# Beschreibung sicherheitsrelevanten Verhaltens in der Open Source Softwareentwicklung

Masterarbeit an der Freien Universität Berlin
für den Abschluss zum Master of Computer Science

Tobias Opel

18. August 2008

Fachbereich Mathematik und Informatik, Institut für Informatik,
Arbeitsgruppe Software Engineering
Verantwortlicher Hochschullehrer: Prof. Dr. Lutz Prechelt
Betreuer: Dipl.-Medieninf. Martin Gruhn

# Erklärung

Hiermit erkläre ich, dass ich diese Arbeit vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Berlin, den 18.08.2008

Tobias Opel

## Kurzfassung

Diskussionen zu den Fragen „Wie sicher ist Open Source?“ und „Ist Open Source Software sicherer als Closed Source?“ sind meist in Literatur zum Thema Open Source Software und Sicherheit zu finden.

Mit dieser Arbeit untersuche ich einen anderen Aspekt von Sicherheit und Open Source Software , nämlich sicherheitsrelevantes Verhalten in der Open Source Softwareentwicklung. Dabei steht im Mittelpunkt die Betrachtung von Verhaltensweisen und Rollen bei der Schwachstellenbehebung. Mit Hilfe der Forschungsmethode Grounded Theory habe ich eine qualitative Datenanalyse durchgeführt. Dabei habe ich Klassifikationen für die Verhaltensweisen und für die Rollen entwickelt, die bei der Behebung von Schwachstellen auftreten.

Bei der Erstellung dieser Klassifikation habe ich das Phänomen „Umgang mit Schwachstellenveröffentlichung“ entdeckt und dieses genauer mit Hilfe der Grounded Theory untersucht. Als Ergebnis dieser Analyse entstanden Aussagen über Muster, welche verschiedene Einflüsse auf den Behebungsverlauf widerspiegeln. Dabei ergaben sich unter anderem „Art und Ort der Veröffentlichung“, „Ausnutzungsbeschreibung von Schwachstellen“ und „Schweregrad einer Schwachstelle“ als Faktoren mit Einfluss auf den Behebungsverlauf.

# Inhaltsverzeichnis


**1 Einleitung** **1**
1.1 Ziel der Arbeit . . . 1
1.1.1 Kernaufgaben . . . 2
1.2 Begriffe . . . 2
1.3 Einordnung meiner Arbeit innerhalb der Informatik . . . 7
1.4 Vorgehen . . . 12

**2 Methodik und Praktiken der Analyse** **14**
2.1 Datenerhebung . . . 14
2.1.1 Grundlagen und Projektbestimmung . . . 14
2.1.2 Datenerhebung innerhalb der Projekte . . . 16
2.1.3 Eigenschaften der Datenbasis . . . 17
2.1.4 Projekte zu Beginn der Analyse . . . 18
2.2 Forschungsmethode Grounded Theory . . . 22
2.2.1 Offenes Kodieren . . . 23
2.2.2 Axiales Kodieren . . . 24
2.2.3 Selektives Kodieren . . . 25
2.2.4 Werkzeug zur Unterstützung der Datenanalyse atlas.ti . . . 25
2.3 Angewandte Praktiken bei der Datenanalyse . . . 26
2.3.1 Blickwinkel . . . 27
2.3.2 Konzeptnamensschema . . . 28
2.3.3 Analyse-Modell . . . 30
2.4 Zusammenfassung . . . 32

**3 Analyse und Ergebnisse** **33**
3.1 Klassifikation der Verhaltenskonzepte und -kategorien . . . 33
3.1.1 Die Konzeptebene . . . 34
3.1.2 Verlauf der Behebung . . . 35
3.1.3 Die Verhaltenskategorien . . . 36
3.1.4 Die Hauptkategorien . . . 39
3.2 Klassifikation der Rollen . . . 41
3.2.1 Rollenkategorien . . . 42
3.3 Phänomen „Umgang mit Schwachstellenveröffentlichung“ . . . 44

3.3.1 Begriff Schwachstellenveröffentlichung . . . . . 44
3.3.2 Festlegung des Untersuchungsgegenstands . . . . . 45
3.3.3 Ziel und Vorgehen bei der Phänomenuntersuchung . . . . . 47
3.3.4 Untersuchungsergebnisse . . . . . 55
3.3.5 Kernaussagen über das Phänomen . . . . . 66
3.4 Ausblick: Drängen nach Schwachstellenbehebung . . . . . 68
3.5 Vermeidung . . . . . 69
3.6 Zusammenfassung . . . . . 71

**4 Fazit und Ausblick** **72**
4.1 Herausforderungen in der Arbeit . . . . . 72
4.2 Fazit . . . . . 73
4.3 Ausblick und weiteres Vorgehen . . . . . 75

**Literaturverzeichnis** **77**

**A Anhang** **80**
A.1 Konzepte . . . . . 80
A.2 Konzept-/Kategorieklassifikation als hierarchische Liste . . . . . 96
A.3 Rollenklassifikation . . . . . 101
A.3.1 Rollenkonzepte . . . . . 101
A.3.2 Rollenkategorien . . . . . 102
A.4 Beschreibung der Episoden . . . . . 103
A.4.1 Wordpress . . . . . 104
A.4.2 Joomla . . . . . 107
A.4.3 TWiki . . . . . 112
A.5 Beschreibung des CD Inhalts . . . . . 114
A.5.1 Meine Atlas.ti Hermeneutic Units . . . . . 114
A.5.2 Tabellen und Diagramme des Axialen Kodierens . . . . . 115

# Abbildungsverzeichnis

1.1 Beziehung zwischen Schwachstelle, Sicherheitsbedrohung und Sicherheitsrisiko nach Eckert [5] . . . . . 4
1.2 Einordnung meiner Untersuchung in den wissenschaftlichen Kontext . . . 12

2.1 Dokument-Texteintrag-Autor Beziehung . . . . . 18
2.2 Paradigmatisches Modell . . . . . 24
2.3 Analyse Meta-Modell . . . . . 31

3.1 Behebungsverlauf . . . . . 35
3.2 Veränderung der Rollenzuordnung über Zeit . . . . . 43
3.3 Dauer von Veröffentlichung bis zur Behebung der einzelnen Episoden . . . 56

# Tabellenverzeichnis

3.1 Eigenschaften des Phänomens . . . . . 51
3.2 Eigenschaften der Konsequenz Schwachstellenbehebung . . . . . 54
3.3 Beziehung von Dauer und optionalen Eigenschaften der ursächlichen Bedingung . . . . . 58
3.4 Beziehung der Ausnutzungsangabe und der Behebungsdauer . . . . . 58
3.5 Beziehung zwischen Behebungsdauer und niedrig bzw. moderat eingestufter Schwachstellen . . . . . 60
3.6 Schutzmaßnahmen durch den externen Helfer in den verschiedenen Episoden 65

# 1 Einleitung

Die Veröffentlichung der Schwachstelle ohne Einhaltung des dafür vorgesehenen Vorgehens, die Angabe der Schwachstellenausnutzung bei einer Schwachstellenveröffentlichung, aber auch die Bereitstellung einer Schwachstellenmeldemöglichkeit durch den Hersteller können zu einer schnellen Behebung von sicherheitsrelevanten Schwachstellen in Open Source Software führen.

Das ist eine Erkenntnis aus meiner Beschreibung von sicherheitsrelevantem Verhalten in der Open Source Softwareentwicklung. Diese Arbeit stellt die Ziele, das Vorgehen und die Ergebnisse meiner Untersuchung, welche die Grundlage für meine Beschreibung von sicherheitsrelevantem Verhalten in der Open Source Softwareentwicklung bildete, vor. Dabei habe ich konkret die Behebung von Schwachstellen untersucht.

Im ersten Abschnitt dieses Kapitels stelle ich die Zeile und konkreten Aufgaben meiner Arbeit vor. Danach beschreibe ich für meine Arbeit relevante Begriffe. Im dritten Abschnitt dieses Kapitels ordne ich meine Arbeit innerhalb der Informatik ein. Im letzten Abschnitt beschreibe ich das Vorgehen meiner Arbeit.

## 1.1 Ziel der Arbeit

Das Ziel meiner Arbeit ist mit Hilfe der Grounded Theory eine qualitative Datenanalyse auf existierenden Datenbeständen (z.B. in Patch Trackern, Advisory Boards und Mailinglisten) durchzuführen, um dabei Verhaltensweisen und Rollen, welche bei der Behebung von sicherheitsrelevanten Schwachstellen in der Open Source Softwareentwicklung involviert sind, zu identifizieren und zu beschreiben. Dabei erstelle ich zuerst eine Klassifikation für Verhaltensweisen und eine für Rollen. Die bei der Erstellung der Verhaltensklassifikation entdeckten Phänomene untersuche ich im Anschluss, mit dem Ziel, Bedingungen, Beziehungen und weitere Einflussfaktoren zu erkennen und deren Wechselwirkung mit den untersuchten Phänomenen zu verstehen und zu beschreiben.

Die Analyse beschränke ich dabei auf Open Source Webanwendungen. Außerdem beschreibe ich leicht erkennbare Verhaltensweisen und Praktiken zur Vermeidung von sicherheitsrelevanter Schwachstellen , die ich bei der Analyse der Schwachstellenbehebung in den ausgewählten Projekten beobachte.

