Sicherheitsanforderungen mit SAS umsetzen

Andreas Menrath
HMS Analytical Software GmbH
Rohrbacher Str. 26
69115 Heidelberg
Andreas.Menrath@analytical-software.de

Zusammenfassung

Der folgende Artikel widmet sich diesem (bei vielen unbeliebten) Thema und versucht es möglichst praxisnah zu vermitteln. Nach einer theoretischen Einführung sollen Möglichkeiten vorgestellt werden, um zunächst den eigenen Bedarf an Sicherheit ermitteln zu können. Anschließend werden ausgewählte SAS Sicherheitsfeatures vorgestellt und anschließend ihr Schutzpotential ermittelt, um den Leser in die Lage zu versetzen selbstständig die eigene Sicherheitsarchitektur kritisch zu hinterfragen.

Schlüsselwörter: Sicherheit, SAS Plattform, PROC PWEncode, Secure Macro, Verschlüsselung

1 Einleitung

Durch viele zuletzt bekannt gewordene Sicherheitslücken mit klangvollen Namen wie HeartBleed oder ShellShock, sowie die Tätigkeiten der NSA Behörde in den USA erlebt das Thema Sicherheit nicht nur in den Nachrichten eine Renaissance, sondern wird auch zunehmend in Unternehmen wieder in den Fokus gerückt.
Sucht man im Internet nach Empfehlungen zur Realisierung von Sicherheitsanforderungen mit der SAS Software, so ist man hinterher meist genauso schlau wie zuvor. Entweder muss man sich durch Hunderte Seiten SAS Dokumentation kämpfen, oder man findet Implementierungsbeispiele für einzelne Sicherheitsfeatures. Eine systematische Aufarbeitung des Themas Sicherheit und auch eine Beurteilungsgrundlage zum Schutzpotential einzelner Features sucht man hingegen vergeblich.
Diese Lücke versucht dieser Beitrag zu schließen und stellt dem Leser zunächst ein Framework zur Bedarfsermittlung der eigenen Sicherheitsanforderungen vor. Anschließend werden einzelne SAS Sicherheitsfeatures vorgestellt und bewertet.

2 Sicherheit bewertbar machen

Was verbirgt sich hinter dem Begriff „Sicherheit“ und wie kann man diese Sicherheit sichtbar und transparent und somit bewertbar machen? Dieser Beitrag versucht sich diesen Fragen aus einer praktischen Anwendung heraus in mehreren Zwischenschritten anzunähern.

Meistens bringen Projekte bereits zu Projektstart eine ganze Reihe an Sicherheitsanforderungen mit. Diese werden häufig durch **gesetzliche Vorgaben** oder **Unternehmensrichtlinien** maßgeblich bestimmt, können aber auch durch die **Stakeholder** oder **Datenlieferanten** ergänzt werden. Hier eine Auswahl an häufig „in der Wildbahn" anzutreffenden Sicherheitsanforderungen:

- vertrauliche Daten müssen vertraulich bleiben,
- nicht autorisierte Datenmanipulation soll vermieden werden,
- Kennwörter und Anmeldedaten müssen geheim bleiben, um z.B. vor Identitätsdiebstahl zu schützen und jederzeit nachvollziehen zu können, welche Einzelpersonen auf die Daten zugegriffen haben,
- technische Benutzer müssen besonders geschützt werden, da sie erweiterte Berechtigungen besitzen. Die Anmeldedaten müssen geheim bleiben und dürfen daher den Anwendern nicht bekannt werden.

Auf der anderen Seite bringen viele Sicherheitsmaßnahmen zur Umsetzung dieser Anforderungen auch **Nachteile** mit sich. So gehen für den **Fachanwender** ein hohes Maß an Bedienbarkeit und Benutzerfreundlichkeit verloren, wenn es z.B. erforderlich ist bei jedem Datenzugriff das richtige Kennwort in ein Formularfeld einzutragen. Auch kann sich z.B. durch Verschlüsselungsmaßnahmen von Daten die Performance einer SAS-Anwendung erheblich verschlechtern. Für **Entwickler** bedeutet die Umsetzung von Sicherheitsmaßnahmen evtl. höhere Aufwände und auch Einbußen in Qualitätsattributen wie Wartbarkeit und Nachvollziehbarkeit wenn beispielsweise gewisse Informationen unter keinen Umständen im SAS Log erscheinen dürfen. Solche Maßnahmen erschweren das Debugging erheblich. Und für den **Projektleiter** ist vor allem der zusätzliche Aufwand (und die damit verbundenen Kosten) eine Motivation nur so wenig Sicherheit implementieren zu lassen, wie absolut erforderlich ist.

