Bedrohungsmodellierung (Threat Modeling) in der Softwareentwicklung

Fabian Schwab, B.Sc.; Alexander Findeisen;
Peter Sakal, B.Sc.; Prof. Dr. Hartmut Pohl

Informationssicherheit, Fachbereich Informatik
Hochschule Bonn-Rhein-Sieg
Grantham-Allee 20
53757 St. Augustin
Fabian.Schwab@h-brs.de
Alexander.Findeisen@smail.inf.h-brs.de
Peter.Sakal@h-brs.de
Hartmut.Pohl@h-brs.de

Abstract: Threat Modeling ermöglicht als heuristisches Verfahren die methodische Überprüfung eines Systementwurfs oder einer Softwarearchitektur, um Sicherheitslücken kostengünstig und frühzeitig - idealerweise in der Design Phase - im Software-Entwicklungsprozess zu identifizieren, einzugrenzen und zu beheben. Threat Modeling lässt sich aber auch noch erfolgreich in der Verifikationsphase oder noch später - nach dem Release - zur Auditierung der Software einsetzen.

Durch die Früherkennung von Sicherheitslücken können die Kosten zur Behebung bis auf ein Hundertstel reduziert werden. Die auf dem Markt verfügbaren Threat Modeling Tools werden identifiziert, analysiert und hinsichtlich Ihrer Eignung zur Erstellung komplexer, vollständiger Threat Models mit entwickelten Bewertungsparametern einem einfachen Bewertungsverfahren unterworfen.[1]

[1] Förderung des Projektes SoftSCheck durch das Bundesministerium für Bildung und Forschung (Förderkennzeichen 01 IS 09030)

1 Einführung Threat Modeling

Das heuristische Verfahren Threat Modeling unterstützt die Identifizierung von Sicherheitslücken. Durch Bewertung und Kategorisierung von Sicherheitslücken können Schutzmechanismen und Gegenmaßnahmen bestimmt werden, um ein Sicherheitsrisiko zu mindern, zu minimieren, zu beheben oder auch zu akzeptieren [Howard 2006]. Allerdings können grundsätzlich nicht alle Sicherheitslücken identifiziert werden [Turing 1936, Dijkstra 1972].

Die Erstellung eines Threat Models für einen Systementwurf erfolgt idealerweise bereits in der Design Phase, da dort die Kosten zur Beseitigung von Sicherheitslücken minimal sind. Werden Sicherheitslücken dagegen erst nach dem Release der Software an die Kunden entdeckt, so müssen Patches erstellt und ausgeliefert werden – ein deutlicher Mehraufwand gegenüber der Entdeckung in der Design Phase. Nach vollständiger Identifizierung schützenswerter Komponenten (Assets) sowie zugehöriger Bedrohungen und Sicherheitslücken ist ihre Bewertung erforderlich. Identifizierung und Bewertung der Bedrohungen und Sicherheitslücken kann z.B. durch Attack Trees erfolgen [Schneier 1999]. Auf Grundlage dieser Bewertung kann eine Minderung (Mitigation) der Bedrohungen und eine Behebung der Sicherheitslücken erfolgen. Neben den unterschiedlichen, in den Threat Modeling Tools implementierten, Maßnahmen (z.B. Redesign, Standard Mitigation, Custom Mitigation oder Accept Risk) ist eine individuelle Behandlung einzelner Bedrohungen und Sicherheitslücken sowie die Kontrolle der implementierten Verfahren erforderlich. Die Entstehung neuer Bedrohungen für ein bereits durch Threat Modeling spezifiziertes System ist nach [OWASP 2009] unwahrscheinlich. Für den systematischen Ablauf der Verfahrens Threat Modeling vgl. Abbildung 1: Threat Modeling Ablauf [nach: Swiderski 2004].