### 1.1.1 Kernaufgaben

Aus dem oben genannten Ziel habe ich folgende fünf Kernaufgaben abgeleitet, welche ich innerhalb dieser Arbeit bearbeite:

1. Datenauswahl - Auswahl existierender Datenbestände und Auffinden geeigneter Beschreibungen von Verhalten und Rollen, welche bei der Behebung sicherheitsrelevanter Schwachstellen auftreten. Ich stelle das Vorgehen der Datenauswahl im Kapitel 2 vor.
2. Datenanalyse - Ableitung von Verhaltens- und Rollenkategorien und Untersuchung der Beziehungen zwischen den Kategorien, welche beim Phänomen „Umgang mit Schwachstellenveröffentlichung“ auftreten, um Aussagen zu erhalten, welche das sicherheitsrelevante Verhalten in der Open Source Softwareentwicklung, beschreiben. Im Kapitel 2 beschriebe ich die Methodik und angewandte Praktiken meiner Datenanalyse. Das Vorgehen und die Ergebnisse der Analyse stelle ich im Kapitel 3 vor. Diese Aufgabe stellt den Schwerpunkt der Arbeit dar.
3. Literaturarbeit - Darstellung und Vergleich von Literatur zu Qualitätssicherung, Sicherheit, Behebung und Vermeidung von sicherheitsrelevanten Schwachstellen, sowie zur Schwachstellenveröffentlichung mit den eigenen Erkenntnissen. Mit Literatur zu den ersten drei Themen setze ich mich in den folgenden beiden Abschnitten dieses Kapitels auseinander. Im Rahmen der Untersuchung des Phänomens „Umgang mit Schwachstellenveröffentlichung“ gehe ich im Kapitel 3 auf Literatur dazu ein.
4. Behebungsdauer und Erkenntniszusammenfassung - Untersuchung der Einflüsse auf die Behebungsdauer durch die Betrachtung der Beziehungen zwischen den Kategorien, welche beim Phänomen „Umgang mit Schwachstellenveröffentlichung“ auftreten. Sowie die Zusammenfassung der Erkenntnisse aus dieser Untersuchung. Diese Ergebnisse stelle ich im Kapitel 3 vor.
5. Schwachstellenvermeidung - Diese Aufgabe liegt nicht im Fokus meiner Arbeit. Bei dieser Aufgaben beschreibe ich lediglich leicht erkennbare Verhaltensweisen und Praktiken zur Vermeidung sicherheitsrelevanter Schwachstellen in ausgewählten Projekten. Die Beschreibung der Vermeidung stelle ich im Kapitel 3 vor.

## 1.2 Begriffe

Die International Standards Organization kurz ISO [8] definiert den Begriff Sicherheit als „die Minimierung der Verwundbarkeit von Werten und Ressourcen“. Eckert [5] erweitert diese Definition und unterteilt den Sicherheitsbegriff in Informationssicherheit (engl. security), Datensicherheit (engl. protection) und Funktionssicherheit (engl. safety):

- Die Informationssicherheit ist gewährleistet, wenn es zu keinen unautorisierten Informationsveränderungen oder -gewinnen kommt.
- Datensicherheit beschreibt die Eigenschaft eines Systems keine Systemzustände anzunehmen, die zu einem unautorisierten Zugriff auf Systemressourcen, wie z.B. Daten, führen könnten. Darüber hinaus fallen auch Maßnahmen zur Datensicherung unter den Begriff der Datensicherheit.
- Die Möglichkeit ausschließlich spezifizierte Funktionalitäten eines Systems auszuführen bedeutet Funktionssicherheit. Sie bildet die Grundlage für die anderen beiden Sicherheitsbegriffe.

Eckert [5] definiert im Zusammenhang mit Sicherheit noch weitere Begriffe, welche innerhalb dieser Arbeit häufiger auftauchen. Diese Begriffe sind Schwachstelle, Verwundbarkeit (engl. vulnerability), Bedrohung, Risiko und Angriff.

Bei einer Schwachstelle handelt es sich um eine Stelle im Quellcode der Software, welche diese verwundbar macht, d.h. es existiert ein Defekt in der Software durch den die Sicherheit der Software gefährdet ist. Eine Schwachstelle, über welche Sicherheitsmaßnahmen der Software „umgangen, getäuscht oder unautorisiert modifiziert werden können“ [5], ist eine Verwundbarkeit.

> Der Begriff sicherheitsrelevante Schwachstelle wird im Lauf der Arbeit zur Verwundbarkeit von mir als Synonym verwendet.

Bei einer Sicherheitsbedrohung handelt es sich um die mögliche Ausnutzung einer (sicherheitsrelevanten) Schwachstelle, um dadurch die Sicherheit der Software zu gefährden. Der Begriff des Sicherheitsrisikos beschreibt die Wahrscheinlichkeit eines potentiellen Eintritts von Schaden und dessen Höhe durch eine Sicherheitsbedrohung. Abbildung 1.1 auf der nächsten Seite zeigt den Zusammenhang zwischen Schwachstelle, Bedrohung und Risiko nach Eckert [5].

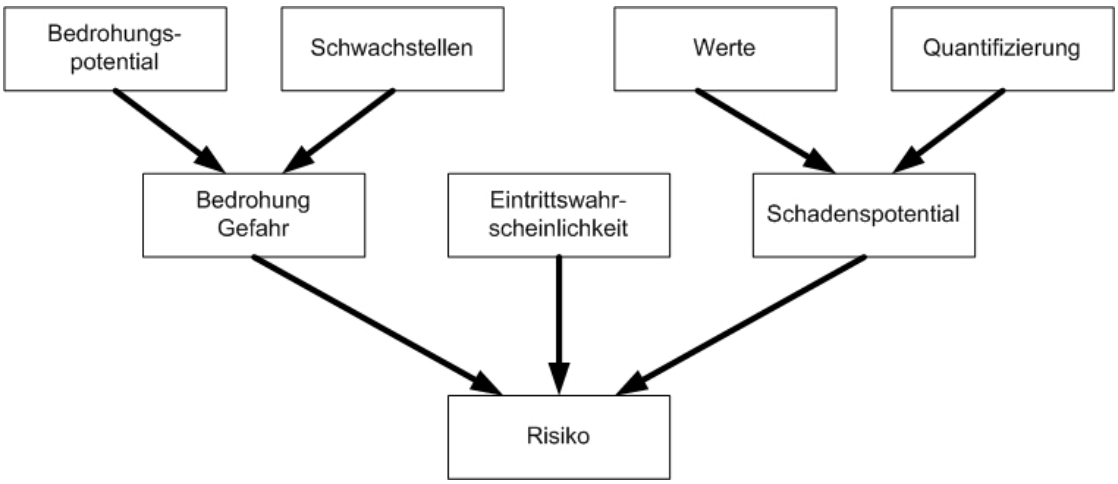


Abb. 1.1: Beziehung zwischen Schwachstelle, Sicherheitsbedrohung und Sicherheitsrisiko nach Eckert [5]

Ein Angriff stellt die Aktion eines Akteurs dar, welche die Sicherheit gefährdet. So verletzt z.B. ein Angriff, durch unautorisierten Zugriff bzw. Versuch eines Zugriffs, auf die Software, um an Informationen zu gelangen, die Informationssicherheit. Als Angreifer bezeichnet man den Akteur, welcher einen Angriff ausführt. Im Kontext meiner Arbeit – die Betrachtung der Behebung von sicherheitsrelevanten Schwachstellen – ist der Angreifer ein Akteur, welcher absichtlich eine sicherheitsrelevante Schwachstelle ausnutzt.

Im Zusammenhang mit der Schwachstelle, genauer mit der Schwachstellenbehebung, steht der Begriff des Patches. Ein Patch, was zu deutsch Flicken bedeutet, bezeichnet die nachträgliche Änderung an der Software, wodurch Defekte bzw. Schwachstellen korrigiert oder neue Funktionen nachträglich in die Software eingebaut werden.

Nach den Definitionen der Begriffe zur Sicherheit gehe ich nun genauer auf Arten von sicherheitsrelevanten Schwachstellen ein. Es gibt eine Vielzahl unterschiedlicher Arten von Schwachstellen. Tsipenyuk et al. [19] stellen dafür die Fortify Taxonomie[1] vor, welche sicherheitsrelevante Schwachstellen in sieben Quellcode bezogene und eine durch äußere Einflüsse bedingte Kategorien (kingdoms) einordnen. Diese Kategorien sind:

1. **Input Validation and Representation** umfasst Schwachstellen, welche durch fehlerhafte Eingabeprüfung und -darstellung entstehen, sowie dadurch, dass in der Regel davon ausgegangen einer Eingabe vertrauen zu können.

[1] http://www.fortify.com/vulncat/ überprüft am 15.08.2008

2. **API Abuse** tritt auf wenn der Vertrag zwischen Aufrufendem und Bereitstellendem der Programmierschnittstelle nicht eingehalten wird. Diese Ausnutzung kann von beiden Seiten aus geschehen.
3. **Security Features** umfassen den unvollständigen oder unkorrekten Einsatz von Maßnahmen zur Sicherheitsgewährleistung.
4. **Time and State** gruppiert die Schwachstellen, welche beim unkorrekten Umgang mit Zuständen und Ressourcen, welche geteilt werden können, und der Zeit, die dafür benötigt wird, auftreten.
5. **Errors** beinhaltet Schwachstellen, welche durch den unvollständigen/unkorrekten Einsatz der Fehlerbehandlung, entstehen.
6. **Code Quality** in schlechter Form führt zu den Schwachstellen, welche in dieser Kategorie zusammengefasst werden.
7. **Encapsulation** ist Datenkapselung. Ein mangelhafter Einsatz dieses Konzepts führt zur Beeinträchtigung der Sicherheit.

