Automatisierung von Penetrationstest-Berichten mittels CWE

Konstantin Knorr[1] · Thomas Brandstetter[2] · Thomas Pröll[2]
Ute Rosenbaum[2]

[1]Fachhochschule Trier
knorr@fh-trier.de

[2]Siemens CERT
thomas.brandstetter@siemens.com
thomas.proell@siemens.com
ute.rosenbaum@siemens.com

Zusammenfassung

Sicherheitsuntersuchungen – insbesondere Penetrationstests von Informationssystemen – gehören mittlerweile zu den Standard-Sicherheitsmaßnahmen in den meisten Unternehmen. Eine Möglichkeit für Anbieter solcher Untersuchungen, auf den steigenden Zeit- und Kostendruck zu reagieren, liegt in der zumindest teilweisen Automatisierung bei der Erstellung des Penetrationstest-Abschlussberichts. Der Artikel diskutiert, wie die Automatisierung über die bestehende Schwachstellensammlung „Common Weakness Enumeration“ vorgenommen werden kann. Dazu wird ein Software-Prototyp namens „FindingDesigner“ vorgestellt, und die Erkenntnisse aus seinem praktischen Einsatz werden diskutiert. Den Abschluss bilden eine Zusammenfassung der wichtigsten Erkenntnisse und die Diskussion möglicher Erweiterungen z.B. um weitere Funktionalitäten und die Aufnahme zusätzlicher Schwachstellenbeschreibungen basierend auf geeigneten Sicherheitsstandards.

1 Einleitung

Penetrationstests gehören zu den anerkannten Sicherheitsmaßnahmen, um in produktiven Systemen oder während der Entwicklung solcher Systeme die Informationssicherheit zu überprüfen und durch die Umsetzung der empfohlenen Maßnahmen das Sicherheitsniveau zu erhöhen. Penetrationstests werden typischerweise von unabhängigen internen oder externen Teams durchgeführt. Für die Auftraggeber von Penetrationstests wie z.B. die Sicherheitsbeauftragten der Unternehmen oder Entwicklungsleiter von Produkten und Systemen sind u.a. folgende Punkte wichtig: (1) die Kosten solcher Tests, (2) der „Abdeckungsgrad“ der Untersuchung, d.h., welche Schwachstellen in welchen Systemteilen wurden entdeckt und wie viele Schwachstellen verbleiben unentdeckt, und (3) die Einhaltung gesetzlicher oder regulatorischer Vorgaben.

Außer in den klassischen Büroumgebungen werden Penetrationstests in immer stärkerem Maße auch innerhalb der IT-Systeme unserer kritischen Infrastruktur durch Kunden und Regularien gefordert und durchgeführt. Beispiele sind das Gesundheitswesen, die Energieversorgung und Produktionssysteme, vgl. [NERC, WIB]. Ein Trend geht weg von großen, umfangreichen

P. Schartner · J. Taeger (Hrsg.) · DACH Security 2011 · syssec (2011) pp-pp.

hin zu immer kürzeren Untersuchungen getrieben durch den Druck zur Kosteneinsparung. Gleichzeitig steigen die Anforderungen der Auftraggeber bzgl. der Dokumentation der durchgeführten Tests und beim Reporting.

Anbieter von Penetrationstests müssen sich an den Zeit- und Kostendruck anpassen. Eine Möglichkeit dazu liegt in der automatisierten Erstellung des Abschlussberichts. Der in diesem Artikel beschriebene Ansatz zur Automatisierung beruht auf der Übernahme relevanter Teile aus bestehenden Schwachstellensammlungen. Ein prominentes Beispiel für eine solche ist die Common Weakness Enumeration [CWE], ein Standardisierungsprojekt für die Beschreibung von Sicherheitsschwachstellen getrieben durch die amerikanische MITRE-Cooperation.

Generische Teile eines Findings wie z.B. ein Vorschlag für den Titel, eine allgemeine Beschreibung der Schwachstelle, mögliche Gegenmaßnahmen und Beispielcode können aus der Datenbank übernommen werden. Dadurch wird ein Großteil der sich wiederholenden Bestandteile eines Findings standardisiert. Die knappe zur Verfügung stehende Zeit können die Tester für die Beschreibung der systemspezifischen Teile der Schwachstelle nutzen.

Der Artikel hat folgende Struktur: Nach der Einleitung liefert Kapitel 2 grundlegende Informationen zu den Themen Penetrationstests und CWE. Ferner beschreibt Kapitel 2.3 verwandte Arbeiten. Kapitel 3 bildet den Schwerpunkt des Beitrags: Kapitel 3.1 diskutiert, wie die Findings eines Penetrationstest-Bericht strukturiert werden können. In Kapitel 3.2 wird die Funktionsweise und technische Umsetzung des FindingDesigners vorgestellt. Die praktischen Erfahrungen beim Einsatz des Tools beschreibt Kapitel 3.3. Den Abschluss des Artikels in Kapitel 4 bildet ein Fazit der wichtigsten Erkenntnisse und ein Ausblick auf zukünftige Arbeiten. Darin werden insbesondere mögliche Erweiterungen des FindingDesigners beschrieben.

2 Grundlagen

2.1 Penetrationstests

Als Penetration-Testing wird der kontrollierte Versuch verstanden, von außen in ein bestimmtes Computersystem bzw. -netzwerk einzudringen, um Schwachstellen zu identifizieren. Dazu werden die gleichen oder ähnliche Methoden eingesetzt, die auch bei einem realen Angriff verwendet werden. Die hierbei identifizierten Schwachstellen können durch entsprechende Maßnahmen behoben werden, bevor diese von unautorisierten Dritten genutzt werden können [BSI03]. Daher wird für solche Tests auch der Begriff „Friendly Hacking“ verwendet.