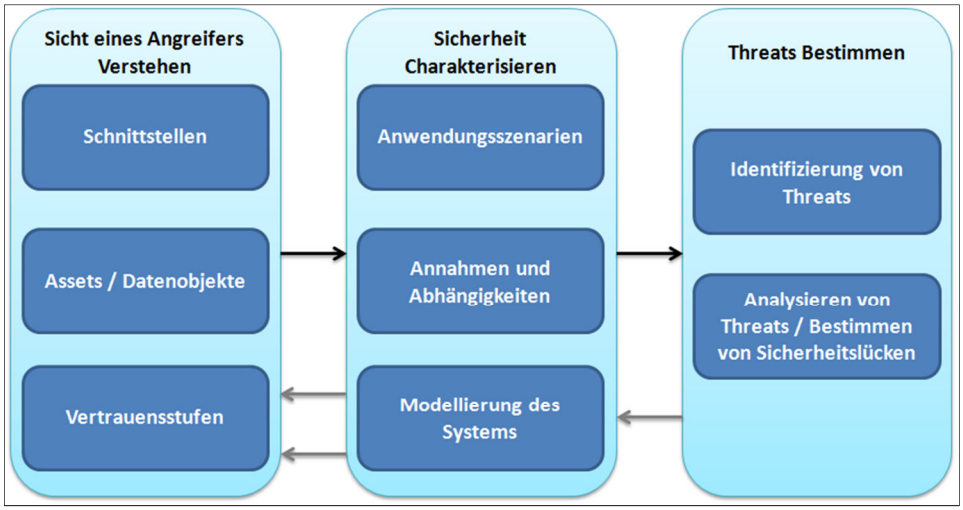

Abbildung 1: Threat Modeling Ablauf [nach: Swiderski 2004]

Durch den Einsatz von Threat Modeling bereits in der Design Phase ist eine vertrauenswürdige Implementierung von Software bereits vor erstmaliger Veröffentlichung (Release) möglich. Die Kosten zur Behebung einer Sicherheitslücke können durch die Forcierung einer proaktiven Früherkennung von Sicherheitslücken auf ein Hundertstel reduziert werden (vgl. Abbildung 2: Kosten zur Behebung von Sicherheitslücken im Verlauf der Softwareentwicklung [nach: NIST 2002]). Threat Modeling ist ein kosteneffektives und vollständiges Verfahren zur Identifizierung, Vermeidung und Verminderung von Sicherheitslücken und Bedrohungen [Myagmar 2005a, Myagmar 2005b]. Durch Threat Modeling gewonnene Erkenntnisse können ebenso zur Entwicklung strategischer Penetration Tests genutzt werden. Hierbei wird auf Erkenntnisse über Angriffspunkte und Eingabeschnittstellen zurückgegriffen, welche aus der modellierten Systemarchitektur gewonnenen werden können.

Neben der Anwendung in der Design Phase von Software ist es auch möglich ein Threat Model für eine bereits implementierte Systemarchitektur zu erstellen. Hierbei wurden in den durchgeführten Untersuchungen große Erfolge erzielt. Durch herkömmliche Verfahren nicht identifizierte Bedrohungen und Sicherheitslücken konnten durch die Anwendung von Threat Modeling, am Beispiel einer Individualsoftware, identifiziert und behoben werden.

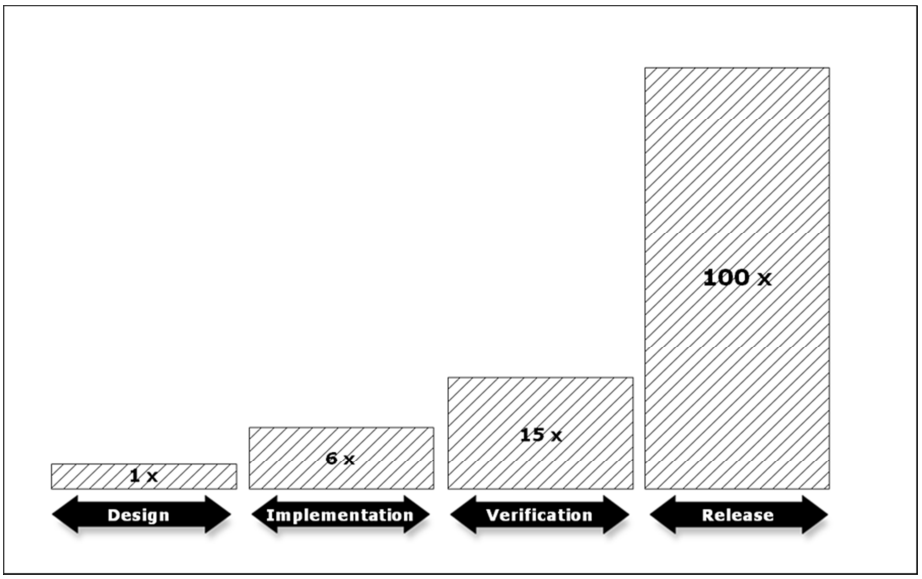

Abbildung 2: Kosten zur Behebung von Sicherheitslücken im Verlauf der Softwareentwicklung [nach: NIST 2002]

Die auf dem Markt verfügbaren Threat Modeling Tools werden identifiziert, analysiert und hinsichtlich Ihrer Eignung zur Erstellung komplexer, vollständiger Threat Models bewertet.