*. **Environment** umfasst die Schwachstellen, die nicht durch einen Defekt im Quellcode entstehen sondern durch äußere Einflüsse ermöglicht werden. Dadurch wird sie als zusätzliche Kategorie angesehen.

Viele Angriffe zielen auf Schwachstellen von Anwendungen im Internet, den so genannten Webanwendungen. Eine Webanwendung (engl. web application) ist eine Software Anwendung, welche auf einem Server ausgeführt wird und auf die mit Hilfe eines Webbrowser über das Internet zugegriffen werden kann. Beispiele für Webanwendungen sind:

- Content Management Systeme (CMS), eine Anwendung zur Bereitstellung und Verarbeitung von Inhalten im Internet,
- Wikis, erlauben dem Nutzer neben der Bereitstellung von Inhalten auch die Möglichkeit der direkten Onlineänderung des Inhaltes,
- Weblogs oder kurz Blogs, dienen der Darstellung von Inhalten in Form eines Tagebuchs im WWW,
- Foren, stellt eine Plattform für Diskussion und Informationsaustausch bereit,
- E-Commerce Anwendungen bzw. Shopsysteme, über die materielle oder immaterielle Güter zum Verkauf angeboten werden.

Aufgrund der globalen Vernetzung, des einfachen Zugangs zum Internet und des stetig wachsenden Einsatzes von Webanwendungen nimmt die Zahl der Angriffe auf diese Art von Softwaresysteme zu.

Das Open Web Application Security Project (OWASP)[2] listet mit dem Projekt OWASP Top 10[3] die zehn häufigsten sicherheitsrelevanten Schwachstellen für Webanwendungen auf. Neben der Auflistung und Erklärung der einzelnen sicherheitsrelevanten Schwachstellen stellt das Projekt auch Möglichkeiten zum Schutz und Verweise auf weitere Informationen bereit. Unter den ersten fünf Plätzen der Top 10 befinden sich unter anderem Cross Site Scripting (XSS), Injection Flaws, Malicious File Execution und Cross Site Request Forgery (CSRF), auf welche ich im Rahmen meiner Untersuchung gestoßen bin.

Das Cross Site Scripting steht auf Platz eins der OWASP Top 10. Dabei nimmt die Anwendung eingegebene Daten von Benutzern entgegen, ohne diese zu prüfen. Diese Daten können schadhaften Code enthalten, der dann im Browser eines anderen Benutzer ausgeführt wird und Schaden anrichtet. Zum Beispiel kann dabei Code ausgeführt werden, welcher Benutzer- und Sitzungsdaten ausliest, um so dem Angreifer Zugriff auf die Anwendung unter den Zugangsdaten des Benutzers zu ermöglichen.

Ähnlich im Namen, aber eine andere Art der Schwachstelle beschreibt das Cross Site Request Forgery, welches auf dem fünften Platz der OWASP Top 10 zu finden ist. Dabei führt der Benutzer eine HTTP-Anfrage aus, die Aktionen beinhaltet, welche vom Nutzer nicht beabsichtigt sind und auf eine bestimmte Webanwendung zielen. Diese Anfrage kann je nach Status des Benutzers innerhalb der Webanwendung und Art der Manipulation verschiedene Folgen innerhalb der Anwendung haben. Beispielsweise kann ein neuer Nutzer angelegt werden oder ungewollte Eingaben an Formulare übergeben werden.

Über Injection Flaws Schwachstellen werden Daten oder Code in die Anwendung eingeschleust, was in einer unautorisierten Modifizierung von Daten oder einer unzulässigen Ausführung von nicht spezifizierten Funktionalitäten resultiert. Die bekannteste Form dieser Schwachstelle ist die SQL-Injection, bei welcher SQL Abfragen in das System eingeschleust und ausgeführt werden. Die Injection Flaws stehen auf Platz zwei der OWASP Top 10.

Platz drei in der OWASP Top 10 nimmt das Malicious File Execution ein. Dabei kann der Angreifer Schadcode in Dateien ausführen, welche sich nicht auf dem Server befinden. Diese Art der Schwachstellen verletzt die Funktionssicherheit.

Das Cross Site Scripting, das Malicious File Execution und die Injection Flaws sind in der Fortify Taxonomie in die Kategorie Input Validation and Representation eingeordnet, während CSRF in die Kategorie Encapsulation einzuordnen ist. Das zeigt, dass die Ursache der Schwachstellen bei Webanwendungen vor allem bei fehlender Eingabeprüfung und Absicherung in der Darstellung der Eingabe gegen schadhaften Code liegt.

[2] http://www.owasp.org/index.php/Main_Page überprüft am 14.08.2008
[3] http://www.owasp.org/index.php/Top_10_2007 überprüft am 14.08.2008

## 1.3 Einordnung meiner Arbeit innerhalb der Informatik

Der Begriff Sicherheit lässt sich in den Bereich Qualitätssicherung innerhalb der Softwaretechnik einordnen. Dies wird durch die begriffliche Nähe der Funktionssicherheit zu den Begriffen Verlässlichkeit, d.h. keine Annahme von unzulässigen Systemzuständen, und Zuverlässigkeit, damit ist Gewährleistung der Erfüllung der spezifizierten Funktion gemeint, deutlich.

Wang und Wang [21] gehen dabei noch einen Schritt weiter und zeigen auf, dass schon während der Entwurfsphase die Sicherheit der zu entwerfenden Software innerhalb der Softwarequalitätsüberlegungen betrachtet werden muss. Dabei adressieren sie Sicherheitsrisiken und -bedrohungen explizit zu Softwarequalitätsaspekten und zeigen Auswirkungen, welche durch die zugewiesenen Sicherheitsrisiken entstehen. Sie unterscheiden zwei Arten von Auswirkungen. Zum einen negative Auswirkung ohne irreversiblen angerichteten Schaden und zum anderen starke negative Auswirkung bis hin zu irreversiblen angerichteten Schaden.

Innerhalb der Qualitätssicherung betrachte ich mit meiner Arbeit die Gewährleistung und Verbesserung des Qualitätsaspekts Sicherheit. Dabei gibt es zwei Vorgehensweisen, welche bei der Sicherung dieses Qualitätsaspekt angewandt werden. Zum einen die Behebung und zum anderen die Vermeidung von sicherheitsrelevanten Schwachstellen.

#### Sicherheitsgewährleistung durch Schwachstellenbehebung

Bei der Behebung von Schwachstellen geht es darum, dass eine Schwachstelle entdeckt, veröffentlicht und behoben wird. Den typischen Ablauf dieser Behebung, in der weiteren Arbeit Behebungsverlauf oder auch Verlauf der Behebung genannt, stellen Arbaugh et al. [1] in einem Schwachstellenlebenszyklus dar, innerhalb dessen eine Schwachstelle die Zustände *Birth, Discovery, Disclosure, Correction, Publicity, Scripting* und *Dead* durchlaufen kann. *Discovery, Disclosure, Correction* entspricht den oben genannten Begriffe der Entdeckung, Veröffentlichung und Behebung. *Birth* stellt den Zeitpunkt während der Softwareentwicklung dar, an welchem der Defekt im Code entstanden ist. *Publicity* stellt die breitere Veröffentlichung der Schwachstelle dar. *Scripting* beschreibt den Vorgang der Automatisierung von Angriffen auf die Schwachstelle. Im Idealfall wird die Schwachstelle in jedem eingesetzten und betroffenen Softwaresystem behoben, so dass sie schließlich ausstirbt, was im Lebenszyklus den Zustand *Dead* bezeichnet.

Die Organization for Internet Safety [13] stellt den Verlauf der Schwachstellenbehebung aus einer anderen Perspektive dar. Sie stellen eine Anleitung zur Meldung einer Schwachstelle an den Hersteller der Software durch den Entdecker zur Verfügung. Darüber hinaus beinhaltet die Anleitung Schritte, welche den Verlauf ab der Meldung bis zur Behebung der Schwachstelle und die darauf folgenden Behebungsveröffentlichung durch den Hersteller

beschreiben. Diese Anleitung soll größtmöglichen Schutz während des beschriebenen Prozesses gewährleisten und mögliche negativen Seiteneffekte vermeiden,

Beide Darstellungen beschreiben den Behebungsverlauf, wobei im ersten Fall nicht genauer auf den Weg von der Veröffentlichung zur Behebung eingegangen wird. Diesen Verlauf betrachte ich im Rahmen meiner Untersuchung genauer. Im zweiten Fall wird der Prozess zwischen Entdeckung und Behebung durch konkrete Schritte genau vorgegeben, aber Gründe für diese Schritte werden nicht angeführt, so dass nicht klar wird, ob es eventuell bessere Möglichkeiten für den Behebungsverlauf gibt. Des weiteren wird die Verantwortung für die Behebung allein dem Hersteller übertragen. Das muss bei Open Source Software (siehe unten) nicht unbedingt der Fall sein. Auf den Einfluss unterschiedlicher Rollen auf diesen Aspekt des Behebungsverlaufs gehe ich ebenfalls in meiner Untersuchung näher ein.