Nun stellt sich natürlich die Frage wie sich die Sicherheitsanforderungen und die Nachteile vieler Implementierungsmaßnahmen ausbalancieren lassen. Hierzu bietet es sich an den **eigenen Schutzbedarf zu ermitteln**, in dem die eigenen Daten und auch die potentiellen Angriffsszenarien in unterschiedliche Kategorien eingeteilt werden.

Daten lassen sich beispielsweise in die folgenden Kategorien unterteilen:

- öffentlich: Daten sind der Allgemeinheit bekannt (z.B. Börsenkurse)
- geringer Schutzbedarf: werden diese Daten Unbefugten bekannt, entsteht kaum ein Schaden
- hoher Schutzbedarf: werden diese Daten Unbefugten bekannt, kann ein hoher Schaden (monetär oder Unternehmensreputation) erfolgen
- sensibel: hoch vertrauliche Daten (z.B. Gehaltsinformation). Diese dürfen beispielsweise selbst den eigenen IT Administratoren nicht bekannt werden.

Auf der anderen Seite kann man seinen Schutzbedarf auch anhand von unterschiedlichen **Angreifern** kategorisieren:

- Fachanwender: geringes bis mittleres technisches Verständnis. Benutzt Point-&Click Oberflächen von SAS, schreibt aber keinen eigenen SAS Code.
- Power User: mittleres bis hohes technisches Verständnis. Hat meist langjährige SAS Erfahrung und kennt nicht nur die allermeisten SAS Features sondern auch Workarounds und ist in der Lage einfache Angriffe selbst durchzuführen.
- Hacker: sehr hohes technisches Verständnis. Befasst sich bewusst mit Sicherheitsmaßnahmen und sucht gezielt nach Möglichkeiten diese zu umgehen.

Nachdem man nun seinen eigenen Schutzbedarf ermittelt hat, muss man sich natürlich auch die Frage stellen inwieweit einzelne Sicherheitsfeatures diesem Bedarf gerecht werden. Da nicht alle Sicherheitsfeatures gleich stark schützen, sollte auch hier eine Klassifizierung stattfinden. Es werden die folgenden vier Kategorien vorgeschlagen:

Tabelle 1: Sicherheitsstufen

Stufe	**Beschreibung**
0	Kein Schutz
1	Geringer Schutz (vor leichtfertigem Missbrauch)
2	Mittlerer Schutz (vor Workarounds & einfachen Angriffen)
3	Hoher Schutz (vor gezielten Angriffen)

Alle bisherigen Überlegungen lassen sich nun in Tabelle 2 zusammenfassen. Auf der horizontalen Achse ist der Schutzbedarf gegenüber den unterschiedlichen Angreifern und auf der vertikalen Achse der Schutzbedarf der Daten abgetragen. Der Inhalt der jeweiligen Zelle gibt an welche Sicherheitsstufe minimal erforderlich ist, um dem Schutzbedarf gerecht zu werden.
So wird beispielsweise für öffentliche Daten die Sicherheitsstufe 0 vorgeschlagen, da die Daten nicht vor Angriffen weiter geschützt werden müssen. Bei sensiblen Daten hingegen muss eine hohe Sicherheitsstufe erreicht werden. Geht man jedoch davon aus, dass die sensiblen Daten nicht vor gezielten Angriffen geschützt werden müssen und ausschließlich von gutmütigen Fachanwendern genutzt werden, kann auch eine mittlere Schutzstufe ausreichen.

Die richtige Auswahl der Schutzstufe ist wichtig, um das richtige Maß zu finden. Üblicherweise wächst mit höheren Schutzanforderungen auch der Implementierungsaufwand rasant in die Höhe und verursacht zusätzliche Kosten.