2 Bewertungsverfahren für Threat Modeling Tools

Um eine einheitliche Klassifizierung der zu untersuchenden Threat Modeling Tools zu gewährleisten, ist die Entwicklung und Begründung von passenden Bewertungsparametern unabdingbar. Diese müssen unter Berücksichtigung der im Projekt festgelegten Ziele neben einer eindeutigen Produktzuordnung die anfallenden Kosten, den Funktionsumfang, die Entwicklungsmöglichkeiten, den Installationsaufwand, die Benutzerfreundlichkeit und die Qualität der Dokumentation berücksichtigen. Die Bewertungsparameter sind in unterschiedliche Kategorien unterteilt. Kategorien beinhalten jeweils Parameter einer Funktionsgruppe. Parameter sind entsprechend ihrer Relevanz gewichtet und fließen in die Berechnung der jeweiligen Kategorie ein. Der Wert eines numerischen Parameters wird anhand der folgenden Formel berechnet:

$$\text{Wert des Parameters} = \frac{\sum \text{Punkte}}{\text{Mögliche Punkte}} \times \text{Parametergewichtung}$$

Die möglichen Punkte werden anhand der Bewertungsmethode bestimmt. Sofern eine Existenzprüfung (E) vorgenommen wird, wird lediglich zwischen Existenz (1 Punkt) und Nicht-Existenz (0 Punkte) unterschieden. Die maximale Anzahl der möglichen Punkte beträgt somit 1. Im Falle einer qualitativen Bewertung (Q) des Parameters beträgt die maximale Anzahl der möglichen Punkte 3. Der Bewertungsmaßstab ist für die einzelnen Parameter individuell definiert.

Einzelne Kategorien werden ebenso auf der Grundlage Ihrer Relevanz unterschiedlich gewichtet. Die Berechnung erfolgt anhand der folgenden Formel:

$$\text{Bewertung der Kategorie} = \frac{\sum \text{Werte der Parameter}}{\sum \text{Gewichtungen d.Parameter}} \times \text{Kategoriengewichtung}$$

Das Endergebnis des Bewertungsparameterkatalogs erfolgt anhand der folgenden Formel und liefert einen für Vergleiche relevanten Wert:

$$\text{Gesamtbewertung} = \frac{\sum \text{Bewertung der Kategorien}}{\sum \text{Kategoriengewichtungen}} \times 100$$

Die Gewichtungen der Kategorien und Parameter sowie die verwendete Bewertungsmethode können Abbildung 3 entnommen werden.

Die Produkte werden anhand der folgenden Kategorien bewertet:

- Erfassung der Modellierungsmöglichkeiten hinsichtlich Assets, Threats, Threat Mitigation, Vulnerabilities und dem eingesetzten System bzw. der Systemumgebung.
- Folgende Visualisierungsmöglichkeiten werden erfasst: Datenflussdiagramm, Bedrohungsbaum, Anwendungsdiagramm sowie sonstige Möglichkeiten.
- Berichtswesen (Reporting) erfasst Möglichkeiten des grafischen und textuellen Exports sowie den Reportumfang.

- Folgende Entwicklungsmöglichkeiten werden berücksichtigt:
  - Vorhandener Quellcode
  - Entwicklungsschnittstellen: Möglichkeit zur Anbindung eigener Erweiterungen und/oder zusätzlicher Module)
  - Communityprojekt: Entwicklung und/oder Support durch eine Community
  - Dokumentation der Entwicklungstools
- Beim Installations- und Nutzungsaufwand wird unterschieden nach der Grundinstallation, speziellen Voraussetzungen (z.B. erforderliche Software), Einarbeitungszeit, Usability und Anwendung (vollständige Modellierung eines Testszenarios).