### Sicherheitsgewährleistung durch Schwachstellenvermeidung

Verschiedene Ansätze und Praktiken dienen dazu Schwachstellen zu vermeiden. Praktiken sind einzelne Regeln und Vorgehensweisen um guten und schwachstellenfreien Code zu schreiben. Eine Auswahl dieser Praktiken, auf welche ich im Rahmen meiner Arbeit gestoßen bin, stelle ich im Abschnitt 3.5 vor. Ansätze zur Vermeidung von Schwachstellen sind Modelle, eigene Prozesse oder Prozessverbesserungen. Drei ausgewählte Ansätze zur Schwachstellenvermeidung stelle ich im Folgenden kurz vor:

Das Systems Security Engineering Capability Maturity Model (SSE-CMM)[4] ist ein Reifegradmodell, welches darauf ausgelegt ist, den Einsatz von Praktiken zur Gewährleistung der Sicherheit bei der Softwareentwicklung innerhalb von kommerziellen, akademischen oder Regierungs- Organisationen zu beurteilen und zu verbessern. Die Anwendung des Modells dient in erster Linie dazu, den Qualitätsaspekt Sicherheit innerhalb des Entwicklungsprozess zu verbessern. Die weitere Anwendungsmöglichkeit ist die Beurteilung des Reifegrads der Prozesse innerhalb der Organisation zum einen als Vergleichsmöglichkeit zu anderen Organisationen und zum anderen als Zertifizierung der Prozessqualität.

Der von Microsoft entwickelte Security Development Lifecycle (SDL)[5] beinhaltet Praktiken zur Gewährleistung und Verbesserung von Sicherheit, welche sich in die einzelnen Phasen des Softwareentwicklungsprozesses integrieren. Das SDL stellt somit eine Erweiterung zum klassischen Entwicklungsprozess dar. Lipner und Howard [12] beschreiben den erweiterten Entwicklungsprozess und besprechen Erfahrungen beim Einsatz der Praktiken innerhalb der Entwicklung von Microsoft Produkten. Praktiken zur Schwachstellenbehebung („Response Plan“ und „Response Execution“) ordnen sich in die Phasen Veröffentlichung und Wartung des SDL ein.

[4] http://www.sse-cmm.org/index.html überprüft am 14.08.2008
[5] http://msdn.microsoft.com/en-us/security/cc448177.aspx überprüft am 14.08.2008

Das Comprehensive, Lightweight Application Security Process (CLASP) Projekt[6] von OWASP ist der Ansatz setzt so früh wie möglich sicherheitsrelevante Aspekte in den Softwareentwicklungsprozesses ein. Dazu bietet das Projekt eine Menge von 24 Aktivitäten, welche zu bewährten Praktiken (engl. best practices) gruppiert werden, und Hilfsmitteln an, welche auf den eingesetzten Entwicklungsprozess angepasst werden können. Die best practice „Build vulnerability remediation procedures“ gruppiert die Aktivitäten „Address reported security issues“ und „Manage security issue disclosure process“, welche bei der Behebung von Schwachstellen angewandt werden können.

Diese drei Ansätze dienen in erster Linie dazu Aktivitäten im Prozess der Softwareentwicklung zu integrieren bzw. den Prozess generell anzupassen, um das durch den Prozess entstehende Produkt sicherer zu machen. Dies ist aber nur ein kleiner Aspekt meiner Arbeit und wird im Abschnitt 3.5 behandelt. Die Behebung wird innerhalb der Ansätze nur in späteren Phasen der Entwicklung bzw. in wenigen Aktivitäten behandelt.

Die Verbesserungen von festgelegten Prozessabläufen können bei Open Source Software, welche sich unter anderem durch einen flexiblen Verlauf der Entwicklung auszeichnet, kaum in ihrer Gesamtheit Anwendung finden. Dort ist es nur möglich ausgewählte Aktivitäten bestimmten Phasen zuzuordnen. Dafür bietet sich das CLASP Projekt am Besten an. Auch Teile aus dem SDL können in der Open Source Softwareentwicklung Anwendung finden. Die Anwendung des auf Organisationen fixierten Ansatzes des SSE-CMM gestaltet sich bei OSS Projekten schwierig. Denn dort gibt es in der Regel keine festen Strukturen und fixe Zeitpläne für die Projektmitglieder, da es sich bei den meisten OSS Entwicklern um Freiwillige Mitarbeiter an diesem Projekt handelt.

### Open Source Software (OSS)

Bei meiner Untersuchung der Sicherheitsgewährleistung, im Speziellen der Schwachstellenbehebung und -vermeidung, betrachte ich ausschließlich Open Source Software.

Die Open Source Initiative (OSI) [14] definiert Open Source Software als Software, welche die folgenden Kriterien erfüllt:

1. Free Redistribution - keine Einschränkung bei der Verbreitung der Software,
2. Source Code - der Quellcode der Software muss uneingeschränkt und öffentlich verfügbar sein,
3. Derived Works - die Softwarelizenz muss abgeleitete Arbeit und Modifikationen an der Software erlauben,
4. Integrity of The Author’s Source Code - Veränderungen im Code müssen explizit gekennzeichnet sein,

[6] http://www.owasp.org/index.php/OWASP_CLASP_Project überprüft am 14.08.2008

5. No Discrimination Against Persons or Groups - die Softwarelizenz darf keine Personen oder Personengruppen benachteiligen,
6. No Discrimination Against Fields of Endeavor - die Softwarelizenz darf kein Einsatzgebiet für die Software ausschließen,
7. Distribution of License - bei Weitergabe soll die Lizenz weiterhin gültig bleiben,
8. License Must Not Be Specific to a Product - Lizenz darf nicht auf ein bestimmtes Softwarepaket beschränkt sein, innerhalb dessen die lizenzierten Software weitergegeben wird,
9. License Must Not Restrict Other Software - Lizenz darf keine andere Software einschränken, welche zusammen mit der lizenzierten Software weitergegeben wird,
10. License Must Be Technology-Neutral - Lizenz muss technologieneutral sein.

Software, welche die genannte Definition nicht erfüllt und sich von Open Source Software abgrenzt, bezeichne ich im Laufe der weiteren Arbeit als Closed Source Software.

### Open Source und Sicherheit

In der Literatur zu Sicherheit und OSS diskutieren die Autoren vorrangig den Vergleich zwischen der Sicherheit von Open Source und Closed Source Software. Die Argumente dieser Debatte beruhen auf den Vor- und Nachteilen, welche OSS für Sicherheit mitbringt.

Vorteile:

- Analog zu Raymonds [16] bekannten *Linus Law* „Given enough eyeballs, all bugs are shallow.“ sind bei Open Source Software mehr Leute in der Lage, durch den freien Zugriff auf den Quellcode mögliche Schwachstellen zu finden und zu beheben
- Cowan [4] führt als Argument an, dass Hersteller von Closed Source Software uneingeschränkt über die Sicherheit ihrer Anwendungen bestimmen können und der Nutzer dem Hersteller dabei völlig ausgeliefert ist. Bei OSS hingegen hat der Nutzer selber die Möglichkeit den Grad an Sicherheit herbeizuführen, den er wünscht und den der ermöglichen kann.
- Kerckhoffs Prinzip [9] besagt, dass es egal ist, ob der Algorithmus einer Verschlüsselung, welcher die Sicherheit dieses Verfahrens gewährleistet, bekannt ist, wenn der Schlüssel dabei geheim bleibt. Als Analogie zur Open Source Software betrachtet, lässt sich folgende Aussage formulieren: die Geheimhaltung des Quellcodes gewährleistet nicht zwingend die Sicherheit der Software.

Nachteile:

- Jeder hat Zugang zum Code und den darin vorhandenen Schwachstellen, auch mögliche Angreifer [4, 7, 22, 23].
- Auch wenn jeder die Möglichkeit hat den Quellcode anzuschauen, bedeutet das nicht unbedingt, dass eine Überprüfung auf Schwachstellen stattfindet. Lawton [10] zitiert Marcus Ranum[7] als Beispiel für dieses Argument. Ranum entwickelte das erste öffentliche FireWall ToolKit[8] (TIS firewall toolkit), welche von ca. 2000 Seiten eingesetzt wurde. Nur zehn Benutzer dieser Software haben Rückmeldung gegeben oder Patches abgeliefert.
- Viega [20] sagt, dass „die Verfügbarkeit von Quellcode und die Tatsache, dass eine große Zahl an Nutzern existiert, welche die Möglichkeit haben nach Schwachstellen zu schauen und diese zu beheben – den Menschen ein falsches Gefühl der Sicherheit vermittelt“.

Neben der Diskussion zum Vergleich der Sicherheit zwischen Open und Closed Source Software sind in der Literatur Metriken zu finden, welche die Sicherheit von Open Source Software messen soll. Witten et al. [23] schlagen dafür eine einfache Methode vor:

Es wird ein Mittelwert des Zeitraums zwischen der Veröffentlichung einer Schwachstelle und deren Behebung für verschiedene Projekte ermittelt und miteinander verglichen. Der Schweregrad der Schwachstelle, die Effektivität der vermeintlichen Lösungen und die Verzerrung der Mittelwertbildung durch Schwankungen in der Zahl der Schwachstellen bleiben bei dieser Methode unberücksichtigt. Dennoch ist die Dauer des Behebungsverlaufs ein wichtiges Kriterium bei der Einschätzung der Sicherheit einer Software.