Tabelle 2: Sicherheitsstufe nach Schutzbedarf

Schutzbedarf Daten	Fachanwender	Power User	Hacker
sensibel	2	3	3
hoch	1-2	2	2-3
gering	1	1	2
öffentlich	0	0	0
	Schutzbedarf Angreifer		

3 Konkrete Sicherheitsmaßnahmen mit SAS realisieren

Nach dieser theoretischen Einführung sollen nun im Folgenden konkrete SAS Sicherheitsmaßnahmen vorgestellt werden und anschließend ihre Sicherheitsstufe ermittelt werden. Zuvor soll jedoch noch darauf hingewiesen werden, dass diese Maßnahmen natürlich nur dann effektiv sein können, wenn die Basisvoraussetzungen dafür erfüllt sind. So sollte es selbstverständlich sein, sowohl für das Betriebssystem wie auch SAS die aktuellen Hotfixes des Herstellers einzuspielen und als Administrator möglichst restriktive Berechtigungen zu vergeben: sowohl im Dateisystem, bei externen Datenbanken und natürlich der SAS Plattform.

3.1 Datasets verschlüsseln

Unter SAS 9.4 gibt es zwei unterschiedliche Methoden zum Verschlüsseln von SAS Datasets: zum einen die Verschlüsselung mit einer SAS proprietären Methode und zum anderen mit AES. Beide werden im Folgenden vorgestellt und separat bewertet.

3.1.1 SAS Proprietary

Die SAS proprietäre Verschlüsselung steht bereits seit SAS Version 6 zur Verfügung. Beim aktuellen Versionssprung auf SAS 9.4 hat SAS selbst die Sicherheit dieser Verschlüsselungsmethode neu bewertet.

Unter SAS 9.3 wurde zur Möglichkeit das Verschlüsselungskennwort durch ausprobieren zu erraten noch wie folgt bewertet: „*The process is time-consuming and resource-intensive*“ [1].
Ab Version 9.4 gesteht SAS ein, dass die Kennwörter mit vertretbarem Aufwand ausgespäht werden können und empfiehlt selbst für sichere Anwendungen auf die AES Verschlüsselung umzustellen:
“*This encryption **provides a medium level of security**. Users must supply the appropriate passwords to authorize their access to the data, but with the speed of today's computers, it could be subjected to a **brute force attack on the 2,563,160,682,591 possible combinations of valid password values**. Many of which must produce the same 32-bit key. SAS/SECURE and data set support of AES, which is also shipped with Base SAS software, provides a higher level of security.*” [2]

Daher wird die proprietäre Verschlüsselung an dieser Stelle nicht weiter vertieft, sondern gleich die von SAS empfohlene Verschlüsselungsmethode vorgestellt.

3.1.2 AES

AES steht für „Advanced Encryption Standard“ und ist ein weit verbreiteter und aktuell als sicher betrachteter Verschlüsselungsalgorithmus. Die Verschlüsselungsart steht mit dem Produkt SAS/SECURE zur Verfügung, das ab SAS 9.4 Teil der Base SAS Software geworden ist.
Neben dem Verschlüsselungskennwort muss der Programmierer noch explizit die Verschlüsselungsmethode AES angeben, wie das folgende Codebeispiel verdeutlicht:

```
data sasdata.class_enc(encrypt=aes encryptkey=geheim);
  set sashelp.class;
run;
```

Jedes Mal wenn nun auf das Dataset zugegriffen werden soll, muss das Kennwort zum Entschlüsseln mit angegeben werden:

```
proc print data=sasdata.class_enc(encryptkey=geheim);
run;
```

Die AES Verschlüsselung hat den Vorteil, dass die Daten nun sicher sind und ein Angreifer, der physikalischen Zugriff auf die Daten hat, diese ohne das Kennwort nicht auslesen kann. Diese Verschlüsselungsform hat jedoch auch den Nachteil, dass die Kennwörter zum Verarbeiten der Daten im Programm oder in einer Makrovariablen gespeichert werden müssen. Um dieses Kennwort geheim zu halten ist daher meist noch zusätzlicher Aufwand notwendig.

3.2 Secure Makros

Secure Makros sind eine komfortable Möglichkeit, um sensible Informationen vor dem Aufrufer zu verstecken und sicherzustellen, dass Ausführungsdetails des Makros nicht im SAS Log erscheinen.