Die Ergebnisse der Penetrationstests und möglicher Gegenmaßnahmen werden in einem Bericht zusammengefasst und dem Auftraggeber bei einem Treffen präsentiert. Ziele für die Beauftragung eines Penetration-Testings sind u.a.:

- Überblick über den generellen Sicherheitszustand eines Systems
- Identifikation konkreter Sicherheitslücken in dem System
- Einschätzung des Schadenpotentials der identifizierten Lücken
- Einhaltung gesetzlicher oder regulatorischer Vorgaben

Die Klassifikation von Penetrationstests kann wegen der Komplexität und Unterschiedlichkeit der untersuchten Systeme anhand verschiedener Kriterien vorgenommen werde. Abb. 1 gibt eine Übersicht. Weitere für diesen Artikel wichtige Unterscheidungskriterien sind:

- Der Umfang bzgl. Zeit und Kosten der Untersuchung (auch „Assessments“ genannt): Kurz- oder Lang-Assessments
- Zusammensetzung des Teams und Kenntnisstand der einzelnen Tester
- Untersuchungstiefe: „Tiefensuche“ bzw. „Breitensuche“

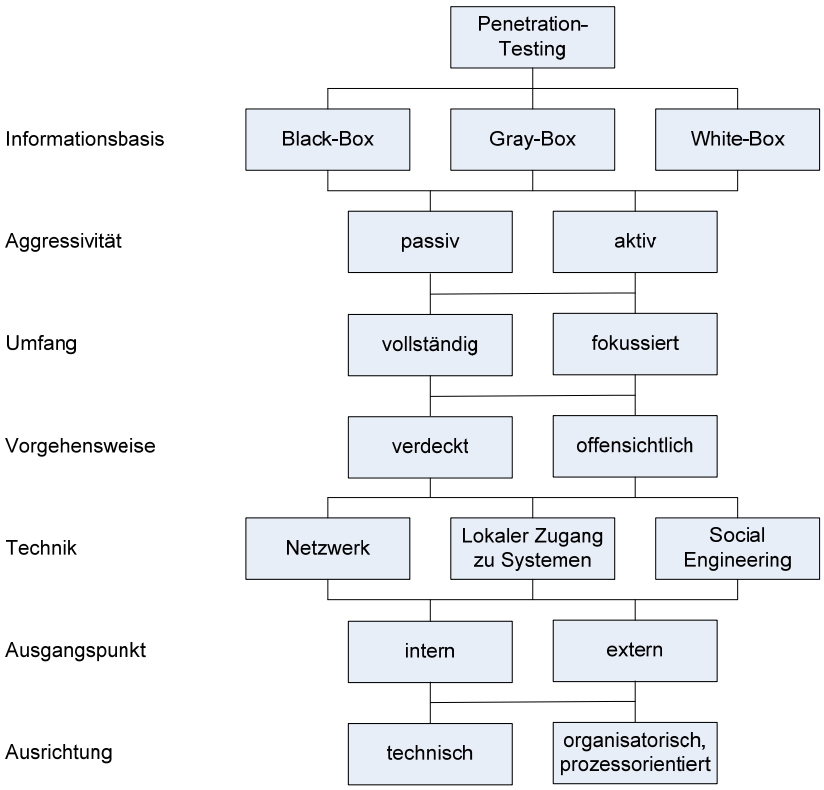

Abb. 1: Klassifikation von Penetrationstests, angelehnt an [BSI03]

Die Komplexität der heutigen Informationssysteme erlaubt leider keine vollständige Überprüfung dieser Systeme, so dass Penetrationstests nur ausgewählte Teile betrachten können. Auch die systematische Sicherheitsüberprüfung und Zertifizierung etwa nach den Common Criteria [CC] fokussiert typischerweise auf ausgewählte Teile eines Systems („Target of Evaluation“) und gibt keine detaillierte Vorgehensweise für die praktische Überprüfung vor.

Die Standardisierung von Penetrationstests ist eine große Herausforderung. Ein allgemein akzeptierter Standard fehlt. Penetrationstests können theoretisch ohne grundlegende Methode durchgeführt werden. Solche Tests, die dann stark von der Expertise und der Arbeitsweise des Testers abhängen, werden oft als “Freestyle-Hacking“ bezeichnet. Das Freestlye-Hacking hat seine Berechtigung, z.B. um neuartige Schwachstellen aufzudecken. Im Sinne der Nachvollziehbarkeit, Vergleichbarkeit und Vollständigkeit der Tests ist es jedoch empfehlenswert, sich einer geeigneten Methode zu bedienen.

Bekannte Penetrationstest-Methoden für Informationssystemen sind [ISSAF, NIST, OSSTMM]; speziell für Web-Applikationen wurde [OWASP] entwickelt. Allerdings beschränken sich diese Quellen auf die Beschreibung des Vorgehens, ohne detaillierte Vorgaben zu den Tests und Testverfahren zu machen, und lassen dem Tester so den nötigen Freiraum, seine Kreativität und seinen Spürsinn auszuleben. Ein ausführlicher Vergleich bestehender Penetrations-Testing-Methoden findet sich in [Kno11].

Auch für den Ablauf eines Penetrationstests gibt es kein einheitliches Vorgehen. Grob lassen aber die in Abb. 2 dargestellten Phasen unterscheiden.