### Zusammenfassung

Die Abbildung 1.2 auf der nächsten Seite visualisiert die Einordnung meiner Arbeit in den wissenschaftlichen Kontext innerhalb der Informatik. Der Fokus liegt auf der Sicherheitsgewährleistung innerhalb der Qualitätssicherung im Bereich Softwaretechnik. Innerhalb dieses Bereichs betrachte ich in meiner Arbeit die Schwachstellenbehebung und -vermeidung innerhalb der Open Source Softwareentwicklung (in der Abbildung rot hervorgehoben), wobei der Schwerpunkt auf der Untersuchung der Behebung von Schwachstellen liegt.

[7] http://www.marcusranum.com/ überprüft am 14.08.2008
[8] http://www.fwtk.org/main.html überprüft am 14.08.2008

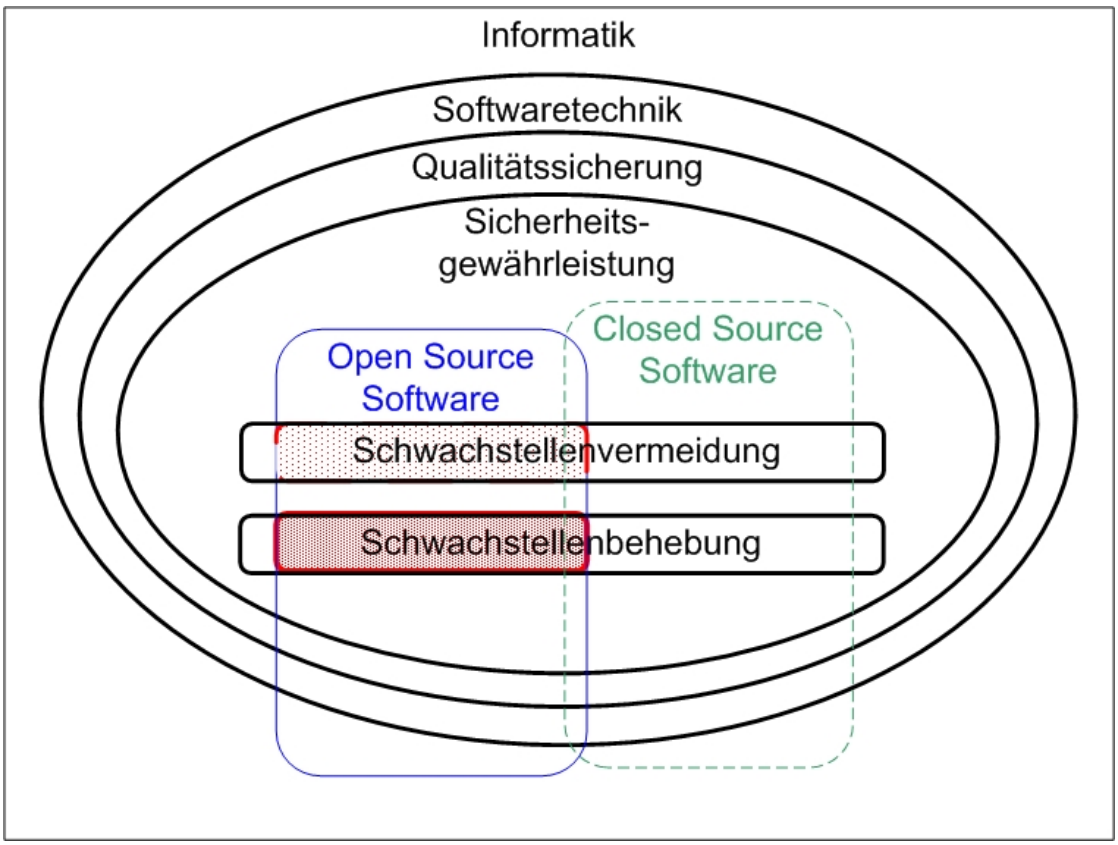


Abb. 1.2: Einordnung meiner Untersuchung in den wissenschaftlichen Kontext

## 1.4 Vorgehen

Bei meiner Arbeit bin ich wie folgt vorgegangen. Nach der Auswahl der Grounded Theory als Methode der Datenanalyse habe ich mich zunächst darin eingearbeitet. Im Abschnitt 2.2 des nächsten Kapitels gehe ich näher auf diese Forschungsmethode ein. Neben der Einarbeitung in die Grounded Theory, habe ich Praktiken adaptiert, welche mich beim Entwickeln von Konzepten unterstützt haben. Diese Praktiken stelle ich im Abschnitt 2.3 des nächsten Kapitels vor.

Danach habe ich angefangen Daten auszuwählen und zu erheben, welche den Verlauf der Behebung von sicherheitsrelevanten Schwachstellen beschreiben. Das Vorgehen der Datenerhebung beschreibe ich im ersten Abschnitt des nächsten Kapitels.

Das erste Ziel der Datenanalyse war die Klassifikation der Verhaltensweisen und der Rollen, welche im Behebungsverlauf auftreten. Dadurch habe ich den den Beobach-

tungsraum näher kennen gelernt und dabei Phänomene konzeptioniert und kategorisiert. Außerdem habe ich interessante Phänomene für die spätere Analyse identifiziert. Diese Untersuchung und die dabei entwickelten Klassifikationen beschreibe ich in den ersten beiden Abschnitten des Kapitels 3.

Aus den interessanten Phänomenen habe ich mir das Phänomen „Umgang mit Schwachstellenveröffentlichung“ ausgewählt und dessen Beziehungen zu anderen Phänomen innerhalb einer weiteren Analyse untersucht, um Aussagen über den Behebungsverlauf zu gewinnen. Darauf gehe ich im Abschnitt 3.3 des Kapitels 3 ein.

Das zweite interessante Phänomen „Drängen nach Schwachstellenbehebung“ stelle ich im Abschnitt 3.4 des Kapitels 3 kurz vor und gebe einen Ausblick darüber, welche Fragen und Vermutungen zu diesem in zukünftigen Untersuchungen erforscht und beantwortet werden können.

Die Verhaltensweisen und Praktiken zur Vermeidung von Schwachstellen, welche ich während meiner Untersuchung gefunden habe, beschreibe ich im Abschnitt 3.5 des Kapitels 3.

In Kapitel 4 beschreibe ich zunächst, welche Herausforderungen während meiner Arbeit aufgetreten sind und wie ich damit umgegangen bin. Danach fasse ich die Erkenntnisse und Erfahrungen, welche ich durch diese Arbeit gewonnen habe zusammen. Im letzten Abschnitt des Kapitels 4 gebe ich einen Ausblick auf weitere mögliche Untersuchungen zum Thema sicherheitsrelevantes Verhalten in Open Source Softwareentwicklung und stelle Phänomene vor, welche vielleicht in zukünftigen Arbeiten betrachtet werden.

# 2 Methodik und Praktiken der Analyse

In diesem Kapitel stelle ich die angewandte Methodik und die Praktiken, welche meine Analyse unterstützt haben, vor. Zuerst gehe ich in Abschnitt 2.1 auf die Auswahl und Erhebung der Daten ein. Danach beschreibe ich in Abschnitt 2.2 die Forschungsmethode Grounded Theory und welches Werkzeug ich bei der Anwendung dieser Methode genutzt habe. Am Ende des Kapitels in Abschnitt 2.3 erläutere ich die Praktiken, welche mir die Anwendung der Forschungsmethode erleichtert haben.

## 2.1 Datenerhebung

Die Datenerhebung ergibt sich aus der Kernaufgabe 1 und bildet die Grundlage, auf welcher später die Datenanalyse stattfindet. Ich habe mich dafür entschieden die Datenquellen aus dem Internet zu beziehen. Zum einen, da sie leicht zugänglich sind, und zum anderen, da die Kommunikation in der Open Source Software Entwicklung über das Internet und dort zu großen Teilen transparent stattfindet. In den folgenden Abschnitten stelle ich die Systematik der Datenerhebung, die Eigenschaften der Datenbasis und die betrachteten Projekte vor.

### 2.1.1 Grundlagen und Projektbestimmung

Ich habe die National Vulnerability Database (NVD)[1] als Ausgangspunkt für die Suche nach geeigneten Daten gewählt. Diese Datenbank umfasst eine durchsuchbare Liste sicherheitsrelevanter Schwachstellen. Darüber hinaus beheimatet die NVD Sicherheits-Checklisten, eine Übersicht über Metriken zu Auswirkungen von sicherheitsrelevanten Schwachstellen und liefert Informationen zur Anwendung und Einhaltung von Protokollen für die Anwendung von Standards, welche automatisiertes Schwachstellen Management, das Messen von Sicherheit und die Einhaltung von Richtlinien ermöglichen. Die letztgenannten Eigenschaften der NVD sind in meiner Arbeit nicht weiter von Bedeutung. Ich gehe hier nur auf die Liste sicherheitsrelevanter Schwachstellen ein.

Der Vorteil der Schwachstellenliste der NVD liegt darin, dass jeder dort verzeichneten Schwachstelle einem Common Vulnerabilities and Exposures (CVE) Identifikator[2]

[1] http://nvd.nist.gov/ überprüft am 14.08.2008
[2] http://cve.mitre.org/cve/identifiers/index.html überprüft am 14.08.2008

zugewiesen wird. Die Vergabe von CVE Identifikatoren hat das Ziel, eine einheitliche Namenskonvention für Schwachstellen einzuführen, damit jede Schwachstelle eindeutig bestimmbar ist. Somit ist es einfacher Daten zu einer bestimmten Schwachstelle, gekennzeichnet durch ihren eindeutige CVE Identifikator, zwischen verschiedenen Datenbanken und Dienstleistungen auszutauschen.

