IT Security and Service Management
Steinbuch Centre for Computing
Karlsruhe Institute of Technology

Sicherheitsgutachten »POEM@HOME«

Dipl.-Inform. Tobias Dussa

7. August 2008

tobias.dussa@kit.edu

Das vorliegende Dokument begutachtet die Softwarearchitektur des Projekts »POEM@HOME« im Hinblick auf Sicherheitsrisiken. Betrachtet wird die Konzeption der Software, nicht aber die konkreten Implementationen; es erfolgt also keine Analyse im Sinne des Auffindens von Programmierfehlern. Ziel des Gutachtens ist es, eine Risikoabschätzung aus Nutzersicht für die Teilnahme am Projekt zu geben.

1 Beschreibung der begutachteten Software

»POEM@HOME« (»Protein Optimization with Energy Methods«, `http://boinc.fzk.de/poem/`) ist ein Projekt des Insituts für Nanotechnologie (`http://www.fzk.de/int/`). Ansatz ist es, die anfallenden Berechnungen von den Computern freiwillig teilnehmender Dritter durchführen und die Ergebnisse rückmelden zu lassen. Die für dieses verteilte Rechnen benötigte Software wird vom Projekt zur Verfügung gestellt und basiert auf dem Projekt »BOINC« (»Berkeley Open Infrastructure for Network Computing«, `http://boinc.berkeley.edu/`).

BOINC stellt ein client-server-basiertes Softwarepaket zur Verfügung, das Hilfsmittel für das Durchführen verteilten Rechnens bereitstellt. Die Software eines BOINC-basierten Projekts zerfällt in drei verschiedene Komponenten:

- den BOINC-Server,
- den BOINC-Client und
- den projektspezifischen Codeteil.

Sowohl BOINC-Server als auch -Client werden vom BOINC-Projekt selber zur Verfügung gestellt; der anwendungsspezifische Code stammt dagegen vom

jeweiligen Anwendungsprojekt programmiert und bereitgestellt, in diesem Fall also von POEM@HOME.

Um an einem Rechenprojekt teilzunehmen, ist es nötig, den BOINC-Client zu installieren und sich bei dem Projekt zu registrieren. Der BOINC-Client lädt dann vom Server des Projekts zunächst Programmcode nach, der vom Rechenprojekt bereitgestellt wird. Danach werden nacheinander Rechenaufträge heruntergeladen, die im wesentlichen den bereits geladenen Programmcode parametrisieren, berechnen und die Ergebnisse wieder auf den Server des Projekts hochgeladen. Diese Schleife wird so lange durchlaufen, wie der BOINC-Client ausgeführt wird.

Zusätzlich zur »normalen« Operation eines BOINC-Clients wird vor dem Herunterladen jedes Rechenauftrags vom Server des Rechenprojekts geprüft, ob eine neuere Version des projektspezifischen Programmcodes vorliegt. Ist dies der Fall, wird das Update automatisch heruntergeladen und installiert.

Jedes Rechenprojekt, das BOINC nutzt, hat ein Schlüsselpaar, mit dessen Hilfe der zur verfügung gestellte Programmcode signiert wird. Auf diese Weise wird die Authentizität des Programmcodes dokumentiert.

Das Projekt POEM@HOME nutzt laut eigener Aussage einen nicht ans Internet angeschlossenen Rechner zum Signieren seiner Programmcodes; der private Schlüssel des verwendeten Schlüsselpaares wird ebenfalls nach eigener Aussage sicher verwahrt.

2 Risiken

Dieses Gutachten beleuchtet die Risiken aus Sicht eines Anwenders, der seine Rechenressourcen mit Hilfe von BOINC dem POEM@HOME-Projekt zur Verfügung stellen möchte. Risiken, die daraus für die Server beziehungsweise das Projekt selber erwachsen, liegen ausserhalb der Betrachtung des Gutachtens.

Aus Sicht der Anwender bestehen bei Teilnahme an POEM@HOME zwei wesentliche Risiken:

- Missbrauch eigener Rechenressourcen durch fremde Software, die nicht gemäß ihrer Bestimmung arbeitet, und mögliche dadurch verursachte Nachteile. Ein Beispiel hierfür ist der Missbrauch eines Rechners als Versender von Spam-Mail. Hierdurch wird einerseits als primäre Beeinträchtigung CPU-Zeit des betroffenen Rechners gebraucht, die damit legitimen Benutzern nicht mehr zur Verfügung stehen; andererseits können aber auch sekundäre Beeinträchtigungen auftreten, wenn beispielsweise der betroffene Rechner wegen des Spamversands auf Blacklists eingetragen wird und auch keine regulären E-Mails mehr versenden kann.
- Kompromittierung eigener Daten, die auf den dem Projekt zur Verfügung gestellten Rechnern gespeichert werden. Dabei sind die eigenen Daten in mehrfacher Hinsicht in Gefahr, denn sie können gelöscht, verändert oder ausgespäht werden.