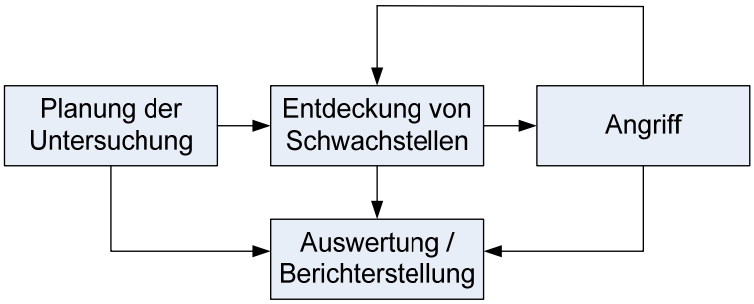

Abb. 2: Ablauf einer Penetrationstest-Untersuchung, angelehnt an [NIST]

Die Erstellung des Abschlussberichts liegt in der Auswertungs-Phase. Er fasst die Ergebnisse des Penetrationstests zusammen und beinhaltet neben einer Management-Zusammenfassung, Informationen über das Projekt, einer Beschreibung der verwendeten Methode, u.v.m. (vgl. [Alha10]) die in der Untersuchung identifizierten Schwachstellen – im Folgenden "Findings" genannt. Abschnitt 3.1 geht näher auf die Finding-Struktur ein. Üblicherweise wird der Abschlussbericht durch weitere Dokumente ergänzt wie z.B. durch Reports der eingesetzten Tools und den der Untersuchung zugrunde liegenden Vertrag. Eine große Schwierigkeit liegt im zu verwendenden Detaillierungsgrad des Berichts, da der Bericht typischerweise von unterschiedlichen Zielgruppen gelesen wird, z.B. dem Auftraggeber aus dem Management und den für die technische Umsetzung der Lösung verantwortlichen Mitarbeiter. Der Abschlussbericht wird dem Auftraggeber übergeben und meist zusätzlich bei einem Treffen präsentiert, um eventuell auftretende Fragen diskutieren zu können.

2.2 Common Weakness Enumeration

CWE basiert auf dem älteren Standard CVE[1] (Common Vulnerabilities and Exposures) und anderen akademischen und Industrie-Sammlungen von Computer- und Netzwerk-Schwachstellen. Seit 1995 wird in von der MITRE-Cooperation geführten Projekten daran gearbeitet, mit CWE einen Standard mit folgenden Zielsetzungen zu schaffen:

- Schaffung einer gemeinsamen Sprache zur Beschreibung von Software-Sicherheitsschwachstellen in Architektur, Design und Code
- Schaffung einer einheitlichen Vergleichsgrundlage für Sicherheits-Tools beim Aufdecken von Schwachstellen
- Schaffung eines Standards für die Schwachstellenidentifikation, -beseitigung und -verhinderung

CWE bietet folgende Vorteile: CWE ist sehr umfangreich, detailliert und wird regelmäßig aktualisiert und erweitert. CWE beinhaltet eine detaillierte, standardisierte und vom Tester unabhängige Beschreibung der Schwachstellen. Zudem erlaubt CWE einen feingranularen Zugriff auf ausgewählte Teile der Schwachstellenbeschreibung durch die Verwendung einer XML-Struktur und eine kostenfreie Nutzung.

[1] http://cve.mitre.org/

Das Herzstück der CWE ist eine XML-Schema-Definition (XSD) und die damit vorgenommene detaillierte Beschreibung von zuletzt 863 Schwachstellen in einer XML-Datei (vgl. [CWE]). Abb. 3 zeigt die Gruppe „Common_Attributes“ aus dem Schema, die für die detaillierte Beschreibung der Schwachstellen genutzt wird. Abb. 3 verdeutlicht auch, wie aufwändig Schwachstellen mittels CWE beschrieben werden können. Viele Einträge unterteilen sich weiter in Untereinträge. Allerdings sind die meisten Einträge optional.

```
- <xs:group name="Common_Attributes">
  + <xs:annotation>
  - <xs:sequence>
    + <xs:element name="Description" minOccurs="1" maxOccurs="1">
      <xs:element ref="Relationships" minOccurs="0" maxOccurs="1" />
      <xs:element ref="Relationship_Notes" minOccurs="0" maxOccurs="1" />
    + <xs:element name="Weakness_Ordinalities" minOccurs="0" maxOccurs="1">
    + <xs:element name="Applicable_Platforms" minOccurs="0">
      <xs:element ref="Maintenance_Notes" minOccurs="0" maxOccurs="1" />
    + <xs:element name="Background_Details" minOccurs="0" maxOccurs="1">
      <xs:element ref="Other_Notes" minOccurs="0" maxOccurs="1" />
      <xs:element ref="Alternate_Terms" minOccurs="0" maxOccurs="1" />
    + <xs:element name="Terminology_Notes" minOccurs="0" maxOccurs="1">
    + <xs:element name="Time_of_Introduction" minOccurs="0" maxOccurs="1">
    + <xs:element name="Modes_of_Introduction" minOccurs="0" maxOccurs="1">
    + <xs:element name="Enabling_Factors_for_Exploitation" minOccurs="0" maxOccurs="1">
    + <xs:element name="Likelihood_of_Exploit" minOccurs="0">
    + <xs:element name="Common_Consequences" minOccurs="0">
    + <xs:element name="Detection_Methods" minOccurs="0" maxOccurs="1">
    + <xs:element name="Potential_Mitigations" minOccurs="0" maxOccurs="1">
    + <xs:element name="Causal_Nature" minOccurs="0">
    + <xs:element name="Demonstrative_Examples" minOccurs="0" maxOccurs="1">
    + <xs:element name="Observed_Examples" minOccurs="0" maxOccurs="1">
    + <xs:element name="Theoretical_Notes" minOccurs="0" maxOccurs="1">
    + <xs:element name="Functional_Areas" minOccurs="0" maxOccurs="1">
    + <xs:element name="Relevant_Properties" minOccurs="0">
    + <xs:element name="Affected_Resources" minOccurs="0" maxOccurs="1">
      <xs:element ref="Research_Gaps" minOccurs="0" maxOccurs="1" />
    + <xs:element name="References" minOccurs="0" type="Reference_List_Type">
    + <xs:element name="Taxonomy_Mappings" minOccurs="0" maxOccurs="1">
    + <xs:element name="White_Box_Definitions" minOccurs="0">
    + <xs:element name="Black_Box_Definitions" minOccurs="0">
    + <xs:element name="Related_Attack_Patterns" minOccurs="0">
      <xs:element ref="Content_History" minOccurs="1" />
    </xs:sequence>
  </xs:group>
```