Für eine vollständige Übersicht der Bewertungsparameter inkl. Gewichtung vgl. Abbildung 3: Gewichtung der Bewertungsparameter.

Nr.	**Bewertungsparameter**	**Prüfung**	**Gewichtung**	**%**
1	**Modellierungsmöglichkeiten**		**1**	**34%**
1.1	Assets	Q	1	29%
1.2	Threats	Q	1	29%
1.3	Threat Mitigation	E	0,5	14%
1.4	Vulnerabilities	Q	0,5	14%
1.5	System/Umgebung	Q	0,5	14%
2	**Visualisierung**		**0,3**	**10%**
2.1	Datenflussdiagramm	E	0,2	18%
2.2	Bedrohungsbaum	E	0,2	18%
2.3	Anwendungsdiagramm	E	0,2	18%
2.4	Sonstiges	Q	0,5	45%
3	**Reporting**		**0,8**	**28%**
3.1	Grafischer Export	Q	0,5	20%
3.2	Textueller Export	Q	1	40%

3.3	Umfang Report	Q	1	40%
4	**Entwicklungsmöglichkeiten**		**0,3**	**10%**
4.1	Quellcode vorhanden	E	0,8	36%
4.2	Entwicklungsschnittstellen	E	0,8	36%
4.3	Communityprojekt	E	0,3	14%
4.4	Dokumentation der Entw.-Tools	E	0,3	14%
5	**Installations-/Nutzungsaufwand**		**0,5**	**17%**
5.1	Grundinstallation	Q	0,8	20%
5.2	Spezielle Voraussetzungen	Q	0,8	20%
5.3	Einarbeitungszeit	Q	0,8	20%
5.4	Usability	Q	0,8	20%
5.5	Anwendung	Q	0,8	20%
Gesamt			**2,9**	**100%**

Abbildung 3: Gewichtung der Bewertungsparameter

3 Marktanalyse und Bewertung

Zum Zeitpunkt dieser Untersuchung waren sechs Software-Produkte (entgeltfrei oder entgeltpflichtig) zur Erstellung von Threat Models verfügbar. Die Produkte können durch folgende Kategorien beschrieben werden:

- Universal: Das Produkt eignet sich zur Erstellung eines Threat Models ohne spezifische Ausrichtung.
- Netzwerk: Das Produkt eignet sich zur Erstellung eines Threat Models für ein Netzwerkszenario. Es werden insbesondere Elemente eines Netzwerks betrachtet.
- Software Design: Das Produkt eignet sich zur Erstellung eines Threat Models im Rahmen des Software Designs und berücksichtigt spezifische Elemente der Software Entwicklung. Die Integration in den Software-Entwicklungsprozess von Microsoft - dem Security Development Lifecycle (SDL) - wird unterstützt.

Zu den untersuchten Produkten sowie deren Einsatzkategorie vgl. Abbildung 4: Erhältliche Threat Modeling ToolsAbbildung 4: Erhältliche Threat Modeling Tools

Name	Hersteller	Kategorie	Version
The CORAS Method	CORAS	Universal	20060714
Microsoft SDL Threat Modeling Tool	Microsoft	Software Design	3.1.3.1
Microsoft Threat Analysis & Modeling	Microsoft	Software Design	3.0
Practical Threat Analysis	PTA Technologies	Universal	1.6 Build 1212
Skybox Secure	Skybox Security	Netzwerk	4.5
Trike	Dymaxion	Universal	1.12a