3 Beurteilung

Die genannten Risiken entstehen durch versehentlich fehlerhafte oder absichtlich schädliche Software. Es stellen sich damit für den Benutzer die folgenden drei Fragen:

1. Ist die Software vertrauenswürdig in dem Sinne, dass davon ausgegangen werden kann, dass kein Schadcode absichtlich eingefügt wurde?
2. Ist die Software fehlerfrei in dem Sinne, dass auch von Dritten durch Programmier- oder Designschwächen der Software Schadcode eingebracht werden kann? Diese Art von Fehlern werden im folgenden als »Sicherheitslücken« bezeichnet.
3. Ist die Software zuverlässig in dem Sinne, dass keine Fehler vorhanden sind, die unabsichtlich eines der genannten Probleme auslösen und so zufällig Schaden anrichten? Diese Fehlerklasse wird zur Unterscheidung von Sicherheitslücken im folgenden einfach »Programmierfehler« genannt, obwohl natürlich auch Sicherheitslücken in der Regel Programmierfehler zugrundeliegen.

Aufgrund der mehrschichtigen Architektur des BOINC-Softwarepakets müssen die obigen Fragen zudem für Software beantwortet werden, die nicht von einem einzigen Autorenteam, sondern vielmehr von zwei unterschiedlichen Quellen stammt. Dies sind die BOINC-Entwickler für die BOINC-Infrastruktur auf der einen Seite, die Mitarbeiter des POEM@HOME-Projekts für die projektspezifischen Softwareteile auf der anderen Seite.

Bei BOINC handelt es sich um ein renommiertes Open-Source-Softwareprojekt, das bereits mehrere Jahre weltweit von verschiedenen Projekten eingesetzt wird; es hat damit eine breite Benutzerbasis. Es kann mit großer Wahrscheinlichkeit davon ausgegangen werden, dass kein Schadcode absichtlich in die vom BOINC-Projekt bereitgestellte Software eingefügt wurde. Selbst wenn dies beabsichtigt wäre, so würde sehr wahrscheinlich im Laufe der Zeit ein unabhängiger Programmierer den Schadcode entdecken, denn es handelt sich um Open-Source-Software, so dass der Quellcode für jedermann offenliegt. Eine derartige Entdeckung würde aufgrund der großen Benutzerbasis zu einem massiven Vertrauensverlust der Projekte und Anwender gegenüber dem BOINC-Projekt führen, der vermutlich nur sehr schwer oder gar nicht umkehrbar wäre. Es ist daher sehr unwahrscheinlich, daß das BOINC-Projekt bewusst Schadcode einfügt, selbst wenn man die Absicht unterstellen wollte.

Im Bezug auf unabsichtliche Programmierfehler ist die Lage ähnlich, wenngleich nicht ganz so kritisch. Auch solche Fehler werden mit großer Wahrscheinlichkeit – je nach Tragweite des einzelnen Fehlers – früher oder später entdeckt und korrigiert. Jedoch ist in diesem Falle nicht davon auszugehen, dass keine Fehler vorhanden sind; im Gegenteil wäre es nicht ratsam anzunehmen, dass eine komplexe Software wie die BOINC-Bibliothek fehlerfrei sei. Jedoch ist es auch hier sehr wahrscheinlich, dass die Anzahl von Sicherheitslücken oder

Programmierfehlern sehr gering ist. Derartige Fehler werden in der Regel aufgrund ihrer offensichtlichen Brisanz sowohl sehr schnell bekannt als auch sehr schnell korrigiert. Hierbei ist allerdings darauf hinzuweisen, dass sich die Situation im Bezug auf Sicherheitslücken sich durchaus von der im Bezug auf Programmierfehler unterscheidet: Während es für den Entdecker eines Programmierfehlers in der Regel nur sehr wenig Anreiz gibt, die Entdeckung geheimzuhalten, so gibt es durchaus eine Reihe von Gründen dafür, eine gefundene Sicherheitslücke für sich zu behalten. Die Wahrscheinlichkeit dafür, dass noch weithin unbekannte Sicherheitslücken existieren, ist folglich durchaus höher als die unbekannter Programmierfehler.

Die bisherige Betrachtung setzt voraus, dass gesichert ist, dass der eingesetzte BOINC-Client auch tatsächlich vom BOINC-Projekt stammt und nicht bereits von Dritten modifiziert wurde. Obwohl insbesondere nicht auszuschließen ist, dass beispielsweise die Webseite des BOINC-Projekts von Einbrechern verändert wurde, so kann doch davon ausgegangen werden, dass ein derartiger Einbruch beziehungsweise eine nicht autorisierte Veränderung des Quellcodes oder der Binärdistribution wegen der breiten Benutzerbasis schnell auffallen würde. Zudem ist der BOINC-Client in gängigen Linux-Distributionen bereits enthalten, so dass dort die distributionsübliche Möglichkeit der Paketvalidierung zur Verfügung steht. In der Regel wird die Authentizität von Programmpaketen mit Hilfe kryptographischer Verfahren sichergestellt.

