entwickeln, um die Funktionen zu demonstrieren.

2 Qualitätsziele

2.1 Reife

Begründung: Unser Projekt soll in anderen Programmen als Bibliothek verfügbar sein. Insbesondere soll es zum Testen eines Tools, mit dem Covert Channels entdeckt werden können, verwendet werden. Der Programmierer verlässt sich darauf, dass die Bibliothek nicht der Grund für Fehlverhalten oder Abstürze ist, sondern fehlerfrei funktioniert.

Maßnahmen: Die Zuverlässigkeit kann durch Testen verbessert werden. Deshalb benutzen wir die boost.test Bibliothek, die automatische Tests in C++ ermöglicht. Diese Tests müssen mindestens Anweisungsabdeckung erreichen. Außerdem führen wir Code Reviews anhand einer Checkliste durch, in der Punkte wie Endlosschleifen und Prüfen von Parametern mit beachtet werden.

Prozess: Die automatischen Tests werden mindestens einmal pro Woche auf der aktuellen Version der Software komplett ausgeführt. Liste mit Datum und Betriebssystem. Fehler werden im Ticketsystem unseres SCM-Servers eingetragen und bei der Planung de Sprints mit berücksichtigt.

Nach Abschluss einer User Story wird der Code von einem an dieser unbeteiligten Teammitglied anhand einer Checkliste (siehe Anhang) überprüft. Mögliche Fehler müssen noch in der selben Iteration von den Entwicklern behoben werden, sonst gilt die User Story nicht als abgeschlossen. Dieser Vorgang des Überprüfens und Verbesserns wird solange wiederholt, bis alle Punkte auf der Checkliste erfüllt sind.

2.2 Testbarkeit

Für unser Projekt ist vorgesehen, dass es als Open Source veröffentlicht wird. Dardurch können Entwickler die Bibliothek an ihre Ansprüche anpassen, indem sie zum Beispiel eigene Covert Channel implementieren. Um das zu erleichtern, muss unser Projekt anpassbar und erweiterbar sein. Außerdem ist die Testbarkeit einiger Funktionen des Projekts komplex, weshalb sie in den Projektzielen vom Auftraggeber als Qualitätsziel hervorgehoben wurde.

Maßnahmen: Damit es einfach ist, diese durchzuführen, wird eine klare Beschreibung und Trennung der Aufgaben benötigt (Seperations of Concerns). Das erreichen wir, indem wir während des Designs alle Aufgaben unserer Software festhalten und einem unserer Module zuweisen. Außerdem müssen die Schnittstellen und Interfaces gut dokumentiert sein.

Es ist sehr schwierig, Testbarkeit durch Werkzeuge sicher zu stellen. Eine gute Testbarkeit ergibt sich durch eine Architektur der Software, die dieses Qualitätsmerkmal beachtet. Es muss beim Entwurf berücksichtigt und während der Entwicklung stets überprüft werden.

Konkrete Maßnahme?

Prozess: Die einfachste Möglichkeit, die Testbarkeit zu überprüfen, ist es, Tests zu schreiben und dabei festzustellen, wie gut dies möglich ist. Wenn dabei auffällt, dass manche Funktionalitäten nur schwer oder nicht zu testen sind, werden wir dies in unseren Teamtreffen besprechen und eine Lösung durch Anpassen der Architektur ausarbeiten. Beleg?

Verständlichkeit + Modifizierbarkeit Auch der hohe Stellenwert der Erweiterbarkeit unserer Bibliothek ergibt sich aus der Veröffentlichung als Open Source Software. Alle Nutzer der Bibliothek sollen in die Lage versetzt werden der Bibliothek so einfach wie möglich eigene Funktionen hinzufügen zu können. Dafür ist die Erweiterbarkeit ein zentrales Qualitätsmerkmal. Begründung: Open Source

Maßnahme: Dokumentation + Code Conventions Das möchten wir durch eine ausführliche Dokumentation aller Module und Funktionen der Bibliothek erreichen. Außerdem haben wir Code Conventions festgelegt, die an die Code Conventions für Google Entwickler angelehnt haben. Damit wollen wir einheitliche Bezeichner und eine einheitliche Struktur des Codes sicherstellen und seine Lesbarkeit verbessern. So sollen Nutzer sich leichter in der Bibliothek zurecht finden wenn sie Änderungen vornehmen möchten. Prozess: automatisch generiert, dann vervollständigt. Im Review beachtet Außerdem müssen die Schnittstellen und Interfaces gut dokumentiert sein. Die Dokumentation wird durch die Software Doxygen und entsprechende Kommentierung aller Funktionen und Klassen der Bibliothek automatisch erstellt und von uns durch zusätzliche Informationen ergänzt. Die Einhaltung der Code Conventions wird in den Code Reviews, die nach dem Abschluss einer User Story durchgeführt werden, überprüft und muss gegebenfalls vom Programmierer angepasst werden, sollten Abweichungen festgestellt werden.