Abbildung 4: Erhältliche Threat Modeling Tools

Um die Interessen von Herstellern zu wahren werden die Ergebnisse im weiteren Verlauf anonymisiert dargestellt. Im Rahmen der durchgeführten Untersuchungen wurden große Diskrepanzen beim Funktionsumfang - explizit bei Modellierungs- und Visualisierungsmöglichkeiten - festgestellt. Fünf der untersuchten sechs Produkte bieten Funktionen zur Modellierung elementarer Bestandteile eines Threat Models - Assets, Threats und Vulnerabilities [Schumacher 2006] - in unterschiedlichem Umfang. Mit dem weiteren Produkt (Tool C) ist ausschließlich die Modellierung von Assets möglich - bei dem Produkt handelt es sich um eine Sammlung von Symbolen zur Darstellung eines Threat Models. Lediglich zwei Produkte (Tool A und Tool E) bieten umfangreiche Visualisierungsmöglichkeiten in Form von Datenflussdiagrammen, Bedrohungsbäumen und Anwendungsdiagrammen. Die anderen Produkte bieten ausschließlich eigene Visualisierungsformen in unterschiedlichem Umfang. In den weiteren Kategorien werden Stärken und Schwächen der Produkte deutlich - extreme Diskrepanzen existieren jedoch nicht. Die Bewertung erfolgte auf Grundlage des entwickelten Bewertungsverfahrens. Die Ergebnisse der einzelnen Parameter, Kategorien und der Gesamtbewertung werden in Prozent der möglichen Punkte dargestellt. Lediglich drei Produkte erzielten in der Gesamtbewertung mehr als 70% der möglichen Punkte (Tool A, Tool B und Tool E).

Die weiteren Produkte weisen große Defizite (weniger als 10% der möglichen Punkte) in mindestens einer Kategorie auf und erzielen in der Gesamtbewertung weniger als 70% der möglichen Punkte. Die Erstellung komplexer, vollständiger Threat Models ist mit drei Produkten möglich. Für eine detaillierte Darstellung der Ergebnisse vgl. Abbildung 5: Tabellarische Darstellung Ergebnisse des Bewertungsverfahren

Nr	Bewertungs-parameter	Tool A	Tool B	Tool C	Tool D	Tool E	Tool F
1	**Modellierungs-möglichkeiten**	++++	++++	+	+++	++++	+++
1.1	Assets	++	+++++	++	+++	+++++	++
1.2	Threats	+++++	+++++	-	+++	+++	+++
1.3	Threat Mitigation	+++++	+++++	-	+++++	+++++	+++++
1.4	Vulnerabilities	+++++	+++++	-	++	+++	++
1.5	System/Umgebung	+++++	+++++	-	++	+++++	++
2	**Visualisierung**	+++++	+	-	+	+++++	+
2.1	Datenflussdia-gramm	+++++	-	-	-	+++++	-
2.2	Bedrohungsbaum	+++++	-	-	-	+++++	-
2.3	Anwendungs-diagramm	+++++	-	-	-	+++++	-
2.4	Sonstiges	+++++	++	-	++	+++++	++
3	**Reporting**	++++	++++	+	++	++++	+
3.1	Graphischer Export	++	++	-	++	++	+++
3.2	Textueller Export	+++++	+++++	++	+++	+++	-
3.3	Umfang Report	+++++	+++++	++	++	+++++	++
4	**Entwicklungs-möglichkeiten**	+	+	-	-	+++	-
4.1	Quellcode vorhan-den	-	-	-	-	-	-
4.2	Entwicklungs-schnittstellen	-	-	-	-	+++++	-
4.3	Communityprojekt	+++++	-	-	-	+++++	-

4.4	Dokumentation der Entw.-Tools	-	+++++	-	-	-	-
5	**Installations-/Nutzungsaufwand**	++++	++++	+++	+++++	++	++++
5.1	Grundinstallation	++	+++++	+++++	+++++	-	+++++
5.2	Spezielle Voraussetzungen	+++	+++++	+++++	+++++	++	+++++
5.3	Einarbeitungszeit	+++++	-	+++++	+++++	+++	++
5.4	Usability	+++++	+++++	-	+++++	+++	++
5.5	Anwendung	+++++	+++++	++	+++++	+++	++
Gesamt		++++	++++	+	+++	++++	++

Abbildung 5: Tabellarische Darstellung Ergebnisse des Bewertungsverfahrens

Die Ergebnisse in Abbildung 5 werden prozentual ausgedrückt, wobei +++++ dem Maximalwert von 100% entspricht. Die durch die Produkte erzielten Punkte in den einzelnen Kategorien sind in Abbildung 6 grafisch dargestellt. Die Stärken und Schwächen sowie die Gesamtbewertung und der Funktionsumfang der einzelnen Produkte werden verdeutlicht.

Tool A zeichnet sich durch gute Modellierungsmöglichkeiten sowie durch eine sehr gute Visualisierung aus; das Reporting ist gut ausgebildet; der Installations- und Nutzungsaufwand ist gut und Entwicklungsmöglichkeiten sind befriedigend - es handelt sich um ein Communityprojekt.