Abb. 3: Auszug aus der CWE XML-Schema-Definition

2.3 Verwandte Arbeiten

Viele Sicherheits-Tools wie z.B. Nessus[2] erlauben das Generieren eines Reports, oft in unterschiedlichen Detaillierungsgraden für unterschiedliche Adressaten. Die meisten Anbieter von Penetrationstests verwenden zur Berichterstellung proprietäre Tools, welche ggf. Reports von den beim Assessment eingesetzten Sicherheits-Tools in den Bericht integrieren. MagicTree[3] und Dradis[4] sind Tools zum Erstellen von Penetrationstest-Berichten, welche die Integration mehrerer anderer Tools erlauben, allerdings ohne dabei CWE zu verwenden.

CWE wird von einigen Sicherheits-Tools z.B. von Source-Code-Analyse-Tools oder von Fuzzing-Tools schon genutzt[5], allerdings nicht von allgemeinen Tools zur Erstellung von Berichten. Nach Wissen der Autoren ist dies daher der erste Versuch, CWE systematisch zur Erstellung generischer Penetrationstest-Berichten zu verwenden.

---

[2] http://www.tenable.com/products/nessus

[3] http://www.darknet.org.uk/2011/01/magictree-penetration-tester-productivity-tool/

[4] http://dradisframework.org/

[5] http://cwe.mitre.org/compatible/organizations.html

3 FindingDesigner

Den Schwerpunkt des Artikels bildet das folgende Kapitel, in dem ein Software-Prototyp namens „FindingDesigner“ vorgestellt wird. Der Prototyp dient der Unterstützung und Automatisierung bei der Erstellung von Penetrationstest-Berichten, indem er die bestehende Schwachstellensammlung CWE nutzt. Dafür ist es erforderlich, eine generische Finding-Struktur zu erstellen und diese mit der CWE-Struktur abzugleichen (Kapitel 3.1). Die „Schnittmenge“ wird den Erstellern des Berichts durch den FindingDesigner angeboten. Kapitel 3.2 beschreibt die Funktionsweise und technische Umsetzung. Den Abschluss bildet eine Beschreibung der Erfahrungen beim praktischen Einsatz der Software in Kapitel 3.3.

Tab. 1: Felder eines Findings, deren Beschreibung, Häufigkeit („H“ = häufig, „G“ = gelegentlich verwendet) und passende Einträge aus dem CWE-Schema

Feld	Beschreibung des Felds	Häufigkeit	Entsprechung im CWE-Schema, vgl. Abb. 3
Name des Findings	Eindeutiger Namen des Findings	H	CWE-Name, Alternate Terms
Fundort	Ort innerhalb des untersuchten Systems, an dem die Schwachstelle gefunden wurde	G	Applicable Platforms, Time of Introduction
Kritikalität	Oft auch Priorität oder Risiko genannt. Gibt das Schadenspotential der Schwachstelle. Setzt sich aus “Impact” und “Likelihood” zusammen	H	Likelihood of Exploit
Beschreibung der Schwachstelle	Beschreibung der Schwachstelle. Detaillierungsgrad abhängig vom Zielpublikum	H	Descripition, Common Consequences, Related Attack Patterns, Mode of Introduction, Demonstrative and observed Example, Black and White Box Definition
Gegenmaßnahmen	Beschreibung der empfohlenen Maßnahmen zur Beseitigung der Schwachstelle	H	Potential Mitigations
Verwendete Tools	Listet die zum Generieren des Findings verwendeten Tools auf	G	-
Status	Ist die Schwachstelle noch offen oder bereits (teilweise) behoben?	G	-
Referenzen	Referenzen und Beziehungen zu anderen Findings	G	References, Weakness Ordinalities, Taxonomy Mappings

3.1 Finding-Struktur

Ein wesentlicher Teil des Penetrationstest-Berichtes ist die Beschreibung der gefundenen Schwachstellen. Hierzu wird meist eine proprietäre Struktur z.B. in Form einer Tabelle verwendet. Da es keinen Standard gibt, werden die unterschiedlichsten Strukturen und Bezeich-

nungen der Felder verwendet. Bei der Durchsicht von Beispielberichten und von [Alha10, BrKR09, BrKR10] konnten jedoch häufig verwendete Felder identifiziert werden. Tab. 1 listet die wichtigsten Felder eines Findings auf, gibt eine Beschreibung und unterscheidet zwischen häufig (H) und gelegentlich (G) genutzten Feldern. Ferner werden zu den Feldern passende Einträge des CWE-Schemas (vgl. Abb. 3) identifiziert.