Die Common Weakness Enumeration (CWE)[3], welche die NVD ebenfalls innerhalb der Schwachstellenübersicht anwendet, stellt eine Klassifikation zur Verfügung, welche die verschiedenen Arten von Schwachstellen einer eindeutigen Kennung zuweist und diese beschreibt. Diese Vereinheitlichung soll Entwicklern und Sicherheitsfachleuten dabei helfen, Schwachstellen genauer zu beschreiben und einzuschätzen.

Neben dem CVE Identifikator und der CWE bietet diese Schwachstellenliste zusätzliche optionale Informationen, wie die standardisierte Einstufung der Schwachstelle in das Common Vulnerability Scoring System Version 2 (CVSS)[4], eine Kurzbeschreibung der Schwachstelle, die Angabe der betroffenen Version, Links zu weiteren Informationen, die Behebungsmöglichkeit der Schwachstelle und weitere technische Details.

Für die Auswahl geeigneter Projekte habe ich in chronologisch absteigender Reihenfolge alle Schwachstellen aus der NVD im Zeitraum vom 01.01.2007 bis 01.03.2008 angeschaut. Die dabei entstandene Schwachstellenliste habe ich aufgrund folgender Filterkriterien weiter eingegrenzt:

- Open Source Software,
- Webanwendungen,
- die sicherheitsrelevante Schwachstelle, welche dort beschrieben wird, ist bereits vom Hersteller behoben.

Die ersten zwei Kriterien ergeben sich aus der Aufgabenstellung. Der letzte Punkt ist relevant, da ich den Verlauf der Behebung untersuche und somit sind primär nur die Schwachstellen und dazugehörige Projekte interessant, die vom Hersteller behoben wurden. Für eine weitere Untersuchung können auch Schwachstellen ohne Behebung wichtig sein, um sie z.B. für Vergleiche heranziehen.

Ich habe dann Projekte herausgesucht, welche zu diesem Zeitpunkt die mit einer Vielzahl an Schwachstellen in dieser Liste vertreten waren. Ich habe nach Projekten gesucht, welche mit um oder mehr als 100 Einträge in der Suchergebnisliste vertreten waren. Diese Zahl erschien für mich angemessen hoch, um dadurch ein Projekt auszuwählen, welches genügend Schwachstellen und dazugehörige Daten aufzuweisen hat.

Ich habe außerdem auf Nachrichten zu sicherheitsrelevanten Themen aus IT-Bereich geachtet, um weitere Kandidaten zu finden. Dieses Verhalten entstand zum einen aufgrund

[3] http://cwe.mitre.org/ überprüft am 14.08.2008
[4] http://www.first.org/cvss/cvss-guide.html überprüft am 14.08.2008

des Themas meiner Masterarbeit und zum anderen aufgrund des vorangegangenem Projektauswahlprozesses. So habe ich z.B. auf Newsseiten zur IT wie heise Security[5], aber auch auf Blogs, welche sich mit dem Thema Sicherheit und Schwachstellen beschäftigen[6], von weiteren möglichen Projekten erfahren.

Darüber hinaus habe ich selber am Anfang Projekte in Betracht gezogen, welche zu den OSS Webanwendungen gehören und die ich aufgrund längerfristiger persönlicher Anwendung, selbständiger Konfiguration und Erweiterung kenne.

Bei der anfänglichen Datensammlung habe ich folgende Projekte durch die NVD Suchergebnisse, Berichte zu sicherheitsrelevanten Themen und aufgrund persönlicher Benutzung als mögliche Kandidaten für meine Datenanalyse ausgewählt:

- Drupal[7]
- Horde[8]
- Wordpress[9]
- Joomla[10]

Im Abschnitt 2.1.4 erkläre ich diese Projekte und gehe dort näher auf die projektspezifische Datenerhebung ein.

### 2.1.2 Datenerhebung innerhalb der Projekte

Nachdem ich die zwei Projekte Wordpress und Joomla für die anfängliche Untersuchung ausgewählt habe, bestand der nächste Teil der Datenerhebung darin zu diesen Projekten Daten zum Behebungsverlauf von Schwachstellen zu sammeln. Dabei habe ich über die NVD zu den Projekten gehörende Schwachstellen aus dem Jahr 2007 herausgesucht. Die zu einer Schwachstelle gehörenden Datenquellen habe ich anhand der CVE Kennung oder über weiterführende Links in den Schwachstellenbeschreibungen identifiziert. Diese Datenquellen liegen in Form von Mailinglisten-, Bugtrackereinträge, Schwachstellenbeschreibungen aus anderen Schwachstellendatenbanken, Chatlogs, Blogeinträgen oder Forenbeiträgen vor. Wie am Anfang des Abschnitts schon erwähnt, habe ich mich dazu entschlossen Daten zu erheben, welche frei im Internet verfügbar sind und darunter fallen die genannten Formen von Datenquellen. Nachdem ich genügend Datenquellen zu

[5] http://www.heise.de/security/ überprüft am 14.08.2008
[6] z.B. http://osvdb.org/blog, http://www.notsosecure.com, http://www.hackerscenter.com/ alle überprüft am 14.08.2008
[7] http://www.drupal.org/ überprüft am 14.08.2008
[8] http://www.horde.org/ überprüft am 14.08.2008
[9] http://www.wordpress.org/ überprüft am 14.08.2008
[10] http://www.joomla.org/ überprüft am 14.08.2008

einer Schwachstelle, d.h. der Verlauf der Schwachstellenbehebung ist durch die erhobenen Datenquellen nachvollziehbar, gefunden habe, habe ich sie zu Episoden zusammengefasst.

> Eine Episode stellt eine Sammlung aller Dokumente und weiterer Informationen zu einer Schwachstelle dar. Der Behebungsverlauf einer Schwachstelle wird anhand einer Episode dokumentiert und zu einem Projekt können mehrere Epsioden gehören.

### 2.1.3 Eigenschaften der Datenbasis

Wie im vorigen Abschnitt erläutert liegen die Datenquellen in verschiedenen Formen vor. Außerdem sind die Datenquellen einer URL zuordnenbar und somit als Webseite in HTML im Internet eindeutig zu identifizieren. Dadurch ist es mir möglich diese unterschiedlichen Formen von Datenquellen als statische HTML Seite zu speichern, so dass mir die Datenquellen für meine Analyse als einzelne Dokumente in Textform vorliegen. Somit gehören verschiedene Dokumente zu einer Episode und zu einem Behebungsverlauf einer Schwachstelle. Diese Dokumente können über Verweise (Hyperlinks) untereinander in Beziehung stehen.

Eine weitere Eigenschaft eines solchen Dokumentes wird in Abbildung 2.1 auf der nächsten Seite dargestellt. Eine Person ist entweder ausschließlich als Autor für den gesamten Textes eines Dokumentes verantwortlich oder mehrere Autoren können zu einzelnen Texteinträgen gehören, welche innerhalb des Dokumentes eine Diskussion führen. Diese Eigenschaft greife ich später beim Konzeptnamensschema und der Betrachtung des Analyse-Modells im Abschnitt 2.3 auf.

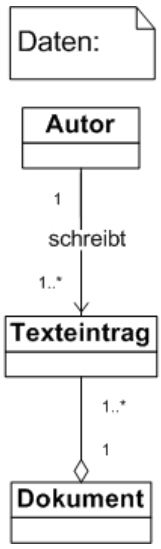


Abb. 2.1: Dokument-Texteintrag-Autor Beziehung

### 2.1.4 Projekte zu Beginn der Analyse

Im Folgenden werden die Eigenschaften der anfänglich gefunden Projekte beschrieben. Dabei gehe ich auf die Besonderheiten und auf die Probleme bei der Datenerhebung ein. Die Episoden werden im Anhang A.4 näher beschrieben.

#### Drupal und Horde

Drupal und Horde sind beides Projekte, die ich aufgrund längerfristiger persönlicher Anwendung kenne und bei denen ich während des Einsatzes Veränderungen an der Anwendung durchgeführt habe. Über die Schwachstellenliste der NVD habe ich bei beiden Projekten Schwachstellen identifizieren können. Bei Drupal waren es 99 inklusive Module und bei Horde waren es 48 Treffer in der NVD Suchergebnisliste.

Drupal ist eine Open Source Webanwendung unter GNU General Public License (GPL)[11] und stellt ein Content Management System (CMS) dar. Dabei liegen das Einsatzgebiet von Drupal im Aufbau von Gemeinschaftsseiten. Es ist in PHP geschrieben und unterstützt die Datenbanken MySQL und PostgreSQL.

Horde ist ein Open Source Framework unter der GNU Lesser General Public License (LGPL)[12], welches verschiedene Anwendungen anbietet. Darunter fallen Webmail, Group-

[11] http://www.gnu.org/licenses/gpl.html überprüft am 14.08.2008
[12] http://www.gnu.org/licenses/lgpl.html überprüft am 14.08.2008

ware und Entwicklungstools. Es ist ebenso wie Drupal in PHP geschrieben und bietet Anbindung an verschiedene Datenbanken.