Tool B zeichnet sich durch gute Modellierungsmöglichkeiten und ein gutes Reporting aus; Entwicklungsmöglichkeiten sind in geringen Umfang vorhanden und gut dokumentiert; die Visualisierung ist ausreichend (nur eigene Visualisierungsformen) und der Installations- und Nutzungsaufwand ist gut.

Tool C bietet unzureichende Modellierungsmöglichkeiten und Berichtsfunktionen; Funktionen zur Visualisierung und Entwicklungsmöglichkeiten sind nicht vorhanden; der Installations- und Nutzungsaufwand ist befriedigend; es handelt sich um eine Sammlung von Symbolen zur Darstellung eine Threat Models.

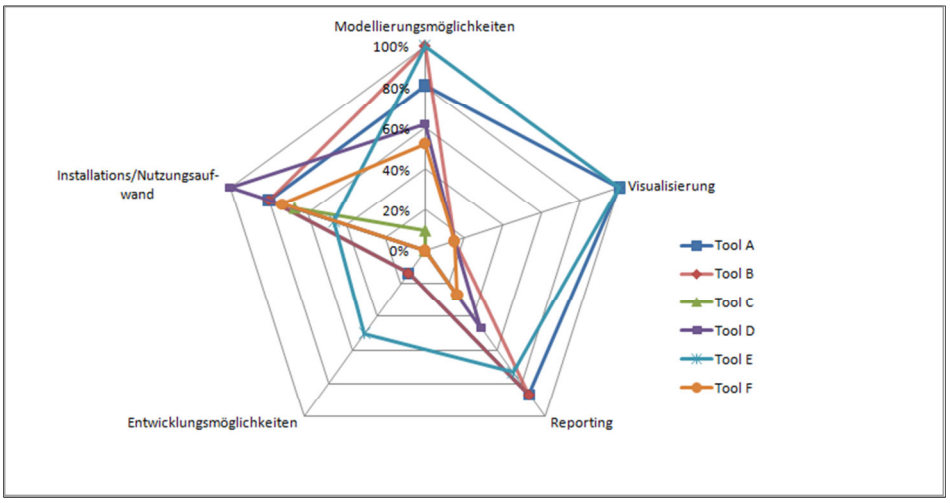

Abbildung 6: Ergebnisse: Grafische Ergebnisdarstellung des Bewertungsverfahrens

Tool D bietet befriedigende Modellierungsmöglichkeiten und ein ausreichendes Reporting; Visualisierung ist mangelhaft und Entwicklungsmöglichkeiten sind nicht vorhanden; der Installations- und Nutzungsaufwand ist sehr gering.

Tool E zeichnet sich durch gute Modellierungsmöglichkeiten und eine sehr gute Visualisierung aus; das Reporting ist gut ausgebildet; die Entwicklungsmöglichkeiten sind gut - es handelt sich um ein Communityprojekt und es sind Entwicklungsschnittstellen vorhanden; der Installations- und Nutzungsaufwand ist ausreichend gering.

Tool F bietet befriedigende Modellierungsmöglichkeiten und einen guten Installations- und Nutzungsaufwand; Visualisierung und Reporting sind mangelhaft; Entwicklungsschnittstellen sind nicht vorhanden.

Zusammenfassend ergibt sich eine Spitzengruppe gebildet aus den Tools A, B und E. Die Tools C, D und F zeigen - bei Anwendung der hier zugrunde gelegten Bewertungsparameter - nur mittlere oder sogar unzureichende Leistungen - vgl. Abb. 7: Vergleich der untersuchten Produkte.