Theoretisch können alle (gefüllten) CWE-Schema-Einträge zur Beschreibung einer Schwachstelle genutzt werden. In der Praxis setzen die Lesbarkeit des Berichts und der gewünschte Detaillierungsgrad Grenzen. Im FindingDesigner kann die Struktur pro Finding konfiguriert werden. Abb. 4 zeigt ein Beispiel für die Umsetzung in Feld (5).

Die meisten Felder insbesondere die Beschreibung und die Gegenmaßnahmen haben einen Finding-spezifischen und einen generischen Teil. Durch den FindingDesigner kann natürlich nur der generische Teil unterstützt werden. Der spezifische Teil muss vom Tester weiterhin manuell bearbeitet werden.

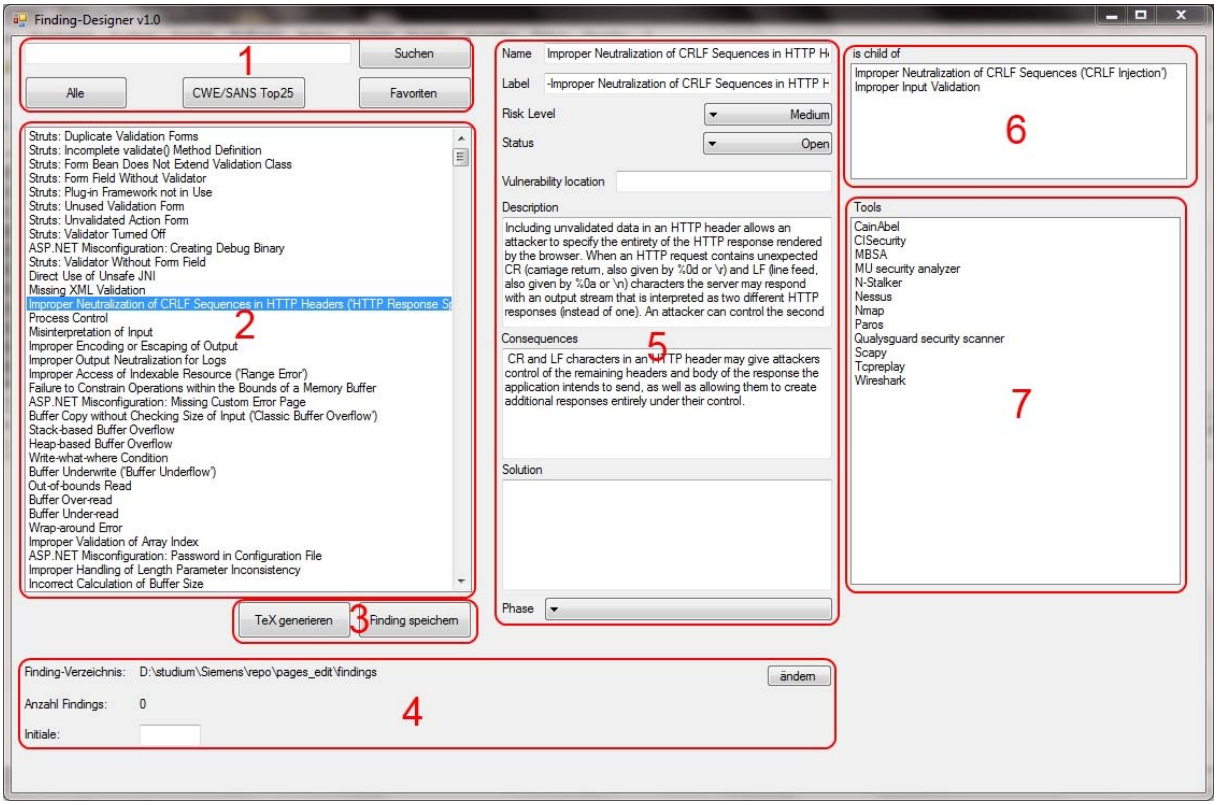

Abb. 4: Grafische Benutzeroberfläche des FindingDesigners

3.2 Funktionsweise und technische Umsetzung

Der FindingDesigner ist ein in C# entwickeltes Programm basierend auf Microsofts .NET-Framework und läuft unter Windows XP SP3 und Windows 7. Er lässt sich auf dem TeXnic-Center[6] starten und generiert TeX-Dateien für die generierten Findings, die zusammen mit einem bestehenden LaTex-Report-Template in einen PDF-Bericht überführt werden. Als Grundlage für seine Ausführung benötigt der FindingDesigner die aktuellen CWE XML- und

[6] http://www.texniccenter.org/

XSD-Dateien. Abb. 4 zeigt die Benutzeroberfläche des FindingDesigners, die anhand der sieben markierten Bereiche erläutert wird:

(1) Über die Eingabefelder „Alle“, „CWE/SANS Top25“, „Favoriten“ werden alle CWE-Einträge, die Top25 nach der CWE/SANS Empfehlung oder die vom jeweiligen Tester am häufigsten verwendeten Schwachstellen geladen. Mit „Suchen“ können die geladenen Schwachstellen durchsucht und gefiltert werden.
(2) Darstellung der in (1) ausgewählten CWE-Einträge.
(3) „Finding speichern“ bzw. „TeX generieren“ erzeugt bzw. speichert eine TeX-Datei für das aktuelle Finding mit allen unter (5) und (7) angegebenen Informationen.
(4) Das „Finding-Verzeichnis“ gibt den Speicherort der Finding-Dateien an. „Anzahl Findings“ gibt die aktuelle Anzahl gespeicherter Findings pro Sitzung. Die Initialen des Testers werden genutzt, um mehrere Findings mit gleicher CWE-ID unterschiedlicher Tester in einem Bericht unterscheiden zu können.
(5) Vom Programm mit Daten aus CWE vorausgefüllte Eingabemaske für die Inhalte der Finding-Felder. Die Text-Felder sind vom Benutzer frei editierbar.
(6) In CWE hinterlegte „ChildOf“-Beziehung der aktuell ausgewählten Schwachstelle. Dies verdeutlicht die Zusammenhänge der einzelnen CWE-IDs und erlaubt ein Browsen in der CWE-Hierarchie.
(7) Erlaubt eine Auswahl der für das aktuelle Finding verwendeten Sicherheit-Tools.