Üblicherweise werden solche Makros von der IT bereitgestellt und enthalten beispielsweise Anmeldeinformationen von technischen Benutzern, die der aufrufende Fachanwender nicht erfahren darf. Secure Makros werden verschlüsselt in einem permanenten SAS Katalog abgelegt. Der Fachanwender muss dann nur die Bibliothek mit dem Katalog kennen und kann dann die Secure Makros wie ein normales Makro verwenden.

Das folgende Codebeispiel enthält alle notwendigen Teilschritte zur Erzeugung von Secure Makros:

```
libname mylib "C:\KSFE-Demo\public";
options mstored sasmstore=mylib;

%macro securePrint /store secure;
  proc print data=mylib.class_secure(pw=geheim);
  run;
%mend securePrint;

%macro secureCreate /store secure;
  %local geheimerInhalt;
  %let geheimerInhalt = geheim;

  data mylib.class_secure(pw=geheim);
    set sashelp.class;
    x = "dieser Text ist auch geheim";
    drop x;
  run;
%mend secureCreate;
```

Zunächst wurde eine permanente Bibliothek allokiert und diese der Option SASMSTORE übergeben. Zusätzlich muss die Option MSTORED gesetzt sein.
Anschließend werden zwei Makros definiert. Durch die zusätzlichen Makrooptionen STORE und SECURE wird SAS angewiesen die Makrodefinitionen verschlüsselt im Katalog der SASMSTORE Bibliothek abzulegen.
Die geheimen Informationen, die innerhalb der Makros verwendet werden, also Kennwörter, lokale Makrovariablen und Inhalte von temporären Datastepvariablen sollten somit vor neugierigen Augen geschützt sein.

Der Anwender bekommt nun die Information, wo der Makrokatalog abgelegt ist und kann sofern die richtigen Optionen gesetzt sind, die Secure Makros wie gewohnt verwenden:

```
libname mylib "C:\KSFE-Demo\public";
options mstored sasmstore=mylib;
%secureCreate;
%securePrint;
```

Da die Secure Makros keine Logausgabe erzeugen und der Anwender auch nicht mehr an den Quelltext des Makros herankommt, sind die sensiblen Informationen innerhalb des Makros geschützt. Soweit die Theorie ☺

In der Praxis empfiehlt es sich jedoch grundsätzlich skeptisch zu gegenüber Sicherheitsmechanismen zu sein und selbst einmal zu versuchen die Sicherheitsmaßnahmen zu umgehen. Im Fall von Secure Makros lohnt sich dieser Mehraufwand und Sie werden möglicherweise etwas Erstaunliches finden. Der folgende Screenshot zeigt einen Ausschnitt aus der Katalogdatei, die mit einem Texteditor geöffnet wurde.

