

Software Kung-Fu

Selbstschutz von Sicherheits-Systemen

Der Selbstschutz von Sicherheits-Software basiert auf systematischen Untersuchungen des Quellcodes und grundlegenden Design-Entscheidungen sowie der Nutzung aktueller Abwehrtechniken. Darüber hinaus sind auch realistische Angreiferszenarien sowie eine ganzheitliche Sicht bis hin zu Anwenderschulungen entscheidend für das erreichbare Sicherheitsniveau.

Von Alexander von Gernler, München

Aufgrund ihrer Funktion sind Sicherheitskomponenten naturgemäß auch selbst ein primäres Angriffsziel. Daher ist es unerlässlich, dass sie sich selbst schützen. Hierbei ist eine systematische Sichtweise gefragt, denn jedes Sicherheitsprodukt besteht letztlich aus Hardware, Betriebssystem und Anwendungsschicht – sowie einem umgebenden System, das auch Bediener (bzw. Administratoren) einschließt.

Wegen der steigenden Komplexität von Software befindet sich tendenziell immer mehr Code auch in Sicherheits-Produkten, was wiederum eine höhere potenzielle Angreifbarkeit bedeutet – zudem setzen auch Security-Hersteller zunehmend Drittsoftware ein. Ein Angreifer kann letztlich in jedem Teil einer Sicherheitslösung verwertbare Lücken suchen, um die Kontrolle über das Gesamtsystem und die dahinter liegenden Netze zu erlangen. Es nützt daher beispielsweise nichts, wenn eine Firewall-Software zwar gut geschrieben ist, aber auf einem unsicheren Betriebssystem läuft – oder umgekehrt. Sicherheit muss immer als Kette betrachtet werden, deren schwächstes Glied über die Sicherheit des Gesamtsystems entscheidet.

Auch die „klassische“ Kryptographie liefert hier noch eine Warnung: Kerckhoffs Prinzip (benannt nach dem niederländischen Linguisten 1835–1903) besagt, dass die Sicherheit eines Kryptosystems nicht dadurch gefährdet sein darf, dass es dem Feind in die Hände fällt. Das gleiche Prinzip muss auch für Software gelten: Beruht ein wesentlicher Teil der Sicherheit auf der Geheimhaltung des Innenlebens einer Komponente, so ist diese von Anfang an ein unkalkulierbares Risiko. Denn heutzutage

muss man davon ausgehen, dass derartige Informationen nach einiger Zeit durchsickern.

Wenn man ohnehin darauf vorbereitet sein muss, dass Quellcode oder wesentliche Prinzipien des eigenen Produkts irgendwann publik werden, dann kann man diese Tatsache auch gleich zum eigenen Vorteil nutzen: Viele Hersteller stellen daher ihren Quellcode Kunden oder einer vertrauenswürdigen dritten Stelle zur Überprüfung bereit. Das schafft nicht nur Transparenz, sondern erhöht auch langfristig die Produktqualität. Eine vertrauenswürdige Stelle ist zum Beispiel das Bundesamt für Sicherheit in der Informationstechnik (BSI), wo Experten im Rahmen einer Zertifizierung Einblick in den Quellcode nehmen – gemäß dem Open-Source-Prinzip, dass viele Augen mehr sehen als wenige.

Gerade im Zusammenhang mit Zertifizierungen ist übrigens auch der Begriff des Angreiferszenarios (Security Target) sehr wichtig: Um ein vernünftiges Maß an Schutz bieten zu können, muss erst einmal klar sein, gegen welche Art von Angriffen man sich zur Wehr setzen möchte. Motivation sowie materielle und zeitliche Ressourcen von Angreifern sind sehr unterschiedlich und letztendlich entscheidend für die Durchschlagskraft ihrer Attacken. Einem Produkt, das keinem realistischen Angreiferszenario ausgesetzt wurde, sollte man von vornherein mit Misstrauen begegnen. Statt ständig löcherstopfend hinter den Angreifern herzulaufen, sollte man als Entwickler den Spieß umdrehen: Sicherheitskonzept und Selbstschutz eines Sicherheits-Systems müssen daher einer systematischen Analyse und einem vorausschauenden Entwurf entspringen.