3.3 Erfahrungen aus dem praktischen Einsatz

Dieser Abschnitt beschreibt die praktischen Erfahrungen beim Einsatz des FindingDesigners. Das Tool wurde dazu über mehrere Monate von unterschiedlichen Testern in verschiedenen Penetrationstests eingesetzt. Der FindingDesigner wurde entwickelt, um beim Verfassen der Reports eine Zeit- und Kostenersparnis zu erreichen und eine bessere Vergleichbarkeit mit anderen Reports sicherzustellen. Ob dies gelungen ist, wird hier qualitativ beschrieben. Im praktischen Einsatz zeigen sich drei unterschiedliche Gruppen von Findings:

- Findings, die weiter manuell beschrieben werden müssen
- Findings, die teilunterstützt beschrieben werden können
- Findings, die fast vollautomatisch beschrieben werden

3.3.1 Manuelle Findings

Der FindingDesigner basiert auf der CWE-Datenbank, die eine Vielzahl von Schwachstellen beinhaltet, einige hingegen nicht. Als Beispiel können etwa überflüssige Firewall-Regeln gesehen werden, z.B. eine Regel, die einen Port öffnet, und an anderer Stelle eine andere Regel, die den gleichen Port wieder schließt. Dies erlaubt kein Eindringen, da die Konfiguration mit geschlossenem Port sicher konfiguriert ist. Entsprechend existiert keine CWE-ID und der FindingDesigner bietet keine Unterstützung beim Verfassen des Berichts. Dennoch müssen solche Fehlkonfigurationen dokumentiert werden, denn offenbar fehlen die Prozesse, die eine sichere Konfiguration sicherstellen, etwa ein regelmäßiges Review und eine saubere Dokumentierung der Firewall-Regeln. Diese Dokumentation muss zusammen mit den Gegenmaßnahmen manuell, ohne die Hilfe des FindingDesigners, erstellt werden.

Ein weiteres Beispiel wäre eine Out-of-the-box Installation des Betriebssystems. Dies kann man wieder als einfaches Prozessfinding sehen: Offenbar sind die Deployment-Prozesse bei dem Zielsystem mangelhaft. Niemand hat sich darum gekümmert, die lokale Firewall anzu-

schalten, Ports abzuschalten oder Passwörter zu ändern. Das Aufspalten dieses einzelnen Prozessfindings in mehrere Dutzend technische Findings würde den Bericht sehr unübersichtlich machen und zudem ein organisatorisches Problem als technisches Problem darstellen. Da sich CWE aber nur auf technische Schwachstellen und nicht z.B. auf organisatorische Schwachstellen bezieht, müssen solche Findings manuell erstellt werden. Ein weiteres Finding dieser Kategorie wäre das Feststellen nicht eingespielter Sicherheits-Patche auf dem System. Rein technisch ließe sich dies auf eine Vielzahl von CWE-IDs abbilden. In den Berichten taucht es allerdings als einzelnes Prozess-Finding auf.

Diese erste Gruppe von Findings liegt also auf nicht-technischer Ebene oder ist eine Zusammenfassung mehrerer Schwachstellen und damit zu generisch für die CWE-Datenbank.

3.3.2 Teilunterstützte Findings

Eine zweite Kategorie von Findings findet sich zwar in der CWE-Datenbank, die Beschreibung ist jedoch sehr generisch und allgemein gehalten. Eine solche Beschreibung alleine hilft dem Leser des Reports nicht, da zu wenige spezifische Informationen enthalten sind.

Solche Findings benötigen eine Nacharbeit, nachdem sie vom FindingDesigner entworfen wurden. Systemspezifische Details, Screenshots, der exakte Angriffsweg u.v.m. müssen nachgetragen werden. Als Beispiel sei hier das Cross Site Scripting angeführt: Man kann natürlich die generischen Felder zur Beschreibung von Cross Site Scripting übernehmen. Die Details des konkreten Findings (Welche URL? Welche Parameter? User Interaktion?) sowie der Auswirkungen (spezifisch für die untersuchte Web Applikation) müssen manuell eingetragen werden.

3.3.3 Automatisierte Findings

Der FindingDesigner kann seine Stärken voll ausspielen bei

- Einfach zu reproduzierenden Findings
- Findings in „Überblicks-Assessment“, d.h. Untersuchungen, die in die Breite statt die Tiefe gehen

Leicht zu reproduzierende Findings, wie häufig bei Flooding oder Fuzzing sind meist Input-Validierungs-Probleme. Die Beschreibung und die Gegenmaßnahmen aus der CWE-Datenbank sind zwar generisch, aber da der Penetrationstester die Software nicht exakt kennt, muss er sich speziell bei den Gegenmaßnahmen recht allgemein halten. Es reicht also aus, das verwendete Flooding- oder Fuzzing-Tool sowie den verwundbaren Dienst in das Finding-Template einzutragen, d.h. der Automatisierungsgrad ist sehr hoch (vgl. Abb. 5, durch den Tester hinzugefügter Text ist anders eingefärbt).