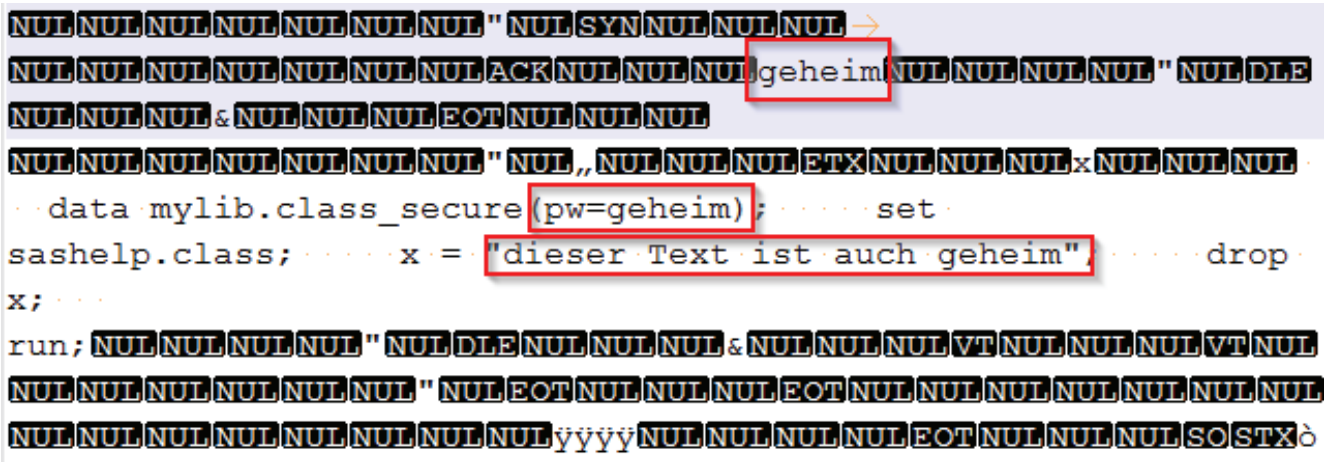

Abbildung 1: Auszug aus der Katalogdatei

Die markierten Stellen zeigen deutlich, dass die sensiblen Daten im Klartext innerhalb der Katalogdatei lesbar sind. Nach mehreren Versuchsreihen und einigen Emails mit dem SAS Support stellte sich heraus, dass es sich hierbei um einen Bug handelt.
Der Bug bewirkt, dass das zuletzt in den Katalog eingefügte Makro verschlüsselt wird. Sobald weitere Änderungen am Katalog vorgenommen werden, wird das Makro nachträglich verschlüsselt. Dieser Bug konnte sowohl unter SAS 9.3 als auch 9.4M2 reproduziert werden.

Für den Bug gibt es die folgenden zwei Workarounds:

1. Kopieren Sie einmal den Katalog mit Hilfe von PROC CATALOG einmal in einen neuen Katalog.
2. Kompilieren Sie ein weiteres Makro in den bestehenden Katalog.

Nach Durchführung eines der beiden Workarounds ist die Katalogdatei nun tatsächlich verschlüsselt und enthält keinerlei Informationen aus dem Makro im Klartext.
Allerdings ist auch hier Vorsicht geboten und man sollte sich nicht vollständig auf diese Sicherheitstechnik verlassen! Dem Autor sind Wege bekannt, um auch aus einem vollverschlüsselten Secure Makro wieder den Großteil des Makro Quelltextes zu extrahieren.

3.3 Proc PWEncode

Eine häufig falsch verwendete Sicherheitstechnik stellt der Umgang mit SAS enkodierten Kennwörtern dar.

Mit Hilfe der Prozedur PROC PWEncode lassen sich Kennwörter in ein SAS internes Format übersetzen, das SAS zur Laufzeit intern wieder in das Kennwort im Klartext umwandeln kann.

Der Aufruf, um z.B. das Kennwort „geheim“ in ein SAS enkodiertes Passwort umzuwandeln lautet wie folgt:

```
proc pwencode in="geheim" method=sas002;
run;
```

Anschließend erscheint im SAS Log das enkodierte Kennwort. Im o.g. Beispiel lautet es *{SAS002}F007332D0EF424A7454971A5*.
Dieses Kennwort kann nun anstelle des Klartextkennworts verwendet werden:

```
data _null_;
  set daten(pw="{SAS002}F007332D0EF424A7454971A5");
  put name=;
run;
```

Viele Programmierer halten diese Form der Verschlüsselung für sicher, da es aus ihrer Sicht nicht möglich ist wieder an das Kennwort im Klartext zu gelangen. Lassen Sie sich aber gesagt sein, dass man auch mit dem enkodierten Kennwort eine Menge Unfug anstellen lässt!
Aber wozu ist dann PROC PWEncode überhaupt gedacht? Ein Blick in die SAS Dokumentation hilft wie immer weiter:
“*PROC PWENCODE is intended* ***to prevent casual, non-malicious viewing of passwords****.*
You should not depend on PROC PWENCODE for all your data security needs; a determined and knowledgeable attacker can decode the encoded passwords.” [3]

Die enkodierten Kennwörter sind also nur dazu gedacht, dass man sie sich nicht einfach merken kann, wenn man beispielsweise einmal auf den Monitor des Kollegen sieht. Der Verschlüsselungsalgorithmus spielt hier keine Rolle, SAS003 und SAS004 enkodierte Kennwörter sind genauso anfällig für Missbrauch.
Gelangt ein Angreifer an ein enkodiertes Kennwort, dann kann es innerhalb der SAS Umgebung wie das Kennwort im Klartext verwendet werden. Mit etwas Aufwand ist es auch problemlos möglich einen Weg zu finden, um wieder an das Kennwort im Klartext zu gelangen.
Dem Autor sind mindestens zwei unterschiedliche Wege bekannt, um Kennwörter wieder zu dekodieren. Beim Einsatz dieser „Sicherheitsmaßnahme“ ist daher generell Vorsicht geboten!