Obwohl Drupal knapp 100 Schwachstellenmeldungen im NVD verzeichnete, wurde die eigentliche Datenerhebung durch den Umgang mit Sicherheit durch Drupal verhindert. Drupal setzt ein eigenes Sicherheitsteam ein, an das Entdeckungen von Schwachstellen gemeldet werden sollen. Die Diskussionen zur Behebung findet innerhalb einer geschlossenen Sicherheits-Mailingliste statt. An die Öffentlichkeit gelangt die volle Schwachstellenveröffentlichung, sowie eine mögliche Schwachstellenbehebung, nachdem die Behebung stattgefunden hat. Darüber hinaus werden keine weiteren Informationen zum Behebungsverlauf veröffentlicht. Auch die direkte Nachfrage an das Sicherheitsteam, auf einen Teil der Daten der geschlossenen Sicherheits-Mailingliste zugreifen zu können, blieb aussichtslos. Somit musste ich ein Projekt verwerfen, welches viele Schwachstellenmeldungen verzeichnet und welches ich durch persönlichen Einsatz gut kennengelernt habe.

Horde bot mit seinen 48 Treffern bei genauerer Überprüfung zu wenig brauchbare Daten. Bis auf die Schwachstelleninformationen in der NVD und anderer verlinkter Schwachstellendatenbanken fand ich keine brauchbaren Datenquellen, welche den Verlauf der Schwachstellenbehebung nachvollziehbar gestalteten.

Diese beiden Projekte gaben mir nicht genügend vollständige Daten für die weitere Analyse und wurden als Projekte für meine Untersuchung verworfen. Nach diesen Rückschlägen in der Datenerhebung habe ich mir das Projekt Wordpress angeschaut.

### Wordpress

Wordpress ist eine Open Source Webanwendung, welche als Weblog-Publishing-System eingesetzt wird, d.h. diese Anwendung dient der Bereitstellung von Weblogs. Es ist in PHP geschrieben, bietet eine Anbindung zu einer MySQL Datenbank und steht unter der GPL. Wordpress zeichnet sich durch die Einhaltung von Webstandards nach dem World Wide Web Consortium (W3C)[13] aus. Das NVD listete 113 Schwachstellen Treffer zu Wordpress inklusive dazugehöriger Plugins auf.

Das Wordpress Team hat eine Kontakt Email-Adresse für die Meldung von sicherheitsrelevanten Schwachstellen an das Team eingerichtet. Darüber hinaus besteht die Möglichkeit Schwachstellenmeldungen als Ticket im Bugtracker von Wordpress unter der Komponente Security zu eröffnen.

Die konkrete Erhebung der Wordpress Episoden ging von dem Bugtracker[14] aus. Dabei habe ich eine Abfrage erzeugt, welche mir alle Schwachstellen von Wordpress, welche

[13] http://www.w3.org/ überprüft am 14.08.2008
[14] http://trac.wordpress.org/ überprüft am 14.08.2008

innerhalb des Bugtrackers bearbeitet wurden, in einer Liste ausgab. Die Abfrage[15] wurde wie folgt eingeschränkt:

**status=closed** Geschlossene Tickets zusammen mit resolution=fixed deuten auf erfolgreiche Behebung hin.

**status=reopened** Fand eine Wiedereröffnung eines Tickets statt? Was sind Gründe dafür und können sie weitere Hinweise geben?

**component=security** Ich betrachte Tickets in denen sicherheitsrelevante Schwachstellen bearbeitet werden.

**resolution=fixed** Es wird der Verlauf der Behebung untersucht. Nur die Fälle interessant, bei denen eine erfolgreiche Behebung stattgefunden hat.

**order=priority** Untersuchung verschiedener Episoden aufgrund ihrer zugewiesenen Priorität. Die Sortierung hilft erste Episoden aufgrund der Wichtigkeit auszuwählen. Die nachrangig standardmäßige Sortierung ist die Sortierung nach der TicketID. Sie gibt indirekt einen Hinweis auf das Eröffnungsalter des Tickets, je niedriger eine ID, desto älter ist das Ticket.

Aus dieser Liste wählte ich fünf Tickets mit unterschiedlicher Priorität aus. Ich habe die Episoden anschließend aufgrund der folgenden Faktoren ausgewählt:

- Hohe TicketID, damit recht junge Ticketeinträge,
- die Tickets sollten bis zum 01.01.2007 zurückreichen,
- interessante und reichhaltige Tickethistorie,
- weitere Informationen zur Schwachstelle, welche im Ticket beschrieben wird, sind vorhanden, z.B. Schwachstellendatenbankeintrag, Diskussion im Wordpressforum, Blogeinträge, Mailinglistendiskussion.

Die Bugtrackereinträge und die Dokumente, welche letzten Punkt beschrieben werden, bilden die Episoden des Projekts Wordpress. Im Rahmen der Analyse habe ich innerhalb der Datenquellen weitere Verknüpfungen zu neuen Datenquellen gefunden und diese zur jeweiligen Episode hinzugefügt. Wordpress bildete mit seinen fünf Episoden die Grundlage für die ersten Schritte der Analyse (offenes Kodieren siehe Abschnitt 2.2).

### Joomla

Nach der Wordpress Datenerhebung stellte ich folgende Anforderungen an die nächsten Projekte:

[15] URL zur Abfrage:
http://trac.wordpress.org/query?status=reopened&status=closed&component=Security&resolution=fixed&order=priority
überprüft am 14.08.2008

- Webanwendung,
- vom Typ Content Management System, Newsseite, Gemeinschaftsportal oder Blogsoftware ,
- Lieferung genügender Anhaltspunkte für qualitative Datenanalyse.

Diese Anforderungen dienten mir dazu ein Projekt zu finden, welches funktional vergleichbar mit Wordpress ist. Die ersten beiden Kriterien erfüllten die beiden Content Management Systeme Drupal und Joomla, welche unter meinen ersten interessanten Projekten zu finden sind. Aber wegen dem eingeschränkten Umgang mit sicherheitsrelevanten Informationen war das Projekt Drupal nicht mit der dritten Anforderung vereinbar. Somit blieb mir als Folgeprojekt zu Wordpress die Open Source Webanwendung Joomla.

Joomla bietet viele Ähnlichkeiten zu Wordpress. Es basiert auch auf PHP, besitzt als Datenbankanbindung MySQL und steht ebenso unter der GPL. Mit 173 Schwachstellenangaben inklusive Module bot Joomla sogar die meisten Schwachstelleneinträge der betrachteten Projekte im NVD zum Zeitpunkt meiner Suche.

Das Joomla Team bittet darum, dass der Fund von kritischen Schwachstellen als Ticket im Bugtracker gemeldet wird. Auf vom Team als kritisch bewertete Schwachstellen reagiert das Joomla Team innerhalb von 24-48 Stunden. Weniger kritisch eingestufte Schwachstellen werden je nach ihrer Art behandelt. Auf jeden Fall liegt die dafür vorgesehene Bearbeitungszeit über den 48 Stunden[16].

Für die Datenerhebung bietet Joomla neben dem Bugtracker, zwei Sicherheitsforen (je eins für die beiden Versionszweige), eine Sicherheits-FAQ und eigene Übersicht zu bekannten Schwachstellen in den Modulen. Die Erstellung der einzelnen Episoden habe ich durch zwei Vorgehensweisen realisiert. Zum einen bin ich die beiden Foren[17] zur Sicherheit durchgegangen und habe interessante und reichhaltige Forendiskussionen zu konkreten Schwachstellen als Datenquellen für einzelne Episoden ausgewählt. Zum anderen habe im Joomla Bugtracker[18] im Zeitraum vom 01.01.2007 bis 31.12.2007 nach Begriffen zur Sicherheit (z.B. *Vulnerability, Security, Hack, XSS*) gesucht um Bugtrackereinträge zu Schwachstellen zu erhalten. Die Recherche im Forum ergab drei Episoden, wovon die Erste mit einem Bugtrackereintrag verknüpft ist. Die Suche nach Schwachstelleneinträgen im Bugtracker ergab sechs weitere Episoden.

Alle neun Episoden in Joomla beginnen im Behebungsverlauf mit einer direkten Schwachstellenveröffentlichung oder dem Verweis auf eine Veröffentlichung zu einer Schwachstelle

[16] http://forum.joomla.org/viewtopic.php?f=267&t=70117&st=0&sk=t&sd=a&start=30#p373406 überprüft am 14.08.2008

[17] http://forum.joomla.org/viewforum.php?f=267 überprüft am 14.08.2008
http://forum.joomla.org/viewforum.php?f=432 überprüft am 14.08.2008

[18] http://joomlacode.org/gf/project/joomla/search/?action=index.php&project_id=5 überprüft am 14.08.2008

in Joomla. Dieses Phänomen habe ich im späteren Verlauf meiner Arbeit untersucht und alle erhobenen Episoden in Joomla sind in diese Untersuchung eingeflossen. Die Ergebnisse der Untersuchung bespreche ich in Abschnitt 3.3.

## 2.2 Forschungsmethode Grounded Theory

Die Grundlage meiner Forschungsarbeit bildet die Grounded Theory nach Strauss und Corbin [18], da dies eine bewährte Forschungsmethode ist um wissenschaftliche Theorien zu entwickeln. Im Deutschen wird diese Methode als gegenstandsverankerte Theoriebildung oder als Entwicklung von in Daten begründeten Theorien bezeichnet. Bei der Grounded Theory handelt es sich um eine Methode der qualitativen Forschung, welche die Analyseergebnisse nicht statistisch erhebt, sondern reiche und präzise Beschreibungen, Interpretationen oder Erklärungen zum Untersuchungsgegenstand liefert.