Ähnlich ist es bei den Findings in Überblicks-Assessments. Viele Penetrationstests haben die Zielsetzung statt einer „Tiefensuche“ nach wenigen, sehr präzise beschriebenen Schwachstellen stattdessen eine „Breitensuche“ zu vollziehen. In diesem zweiten Ansatz werden wenige Details über den Angriff gesammelt und dokumentiert. Da individuelle Informationen rar sind, muss auch der Report allgemein gehalten werden. Die generierten Templates werden nur mit wenigen, individuellen Details versehen, die Nacharbeiten sind minimal. Assessments, deren Fokus mehr der Überblick ist, können gut mit den FindingDesigner dokumentiert werden.

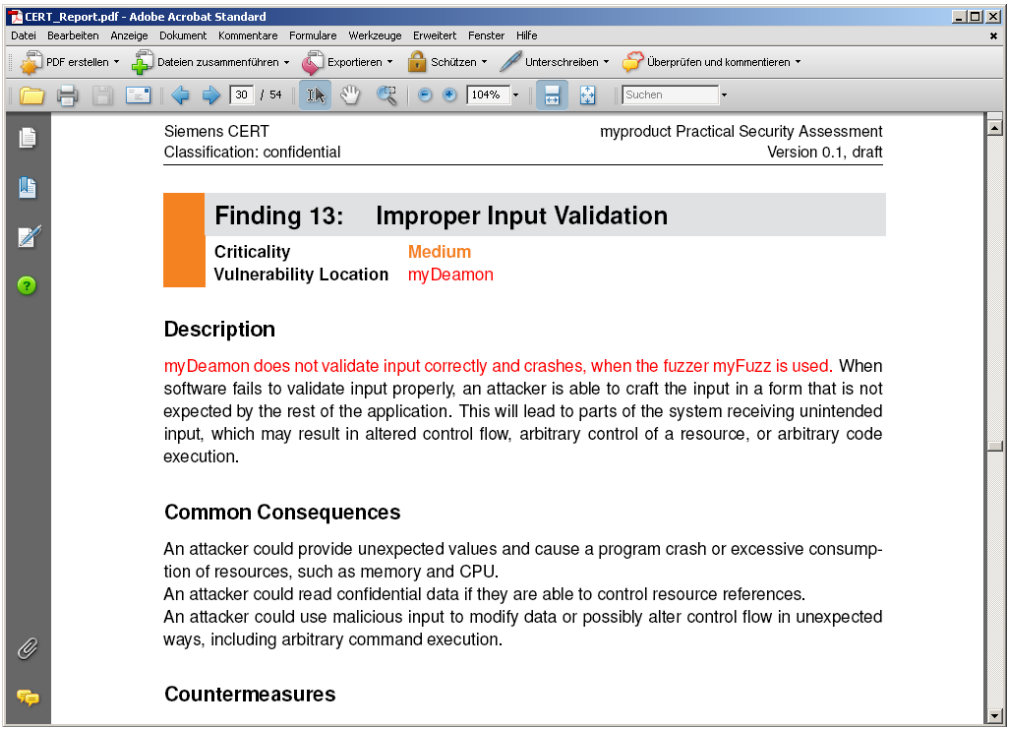

Abb. 5: Beispiel eines mit dem FindingDesigner generierten Findings

3.3.4 Zusammenfassung der praktischen Erfahrungen

Wie stark die Unterstützung durch den FindingDesigner ist, hängt von der Ausrichtung des Penetrationstests ab (vgl. Abschnitt 2.1 und Abb. 1). „Freestyle Hacking“ wird immer Freestyle Ergebnisse produzieren, die nur generisch in der CWE-Datenbank zu finden sind. Während Software-Fehler sehr gut durch CWE beschrieben werden, sind Schwachstellen in den Prozessen und der Organisation in CWE nicht oder nur unzureichend hinterlegt. Beschränkt man sich aber auf technische Untersuchungen, so bietet der FindingDesigner eine gute Unterstützung, die das Schreiben des Reports erheblich erleichtert und vereinheitlicht. Speziell bei Kurz-Assessments mit knappen Zeit- und Kostenrahmen und einer Breitensuche nach Schwachstellen ist das Potential des FindingDesigners enorm. Dort, wo die Untersuchungstiefe gering ist, wird der Report fast ohne Nacharbeiten erstellt.

Eine weitere Erfahrung ist, dass das Schreiben des Reports sich durch den FindingDesigner erheblich verändert hat: Es wird nicht mehr frei geschrieben. Stattdessen werden die vorgegebenen Textblöcke vom Autor gelesen, auf Anwendbarkeit und Sinn überprüft, und dann erst umgestellt, gestrichen oder ergänzt. Speziell für Tester, die erst wenig Erfahrung mit Penetrationstests haben, erfüllt der FindingDesigner damit auch einen edukativen Zweck, in dem er den Tester zur Verwendung professioneller Formulierungen und einer geeigneten Struktur anhält.

Eine zusätzliche Erkenntnis ist, dass die CWE-Komplexität eine hohe Einstiegshürde für den Anwender darstellt. Es ist bedingt durch die hohe Anzahl der CWE-IDs und deren komplexen Beziehungen untereinander anfangs schwierig die passende CWE-ID zu einer identifizierten Schwachstelle zu finden. Der FindingDesigner unterstützt dies durch die Suchfunktion, die „CWE/SANS Top 25“- Auswahlmöglichkeit und das Auflisten der Beziehungen zwischen den CWE-IDs (vgl. (1) und (6) in Abb. 4).