Etwas anders stellt sich die Situation im Bezug auf die projektspezifische Anwendung, in diesem Fall also den Programmcode von POEM@HOME, dar. Auch hier besteht kein Grund zur Annahme, dass von den Autoren absichtlich Schadcode eingefügt wurde; im Gegenteil wäre dies im Falle der Entdeckung mit einem kaum wiedergutzumachenden Reputationsverlust verbunden, selbst wenn man eine derartige Absicht unterstellen wollte. Auch im Bezug auf Sicherheitslücken und Programmierfehler gilt prinzipiell dieselbe Argumentation wie für die BOINC-Software. Allerdings handelt es sich beim POEM@HOME-Applikationscode nicht um quelloffene Software; auch ist die Verbreitung naturgemäß nicht so groß wie die des gesamten BOINC-Projekts. Ein Fehler wird daher vermutlich nicht so schnell entdeckt werden. Andererseits ist der Anwendungscode auch programmiertechisch deutlich weniger komplex und umfangreich als das BOINC-Softwarepaket, so dass die Fehlerwahrscheinlichkeit von vornherein kleiner ist und damit auch weniger Sicherheitslücken und Programmierfehler auftreten.

Auch für den Applikationscode muss sichergestellt sein, dass der verwendete Code tatsächlich Originalcode des POEM@HOME-Projekts ist. Hier ist die Lage bezüglich des POEM@HOME-Anwendungscodes für den Benutzer etwas schwieriger. Der POEM@HOME-Anwendungscode kann nur von einer einzigen Quelle bezogen werden, dem vom POEM@HOME-Projekt betriebenen BOINC-Server. Daher gibt es keine Redundanz, die es dem Benutzer erlauben würde, eine eventuelle Kompromittierung des POEM@HOME-BOINC-Servers auf diesem Wege zu entdecken. Die BOINC-Software erlaubt allerdings die

kryptographische Signierung des Anwendungscodes mit Hilfe eines privaten Schlüssels. Diese Signatur wird vom BOINC-Client geprüft. Dieser Automatismus ist insbesondere deswegen wichtig, weil der Anwendungscode automatisch aktualisiert wird, wenn eine neue Version vorliegt und sich der Client mit dem POEM@HOME-Server verbindet. Der für die Codesignierung notwendige private Schlüssel wird nach Angaben des POEM@HOME-Projekts sachgerecht auf einem Datenträger gespeichert, der zu keinem Zeitpunkt mit einem Netzwerk verbunden ist; die Signatur des Codes wird dementsprechend auf einem alleinstehenden Rechnersystem vorgenommen. Diese Vorgehensweise sichert die Integrität des privaten Schlüssels auf hinreichende Weise und gewährleistet damit die Aussagekraft der Codesignatur.

4 Empfehlung

Der Beurteilung folgend muss sich ein Anwender, der dem POEM@HOME-Projekt seine Rechenressourcen zur Verfügung stellen will, die folgenden Fragen und deren Implikationen klarmachen, um die mit der Teilnahme am POEM@HOME-Projekt verbundenen Risiken einschätzen zu können:

1. Ist die BOINC-Software vertrauenswürdig?
2. Ist die POEM@HOME-Anwendungscode vertrauenswürdig?

Die Antwort auf diese Fragen hängt von der genauen Art der einzubringenden Ressourcen beziehungsweise der ansonsten darauf abgelegten Daten ab. Grundsätzlich gilt: Je wichtiger die auf einem System abgelegten Daten, desto weniger Vertrauen sollte Software Dritter entgegengebracht werden, die auf diese Daten Zugriff hat. Ein Rechnersystem, das Daten unter Geheimschutz trägt oder Personaldaten enthält, sollte keinesfalls an POEM@HOME teilnehmen. Bei anderen Computern mag die Abwägung durchaus zugunsten von POEM@HOME ausfallen. Es ist in jedem Fall anzuraten, sicherzustellen, dass auf einem teilnehmenden System nur solche Daten vorhanden sind, deren Veröffentlichung oder Verlust keine besonders schmerzhaften Nachteile mit sich brächte.

Als mögliche Verbesserung der Situation bietet sich der Einsatz eines virtuellen Rechners für den POEM@HOME-Client an. Wird die POEM@HOME-Software in einer eigenen virtuellen Maschine aufgesetzt, so ist sie durch eine weitere Schicht vom sie ausführenden Rechnersystem und den darauf gespeicherten Daten getrennt. Andererseits muss dann dieser weiteren Softwareschicht, die die virtuelle Maschine ausführt – als Beispiele seien VMware oder QEMU genannt –, ebenfalls im Sicherheitssinne vertraut werden. Ist aber beispielsweise ohnehin bereits derartige Software auf einem Rechner installiert, so kann bedenkenlos der POEM@HOME-Client in einer eigenen, zusätzlichen virtuellen Maschine ausgeführt werden, ohne dass daraus weitere Sicherheitsrisiken für den ausführenden Rechner und dessen Daten erwachsen.