3.4 Prompts anstelle von Kennwörtern im Code

Generell erweist es sich als sicherheitstechnisch problematisch, wenn Kennwörter in SAS Programmen gespeichert werden. Gelangt ein Angreifer an die Programme bzw.

die Logdateien der Programmausführung, kann er leicht an Kennwörter gelangen, die ihm eigentlich nicht bekannt sein dürften.
Eine bessere Methode als die Kennwörter im Programmcode zu speichern ist es die Anmeldedaten bei Bedarf per Prompt abzufragen. Das folgende Codebeispiel baut eine Verbindung zu einem SAS Share Server auf.

```
libname test "C:\temp" server=__8551 user=_prompt_
password=_prompt_;
```

Über die speziellen Schlüsselwörter _prompt_ als Platzhalter für Benutzer und Passwort, wird SAS angewiesen bei der Allokation der Bibliothek einen Anmeldedialog zu zeigen, in den der Anwender interaktiv seine Anmeldedaten eintragen kann. SAS/SHARE soll hier nur als Beispiel dienen. SAS bietet an vielen Stellen die Möglichkeit, um Anmeldedaten per Promptdialog abzufragen. Details lesen Sie am besten im Security Guide zu ihrer im Einsatz befindlichen Technologie nach.

Die meisten Promptlösungen von SAS funktionieren jedoch nur im SAS Display Manager und beispielsweise nicht im Enterprise Guide oder SAS Studio. Außerdem sollte noch erwähnt werden, dass sich diese Lösung aus offensichtlichen Gründen nicht für die Batchverarbeitung einsetzen lässt.

Mit der SAS Plattform bietet SAS mit dem Prompting Framework zwar auch Möglichkeiten, um Kennwörter per Prompt abzufragen, jedoch hat diese Lösung den Nachteil, dass die Kennwörter in Makrovariablen vorgehalten werden und auch in den SAS Logdateien erscheinen und somit nur einen geringen Schutz bieten.

4 Sicherheitsbewertung

Die vorgestellten Sicherheitsfeatures werden noch einmal inTabelle 3 zusammengefasst und einer konkreten Sicherheitsstufe zugeordnet.

Tabelle 3: Sicherheitsfeatures (SAS Foundation)

Feature	**Sicherheitsstufe**
Proc PWEncode	1
Secure Makros	1
Dataset verschlüsseln (SAS Proprietary)	2
Option NOXCMD	2
Dataset verschlüsseln (AES)	3
SAS Prompts (SAS Display Manager)	3

Das Sicherheitsfeature von **Proc PWEncode** konnte nicht überzeugen und wurde somit der niedrigsten Sicherheitsstufe zugeordnet. Auch die **Secure Makros** sind von ihrer Konzeption her zwar höchst interessant, jedoch weißt die Implementierung einige Schwächen auf, so dass eine Verwendung nicht vorbehaltlos zu empfehlen ist.
Die nächst höhere Sicherheitsstufe konnten die SAS **proprietäre Dataset-Verschlüsselung** und die Option **NOXCMD** (verhindert die Ausführung von Betriebssystembefehlen aus SAS heraus) erzielen. Die Option NOXCMD wurde im bisherigen Beitrag nicht vorgestellt, da Sie bereits zum Handwerkszeug eines jeden SAS Administrators gehören sollte und in der SAS Plattform für SAS Server bereits als Defaulteinstellung gesetzt ist. Jedoch sei an dieser Stelle ausdrücklich davor gewarnt sich allein auf diese Sicherungsmaßnahme zu verlassen. Mit entsprechendem technischen Know-how und einem mittleren Aufwand, lässt sich diese Optionseinstellung nämlich problemlos umgehen! Auch bei der gesetzten Option NOXCMD ist es möglich Betriebssystembefehle aus SAS heraus auszuführen. Dem Autor sind hierfür sogar mehrere Wege bekannt.
Wirklich überzeugen und damit die höchste Sicherheitsstufe konnten nur die **Dataset-Verschlüsselung mit AES** und die Übergabe von Anmeldedaten per **SAS Prompt** erreichen.