Technische Gegenmaßnahmen

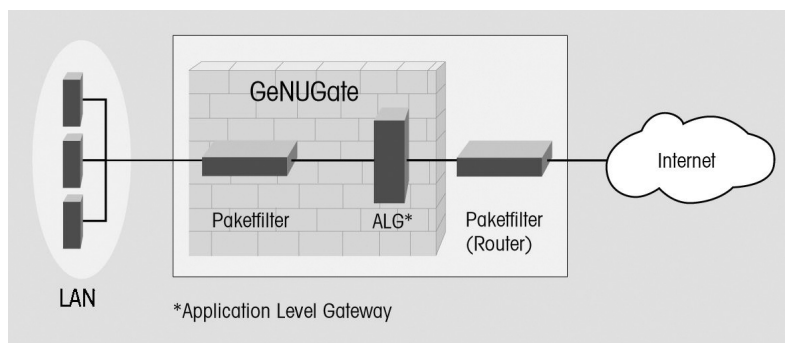
Zudem muss man davon ausgehen, dass ein einzelner erfolgreicher Angriff auf ein Sicherheitsprodukt vollkommen ausreicht, um das System oder seine „Schützlinge“ dauerhaft und nachhaltig zu kompromittieren. Auch wenn selbst eine technisch perfekte (Einzel-)Lösung noch am menschlichen Faktor scheitern kann oder schlicht an der Tatsache, dass man bildlich gesprochen eine Stahltür in eine Strohhütte einbaut: Wo jeder Fehler der letzte sein kann, muss von technischer Seite ein Angriff so weit wie möglich erschwert werden, sogar wenn für dieses Ziel Beeinträchtigungen in der Benutzbarkeit hinzunehmen sind. Einen Überblick über gängige Mechanismen zum Software-Selbstschutz geben die folgenden Abschnitte.

Mehrstufigkeit

Das Verteilen der Sicherheitsfunktionalität auf mehrere physische Maschinen (idealerweise auch auf verschiedene Betriebssysteme) zwingt einen Angreifer dazu, mehrere Einbrüche erfolgreich durchzuführen, um das Sicherheitssystem komplett zu überwinden. Dies kann ein Netzwerk-Administrator zum einen selbst erreichen, etwa durch Einkauf verschiedener Produkte unterschiedlicher Marken. Einfacher kann es sein, wenn ein Hersteller die Mehrstufigkeit bereits im Design seiner Systeme berücksichtigt.

Eingeschränkte Ablaufumgebungen

Das betriebssystemtechnische Konzept eingeschränkter Ablaufumgebungen ist heute in allen freien Unix-artigen Kernen in Form des chroot()-Systemaufrufs realisiert. chroot() sperrt einen Prozess in ein Unterverzeichnis im Gesamtdatenbaum ein, wobei für ihn dieses Unterverzeichnis als neue Wurzel des (seines) Dateibaums erscheint. Hiermit kann verhindert werden, dass ein vom Angreifer übernommener Prozess über seinen eigentlichen Arbeitsbereich hinaus Schreibzugriff auf das Gesamtsystem erhält. FreeBSD erweitert dieses Konzept mit den so genannten Jails noch um eine Separation von Prozesslisten sowie



Ein Beispiel für die Mehrstufigkeit von Sicherheitssystemen ist die Empfehlung des BSI für dreistufige Firewalls. Wenn – wie im Bild – mehrere Komponenten durch ein Produkt umgesetzt werden, so sollte es idealerweise separate physische Rechner benutzen.

Netzwerk-Interfaces und fungiert damit quasi als Betriebssystem-Container.