4 Fazit und Ausblick

4.1 Fazit

Der vorliegende Artikel bietet zunächst eine Einführung in die Themen Penetrationstests und CWE. Den Hauptteil bildet die Beschreibung des FindingDesigners und der Erkenntnisse seines praktischen Einsatzes. Bei technischen Software-Schwachstellen bietet der FindingDesigner einen hohen Automatisierungsgrad, Prozess- oder organisatorische Findings werden weniger gut unterstützt. Der FindingDesigner liefert schnell und einfach einen Vorschlag für wichtige generische Finding-Felder, vor allem für die Beschreibung der Schwachstelle und geeignete Gegenmaßnahmen. Systemspezifische Teile eines Findings müssen weiterhin vom Tester eingetragen werden. Vor allem bei Untersuchungen, die in die Breite statt in die Tiefe gehen, hilft der FindingDesigner. Als zusätzlichen Aspekt hat sich die Ausbildung neuer Penetrationstester mit Hilfe des FindingDesigners als vorteilhaft erwiesen.

4.2 Ausblick

Zukünftige Arbeiten fokussieren auf die folgenden Bereiche: Ein Vorteil der Verwendung von CWE liegt in der Vergleichbarkeit der Findings mehrerer Assessments. Dies kann für verschiedene Zwecke verwendet werden: z.B. zum Vergleich des Sicherheitsniveaus der untersuchten Systeme, zur Messung des Abdeckungsgrads einer Untersuchung (Wie viele Findings wurden zu welcher CWE-ID gefunden?), zur exakteren Definition der Tiefe und des Umfangs einer Untersuchung (Nach welchen CWE-IDs soll im zu untersuchenden System getestet werden?) und zur besseren Zusammensetzung der Penetrationtesting-Teams, denn Tester sind oft auf gewisse CWE-IDs spezialisiert.

Ein weiterer Bereich ist die technische Erweiterung des FindingDesigners.

- Flexiblere Verwendung des CWE-XML-Schemas etwa pro Finding über eine direkte Auswahl der Finding-Felder aus dem CWE-XML-Schema.
- Berücksichtigung eigener, individuell erstellter Findings, die teilweise auf CWE beruhen oder individuell auf Basis von Testerfahrungen erstellt wurden.
- Portierung des FindingDesigners von TeX auf Microsoft Office.
- Sprachunterstützung von deutsch und englisch.
- Erweiterung des FindingDesigner um weitere Standards und Schwachstellensammlungen. Mögliche Ansätze sind z.B. CAPEC[7] zur genaueren Beschreibung des Vorgehens beim Entdecken der Schwachstelle, CVSS[8] für die genauere Beschreibung der Kritikalität einer Schwachstelle, ISO 27002 für organisatorische Schwachstellen und der in [FiRP11, PaRF11] beschriebene Ansatz für Prozess-Schwachstellen.

Literatur

[Alha10] M. A. Alharbi: Writing a Penetration Testing Report, In: SANS Reading Room, http://www.sans.org/reading_room/whitepapers/bestprac/writing-penetration-testing-report_33343 (Zugriff am 19.6.2011)

[7] http://capec.mitre.org/

[8] http://www.first.org/cvss/

[BrKR09] T. Brandstetter, K. Knorr, U. Rosenbaum: A Structured Security Assessment Methodology for Manufacturers of Critical Infrastructure Components, IFIP SEC 2009, Cyprus

[BrKR10] T. Brandstetter, K. Knorr, U. Rosenbaum: Manufacturer-Specific Security Assessment Methodology for Critical Infrastructure Components, Fourth IFIP WG11.10 International Conference on Critical Infrastructure Protection, Washington, DC, USA, March 15-17, 2010

[BSI03] Bundesamt für Sicherheit in der Informationstechnik: Studie zur Durchführungskonzept für Penetrationstests, 2003

[CC] Common Criteria and Common Evaluation Methodology, Version 3.1, http://www.commoncriteriaportal.org/cc/ (Zugriff am 17.4.2011)

[CWE] Common Weakness Enumeration, http://cwe.mitre.org/, aktuelle XML- und XSD-Version: http://cwe.mitre.org/data/index.html (Zugriff am 16.4.2011)

[FiRP11] B. Fichtinger, W. Russwurm, P. Panholzer: Process Models and Implementation Examples for the Development of Secure Products, SEPG Europe 2011 Conference, Contribution #1780

[ISSAF] Open Information Systems Security Group, Information Systems Security Assessment Framework (ISSAF), Version 0.2, May 2006, http://www.oissg.org/issaf (Zugriff am 16.4.2011)

[Kno11] K. Knorr: Überblick bestehender Security-Assessment-Methoden, Projektbericht, Siemens, April 2011

[NERC] North American Electric Reliability Council, Critical Infrastructure Protection (CIP), http://www.nerc.com/ (Zugriff am 16.4.2011)

[NIST] National Institute of Standards and Technology, NIST Special Publication 800-115, Technical Guide to Information Security Testing and Assessment, Sep. 2008, http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf (Zugriff am 16.4.2011)

[OSSTMM] ISECOM, Open Source Security Testing Methodology Manual, v3.0, http://www.isecom.org/osstmm/ (Zugriff am 16.4.2011)

[OWASP] Open Web Application Security Project, Testing Guide, v3.0, 2008, http://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf (Zugriff am 16.4.2011)

[PaRF11] P. Panholzer, W. Russwurm, F. Fichtinger: How to Develop Secure Products Using CMMI and Some “Extras”. SEPG North America 2011 Conference, Contribution #1740

[WIB] International Instrument Users' Association, Process control Domain: Security Requirements for Vendors, Version 2.0, October 2010, http://www.wib.nl/download.html (Zugriff am 16.4.2011)