Zum Abschluss der Sicherheitsbewertung gibt Tabelle 4 einen kleinen Ausblick auf ausgewählte Sicherheitsfeatures der SAS Plattform, die der Autor im Rahmen eines Sicherheits-Audits mit Penetrationstests untersuchen konnte.

Tabelle 4: Sicherheitsfeatures (SAS Plattform)

Feature	**Sicherheitsstufe**
SAS Prompting Framework	1
Kennwörter (z.B. von Datenbankusern) in den Metadaten speichern	1
Metadata Capabilities	1
Row Level Security (SAS Secure Libraries)	2
SAS Server mit technischen Usern (STP, Pooled Workspace)	2
Option LOCKDOWN	3

Auch hier bestätigt sich der Gesamteindruck, dass die meisten Sicherheitsfeatures der SAS Plattform nur für geringe bis mittlere Sicherheitsszenarien geeignet sind und einem gezielten Angriff kaum standhalten. Bis auf die Option LOCKDOWN gelang es dem Autor sämtliche Sicherheitsmaßnahmen zu umgehen.
Leider konnte aus Platzgründen auf die einzelnen Plattform-Sicherheitsfeatures und ihre Schwächen nicht eingegangen werden. Sollten Sie jedoch Fragen haben und Näheres zu einzelnen Features erfahren möchten, beantwortet der Autor gerne Ihre Fragen. Die Kontaktdaten finden Sie am Anfang des Beitrags.

5 Zusammenfassung

Alle bisherigen **Erkenntnisse** lassen sich auf zwei zentrale Punkte zusammenfassen:

- Nicht alle SAS Sicherheitsfeatures schützen gleich gut
- Viele Sicherheitsmaßnahmen lassen sich durch einen versierten Angreifer komplett aushebeln

Hieraus lassen sich nun die folgenden **Empfehlungen** ableiten, um eine Sicherheitsarchitektur zu entwerfen, die auch hohe Sicherheitsanforderungen erfüllt:

- Kennwörter dürfen weder in Programmen, Makrovariablen noch in den SAS Metadaten gespeichert werden. Auch in den Logdateien dürfen keine Kennwörter erscheinen.
- Hohe Sicherheitsanforderungen lassen sich nur über das Betriebssystem und Datenbankberechtigungen erfüllen.
- Sensible Daten sollten daher am besten in eine geschützte Datenbank ausgelagert werden.

Die in diesem Beitrag vorgestellten Empfehlungen sind selbstverständlich nicht abschließend. In der SAS Onlinedokumentation stehen noch etliche Security-, Administration- und Konfiguration-Guides bereit, die ebenfalls wertvolle Hinweise zur Einrichtung und Sicherung einer SAS Installation beinhalten.
Grundsätzlich empfiehlt es sich aber andererseits auch nicht blind auf die Dokumentation zu vertrauen, sondern selbst einmal einen Blick auf die Implementierung und Konfiguration in ihrem Unternehmen zu werfen. Selbst wenn Sie sich absolut sicher sein sollten, dass ihre SAS Umgebung sicher eingerichtet wurde, finden vielleicht andere Anwender noch Lücken, an die Sie selbst nicht gedacht haben. Eine Frage an den Leser zum Schluss: Wann haben Sie zuletzt ein Sicherheits-Audit in ihrem Unternehmen durchgeführt und durch Penetrationstests sichergestellt, dass Ihr SAS Sicherheitskonzept auch hält was es verspricht?

Literatur

[1] http://support.sas.com/documentation/cdl/en/ledsoptsref/63326/HTML/default/viewer.htm#p1hwtxbozzzy4un11ldzgovfhcrf.htm

[2] http://support.sas.com/documentation/cdl/en/lrcon/67885/HTML/default/viewer.htm#n1s7u3pd71rgunn1xuexedikq90f.htm

[3] http://support.sas.com/documentation/cdl/en/secref/68201/HTML/default/viewer.htm#n0lkz988trezwln1kwk6z7sltti1.htm