Exploits verhindern

Es existieren derzeit mehrere maßgebliche Konzepte, um Betriebssysteme und Anwendungen „in sich“ sicherer zu machen (s. a. Kasten). Da wäre zunächst „Write XOR Execute“ (W^X), das auf einigen Plattformen (i386, amd64) auch als NX bekannt ist (No eXecute). Der Grundgedanke dieser Technik ist, dass ein bestimmter Speicherbereich entweder ausführbaren Programmcode enthält oder Daten, aber nie beides. Deshalb muss er auch nur entweder ausführbar sein (Code) oder beschreibbar (Daten), nie aber beides zugleich.

Diese Regel kann auf modernen Architekturen mittels Hardware-Unterstützung (NX-Bit) durchgesetzt werden. Eingebaut ist sie beispiels-

weise in OpenBSD und NetBSD (W^X), Linux (PaX/ExecShield) sowie moderne Windows-Versionen (DEP). Damit kann wirksam verhindert werden, dass Schadprogramme zunächst ihren eigenen Code in den Datenbereich einer Anwendung schmuggeln und diesen anschließend als Sprungbrett für die Programmausführung benutzen.

Auf Compiler-Ebene arbeitet hingegen eine andere Technik, um den Stack einer Anwendung zu schützen, also die Komponente, in der alle lokalen Objekte einer momentan ausgeführten Routine abgelegt sind. Durch eine Überprüfung der Stack-Integrität vor dem Verlassen der Routine wird verhindert, dass Stack-basierte Angriffe mit Buffer Overflows erfolgreich sind, denn diese überschreiben ja gerade Objekte auf dem Stack. Dieses Konzept ist zum einen in einer Erweiterung des GNU C-Compilers 3.x (und ab 4.1



EDV - Audit Consult
Hans-Jürgen Stritter

EDV-Audit Consult
Postfach 12
71718 Oberstenfeld

Tel.: +49 7062 930056
Fax: +49 7062 930057
E-Mail: info@edv-auditconsult.de
URL: www.edv-auditconsult.de

Kontakt: Hans-Jürgen Stritter

- IT-Revision & Beratung IT-Sicherheitsmanagement
- IT-Systemprüfungen (IDW FAIT 1, 2, 3/ PS 330)
- Software-Testate (IDW PS 880)
- SAP™ R/3-Security- und Berechtigungskonzeptionen
- SAP™ R/3-Prüfungen Berechtigungssystem(e)
- Externer Datenschutzbeauftragter (BDSG) & Datenschutz-Audits
- Forensic Computing & Fraud-Untersuchungen (Computermanipulationen & Korruptionsfälle), Risikomanagement (KonTraG) und Management Audits
- Sarbanes-Oxley Act-Einführungskriterien (SOA) - Professionelles Management interner Kontrollen
- (Firmen-) Seminare / Konferenzen / Workshops / Messe-Moderationen

standardmäßig) enthalten, zum anderen auch in einigen kommerziellen Compilern realisiert. Heutzutage besitzen dadurch sowohl die meisten Linux- und BSD-Distributionen als auch Windows Vista einen solchen Stack-Schutz (aber nicht notwendigerweise jede Drittsoftware für diese Betriebssysteme!).

Überwachung von Systemaufrufen

Die Schnittstelle, die eine Anwendung benutzt, um vom Betriebssystem Ressourcen anzufordern oder spezielle Operationen auszuführen, beschreibt die Menge zur Verfügung stehender Systemaufrufe – kurz: Syscalls. Von vielen Anwendungen ist bereits im Voraus bekannt, welche Systemaufrufe während der Laufzeit

getätigt werden. Dieses spezielle Profil für die Anwendung lässt sich im Interesse der Sicherheit überwachen und durchsetzen. Wird die Anwendung von einem Angreifer übernommen, so kann dieser ebenfalls nur die ursprünglich vorgesehenen Systemaufrufe durchführen. Unter Linux und BSD existiert hierzu ein Mechanismus namens *systrace*. Leider wird er aufgrund seiner hohen Komplexität praktisch von niemandem eingesetzt – gerade Hersteller von Sicherheits-Software sollten sich jedoch damit beschäftigen.