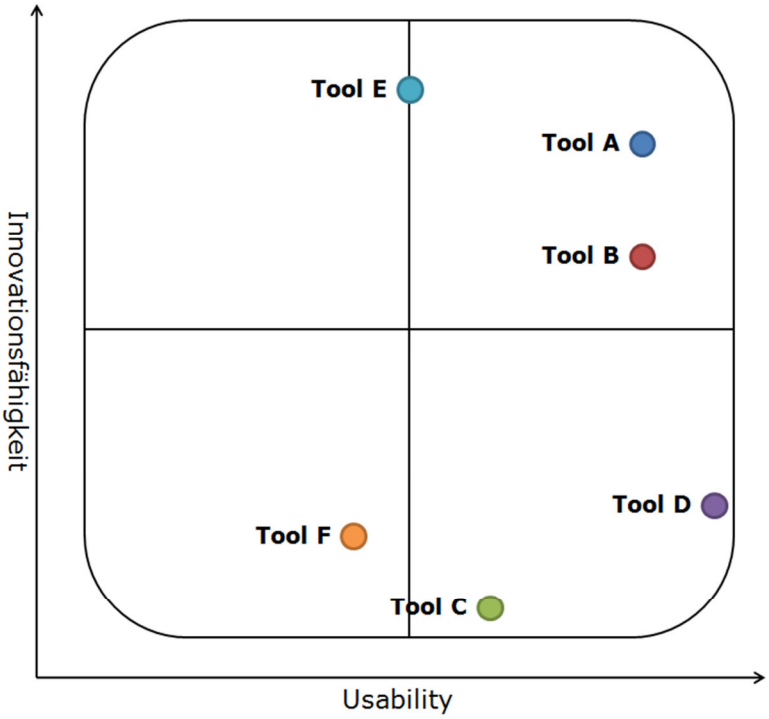

Abbildung 7: Vergleich der untersuchten Produkte

4 Fazit

Die Erstellung eines Threat Models kann mit entsprechenden Tools unterstützt werden. Die erhältlichen Produkte wurden anhand eines entwickelten Verfahrens bewertet. Lediglich drei der untersuchten Produkte erzielten zufriedenstellende Ergebnisse und können zur Erstellung komplexer, vollständiger Threat Models eingesetzt werden. Alle Tools wurden auf Basis des durchgeführten Bewertungsverfahrens zusammenfassend im Hinblick auf ihren Innovationsgrad bewertet – vgl. Abbildung 7: Vergleich der untersuchten Produkte

Das best-bewertetste Tool wurde bereits mehr als 20 mal zur Identifizierung von jeweils etwa 50 aus dem Internet ausnutzbaren Sicherheitslücken in Standard- und Individualsoftware erfolgreich eingesetzt.

Es ist daher zu erwarten, dass sich Threat Modeling zur Identifizierung von Sicherheitslücken durchsetzt - neben Fuzzing, mit dem nach der Implementierung weitere Softwarefehler und Sicherheitslücken erfolgreich identifiziert werden können.

Literaturverzeichnis

[Dikstra 1972] Dijkstra, E. W.: Chapter I: Notes on structured programming Structured programming" (pp. 1-82). Eindhofen 1972

[Howard 2006] Howard, M.; Lipner, S.: The Security Development Lifecycle. SDL: A Process for Developing Demonstrably More Secure Software. Microsoft Press, Redmond 2006.

[Myagmar 2005a] Myagmar S.: Threat Modeling networked and data-centric systems, Urbana 2005, http://www.projects.ncassr.org/threatmodeling/myagmar-msthesis.pdf

[Myagmar 2005b] Myagmar, S.; Lee, A. J..; Yurcik, W.: Threat Modeling as a Basis for Security Requirements. Symposium on Requirements Engineering for Information Security. Paris 2005. http://www.suvda.com/papers/threat_sreis05.pdf.

[NIST 2002] National Institute of Standards and Technology (NIST): The Economic Impacts of Inadequate Infrastructure for Software Testing. Gaithersburg 2002. http://www.nist.gov/director/prog-ofc/report02-3.pdf

[OWASP 2009] OWASP Foundation: Threat Risk Modeling. Columbia 2009. http://www.owasp.org/index.php/Threat_Risk_Modeling

[Schumacher 2006] Schumacher, M. et. al.: Security Patterns. Integrating Security and Systems Engineering. Wiley, Hoboken 2006

[Turing 1936] Turing, A.: On computable numbers, with an application to the "Entscheidungsproblem", Proceedings of the London Mathematical Society, Series 2, 42 pp 230–265, London 1936

[Schneier 1999] Schneier, B.: Attack Trees: Modeling Security Threats. In: Dr. Dobb's Journal, v. 24, n.12. 1999. http://www.schneier.com/paper-attacktrees-ddj-ft.html

[Swiderski 2004] Swiderski, F.; Snyder, W.: Threat Modeling. Microsoft Press, Redmond 2004.