System Lockdown

Viele Sicherheitssysteme, beispielsweise eine Firewall, müssen im normalen Betriebs-Modus nicht flexibel umkonfigurierbar sein. Oft

genügt es, wenn eine Rekonfiguration oder ein Update erst durch den Neustart des Systems wirksam wird (viele Firewalls arbeiten ohnehin redundant, sodass ein Neustart eines Systems keine Unterbrechung der Verbindungen verursachen würde). Ein entsprechender Schreibschutz für die Konfiguration kann die Auswirkungen eines Angriffs drastisch senken.

Zwar ist dieses Verfahren nicht bei jedem Produkt einsetzbar, Hersteller sollten sich aber überlegen, wo sie diese System-Sperrung sinnvoll einsetzen können. Realisierbar ist das beispielsweise durch das Setzen von Dateiflags unter Linux (*grsecurity*) und BSD (*file flags* und *securelevel*), die bestimmte Dateien sogar gegenüber dem Administrator schreibschützen. Die Flags selbst können nur in Verbindung mit einem System-Neustart wieder gelöscht werden. Windows besitzt ebenfalls derartige Mechanismen: Die Windows File Protection etwa verhindert, dass wichtige, vom Betriebssystem selbst installierte Systemdateien verändert oder gelöscht werden.

Faktor Mensch

Es wäre allerdings kurzfristig, die Widerstandsfähigkeit eines Sicherheitsprodukts nur mit technischen Maßnahmen garantieren zu wollen. Spätestens beim Kunden wird aus dem Produkt zusammen mit Einsatzumgebung, Bedienern (und ggf. Bedienerfehlern) ein Gesamtsystem, das vielfältigen neuen Einflüssen unterliegt. Der Einfluss des menschlichen Faktors beginnt aber auch schon beim Entwicklungsprozess. Sicherheit ist ein Querschnittsthema, daher muss man sie bei jedem noch so kleinen Schritt berücksichtigen. Sicherheit ist vor allem kein Modul oder Feature, das nachträglich auf ein Produkt aufgeflanscht werden kann, sondern sie muss sich wie ein roter Faden durch die gesamte Entwicklung ziehen.

Softwareschutz im Überblick

Die nachfolgend skizzierten Techniken zum Softwareschutz sind heutzutage praktisch einsetzbar, haben jedoch einen unterschiedlichen Verbreitungsgrad unter den aktuellen Betriebssystemen. Als einziges System besitzt OpenBSD alle diese Mechanismen (s. a. www.openbsd.org/papers/ven05-deraadt/).

_____ **Data Execution Prevention / W^X / PaX:** verhindert das Ausführen von Datensegmenten als Programmcode.

_____ **Stack-smashing-Protection / ProPolice:** automatischer Einbau von Integritäts-Checks für den Stapelspeicher durch den Compiler und dadurch Verhinderung Stack-basierter Buffer-Overflows.

_____ **Stackgap:** Beim jedem Programmstart wird der Stack an einer anderen Adresse erzeugt – Buffer-Overflows, die sich auf feste Adressen verlassen, funktionieren so nicht mehr.

_____ **Zufallsadressen für dynamisch gelinkte Bibliotheken:** Bei jedem neuen Start eines Programms liegen die Laufzeitbibliotheken an anderen Stellen im Adressraum – Exploits, die Routinen an festen Adressen suchen, versagen.

_____ **Zufälliges Verhalten für dynamisch allozierten Speicher:** Wenn auch diese Speicherbereiche an nicht vorhersagbaren Stellen liegen, erschwert dies Angriffe noch weiter.

_____ **Guard Pages:** Korrekt benutzbarer Speicher ist hierbei immer flankiert von gesperrten Bereichen – wird über die erlaubte Grenze hinaus geschrieben, führt dies zwangsläufig zu einer Speicherschutzverletzung und damit zur Beendigung des Programms.

_____ **Privilegienseparation von Systemdiensten:** Nur noch ein kleiner Teil eines Programms läuft dabei mit Administratorprivilegien, der viel größere Rest besitzt nur normale Benutzerrechte.

Hilfreich ist hierbei ein Prozess zur Qualitätssicherung. Bei Sicherheitsprodukten bietet sich zudem eine Zertifizierung nach Common Criteria (CC) an, die ebenfalls schon hohe Anforderungen an den Entwicklungsprozess stellt. Auf diese Weise schlägt man zwei Fliegen mit einer Klappe: Sicherheit wird als Querschnittsthema implementiert und man erhält gegenüber potenziellen Kunden einen Nachweis, dass man sauber gearbeitet hat.

Abrunden lässt sich das Ganze durch simulierten Einsatz in einer Testumgebung: Eine Schwachstellen-Analyse in Form eines Penetrations-Tests durch Externe oder auch ein Brainstorming innerhalb der Entwicklerteamschicht fördert oft „interessante“ Lücken zutage und sollte daher noch vor der endgültigen Marktreife durchgeführt werden. Im Falle einer Zertifizierung liefert das ohnehin zu erstellende Security

Target wertvolle Anhaltspunkte, auf welche Schwachstellen man unbedingt testen sollte. Bei früheren Versionen der CC war eine solche Analyse sogar zwingend vorgeschrieben – ein sehr sinnvolles Kriterium.

Auf Kundenseite ist gerade bei Sicherheitsprodukten auch die Schulung der Anwender entscheidend für die Sicherheit der Gesamtlösung – nicht selten finden Einbrüche auch aufgrund von Bedienfehlern oder dank Social Engineering statt (vgl. S.92).

Fazit

Zusammenfassend kann man feststellen, dass der Selbstschutz von Sicherheits-Software gar nicht hoch genug bewertet werden kann – mit ihm steht und fällt in letzter Konsequenz das gesamte Sicherheitskonzept eines Netzwerks. Daher sind die Hersteller von Sicherheits-Soft-

ware in der Pflicht, hohe Maßstäbe an die Entwicklung ihrer Produkte anzulegen. Um gegenüber Kunden und unabhängigen Experten Glaubwürdigkeit zu erlangen, ist nicht nur technisches Know-how erforderlich, sondern auch ein verantwortlicher und transparenter Umgang mit später erkannten Sicherheitsproblemen. Das fängt bei einer Sicherheits-Zertifizierung des Produkts an, zieht sich durch das Bug-Management und endet mit anspruchsvoller Kunden-schulung. ■

Dipl.-Inf. Alexander von Gernler (grunk@openbsd.org) ist OpenBSD-Committer und arbeitet als Softwareentwickler bei der GeNUA mbH in Kirchheim bei München.

Im Mittelpunkt: Die Unternehmenssicherheit



Nürnberg, 20. – 21. Juni 2007



Bayerischer Sicherheitstag 2007

BVSW-Fachtagung · Foyerausstellung Security

Gute Gründe für Ihre Teilnahme:

- **Im Fokus:** Physikalische- und IT-Sicherheit
- Sicherheitsanlagen erstellende Installationsfachbetriebe, Fachplaner TGA und „Sicherheit nachfragende“ Unternehmen als **Besucher**
- **BVSW-Fachtagung** unter dem Motto „Unternehmenssicherheit – ein Mehrwert für die Wirtschaft“
- **Synergieeffekte durch Vernetzung mit der ELTEC**, Süddeutschlands etablierter Fachmesse für elektrische Gebäudeinstallations- und -systemtechnik (www.eltec.info)

Infos und Anmeldung unter:

Tel +49 (0) 9 11. 86 06-84 41
www.bayerischer-sicherheitstag.de

Vier starke Partner:



NÜRNBERG MESSE