Ziel dieser Methode ist es induktiv aus der Untersuchung eines Gegenstands Erkenntnisse zu gewinnen und eine Theorie zu entwickeln. Es stehen somit am Anfang keine Hypothesen zu dem Gegenstand zu Verfügung, welche im Laufe der Untersuchung bewiesen, bestätigt oder verworfen werden, sondern nur der Untersuchungsbereich, zu welchem der Gegenstand gehört und welcher in den Fokus der Betrachtung rückt. Den Begriff Gegenstand werde ich im weiteren Verlauf als Phänomen bezeichnen. Strauss und Corbin [18] definieren das Phänomen wie folgt:

> „Die zentrale Idee, das Ereignis, Geschehnis, der Vorfall, auf den eine Reihe von Handlungen oder Interaktionen gerichtet ist, um ihn zu kontrollieren oder zu bewältigen oder zu dem die Handlung in Beziehung stehen.“

Um von dem Phänomen zu einer Theorie zu gelangen geben Strauss und Corbin [18] dem Forscher Verfahren an die Hand, welche zu methodischem, wissenschaftlichem und schrittweisem Vorgehen innerhalb der Untersuchung anleiten. Durch diese Verfahren wird „systematisches Erheben und Analysieren von Daten, die sich auf das untersuchte Phänomen beziehen“ [18] ermöglicht. Bei den Verfahren handelt es sich um Offenes Kodieren, Axiales Kodieren und Selektives Kodieren, welche ich in den folgenden Abschnitten vorstellen möchte.

Darüber hinaus gehen Strauss und Corbin [18] noch auf weitere Vorgehen ein. Zum einen wie die theoretische Sensibilität also das „Bewußtsein für die Feinheiten der Bedeutung von Daten“ [18] erhöht wird. Zum anderen geben sie Anleitung zum theoretischem Sampling, d.h. wie im späteren Verlauf der Untersuchung neue Daten, unter dem Aspekt das bis dahin erlangte Theoriekonstrukt zu unterstützen, ausgewählt werden können.

### 2.2.1 Offenes Kodieren

Der Forscher vergibt beim offenen Kodieren Codes, im folgenden Konzepte genannt, welche für die Phänomene in den Datenquellen stehen. Dabei sind Konzepte als eigenständige Bezeichnung für das Phänomen im kodierten Datenabschnitt zu sehen und sollen nicht den Inhalt des Phänomens zusammenfassen bzw. nacherzählen. Die Phänomene können beispielsweise in Textpassagen, Abschnitten, Sätzen oder als einzelne Wörter auftreten, wenn es sich bei der zugrunde liegenden Datenquelle um Texte handelt. Jedes neu gefundene Phänomen wird mit den bereits entwickelten Phänomenen verglichen. Ähnliche Phänomene erhalten dabei denselben Namen oder erweitern ihn bzw. vergeben einen neuen Namen aufgrund neuerer Erkenntnisse.

Das Vergeben von Konzepten sollte aufgrund des ausgewählten Untersuchungsbereichs geschehen. Dabei hilft es sich beim Kodieren die Fragen „Was ist das?“ und „Was repräsentiert es?“ beim Betrachteten der einzelnen Stellen in Daten zu stellen. Weitere Hilfsmittel, die mir beim offenen Kodieren geholfen haben, stelle ich in Abschnitt 2.3 vor.

Verschiedene Konzepte, welche sich auf ein Phänomen beziehen, können zu einer Kategorie gruppiert werden, also zu einem Konzept für das Phänomen auf einer abstrakteren Ebene. Diesen Kategorien weist man Eigenschaften zu, welche das verschiedenartige Auftreten des zugrunde liegende Phänomens charakterisieren. Die möglichen Ausprägungen dieser Eigenschaften werden Dimensionen genannt.

Das Vorgehen des offenen Kodierens nahm einen Großteil meiner Arbeit in Anspruch. Als Ergebnis entstand eine Klassifikation von Verhaltenskategorien und -konzepten, welche bei der Behebung von sicherheitsrelevanten Schwachstellen auftreten. Diese Klassifikation werde ich im Abschnitt 3.1 vorstellen. Bei der Erstellung der Klassifikation bin ich auf Phänomene aufmerksam geworden, die ich aufgrund ihres Auftretens und ihrer Behandlung durch die in den Daten beteiligten Personen näher untersucht habe. Diese Phänomene bilden den Gegenstand meiner Untersuchungen im späteren Verlauf meiner Arbeit. Sie stellen damit ausgewählte Aspekte bei der Betrachtung des Verhaltens im Behebungsverlauf dar. Das Vorgehen und die Ergebnisse dieser Untersuchung stelle ich im Abschnitt 3.3 vor.

#### Zur Entscheidung des Neukodierens

Ich habe mich dazu entschlossen ein neues Kodierschema zu entwickeln. Der wichtigste Grund für diese Entscheidung war, dass ich für den Untersuchungsbereich des sicherheitsrelevanten Verhaltens kein vorgegebenes allgemeines Kodierschema gefunden habe, auf dem ich hätte aufbauen können. Hinzu kommt, dass ein solches allgemeines Kodierschema nicht direkt auf den Aspekt der OSS Entwicklung angewendet werden könnte. Unterschiede und Besonderheiten müssten hierbei ebenso herausgearbeitet werden, wie bei

der Entwicklung eines neuen Kodierschemas. Außerdem stellt sich noch die Bedingung, dass menschliches Verhalten im Hinblick auf die Behebung von sicherheitsrelevantem Verhalten untersucht wird.

Ein weiterer Grund für das Neukodieren lag darin, dass sich ein großer und von meiner Seite noch unerschlossener Untersuchungsbereich vor mir erstreckte. Erst durch das offene Kodieren ergaben sich konkreten Phänomene, die ich zum Gegenstand genauerer Untersuchungen auswählen konnte.

### 2.2.2 Axiales Kodieren

Beim axialen Kodieren werden die im offenen Kodieren entwickelten Kategorien im Hinblick auf die Daten wieder neu zusammengesetzt, um Erkenntnisse über spezielle Phänomene, die beim offenen Kodieren entdeckt wurden, zu gewinnen. Dies geschieht anhand von Verfahren, durch welche die Kategorien miteinander in Beziehung gesetzt und anschließend untersucht werden. Diese Beziehungen werden nach dem von Strauss und Corbin [18] vorgeschlagenem paradigmatischen Modell, wie in Abbildung 2.2 dargestellt, aufgebaut. Das paradigmatische Modell ist ein bewährtes Modell in der Sozialforschung und hilft dem Forscher die Beziehungen zwischen den Kategorien systematisch aufzubauen.

Abb. 2.2: Paradigmatisches Modell

Am Anfang der Beziehungskette stehen die ursächlichen Bedingungen, welche zum Entstehen oder der Entwicklung des Phänomens führen. Das Phänomen ist der eigentliche Kern der Untersuchung. Der Kontext, in dem sich das Phänomen befindet, umfasst die konkret ausgeprägten Eigenschaften, die zu dem Phänomen gehören aber auch die Umgebung in der das Phänomen auftritt. Die intervenierenden Bedingungen haben positiven bzw. negativen Einfluss auf die Aktionen bzw. Strategien, welche auf das Phänomen gerichtet sind. Handlungs- und interaktionale Strategien werden eingesetzt, um das Phänomen zu kontrollieren, es zu lösen, es auszuführen oder darauf zu reagieren. Durch die Anwendung von Strategien auf das Phänomen entstehen als Ergebnis Konsequenzen.

Das Setzen der Kategorien in erste hypothetische Beziehungen mit Hilfe des paradigmatischen Modells ist der erste Schritt des axialen Kodierens. Im zweiten Schritt verifiziert der Forscher anhand der Daten diese Hypothesen. Danach sucht er im dritten Schritt nach weiteren Eigenschaften und dimensionalen Ausprägungen der in Beziehung gesetzten

Kategorien. Im letzten Schritt werden durch die dimensionale Einordnung dieser Kategorien aus den Daten Muster entdeckt, welche die in Beziehung stehenden Kategorien auf dimensionaler Ebene verknüpfen.

Bei mir kam das axiale Kodieren im späteren Verlauf meiner Arbeit zur Anwendung. Es diente mir dazu, dass durch das offene Kodieren hervorgetretene Phänomen „Umgang mit Schwachstellenveröffentlichung“ besser verstehen zu können. Dabei war mir vor allem wichtig die Bedingungen zu identifizieren, welche auf dieses Phänomen Einfluss haben und wie dieser Einfluss ausprägt ist. Im Abschnitt 3.3 stelle ich das Vorgehen und die Ergebnisse des axialen Kodierens vor.

### 2.2.3 Selektives Kodieren

Beim Vorgehen des selektiven Kodierens entwickelt der Forscher die Theorie. Er wählt eine aus den beim offenen und axialen Kodieren entwickelten Kategorien, welche als zentrales Phänomen immer wieder auftaucht, als Kernkategorie aus. Diese Kernkategorie wird in einer beschreibenden Darstellung als zentrales Phänomen eingebettet. Die Offenlegung des roten Fadens dieser Geschichte ist der erste Schritt des selektiven Kodierens. Als Nächstes wird die Kernkategorie wie beim axialen Kodieren mit anderen Kategorien in Beziehung gesetzt. Diese Verbindung findet auch auf dimensionaler Ebene statt, d.h. welche dimensionalen Ausprägungen der Eigenschaften einer Bedingung führt zu welcher dimensionalen Ausprägungen der Eigenschaften der Kernkategorie. Danach werden die Beziehungen zwischen Kategorien und Kernkategorie in den Daten validiert. Zum Schluss werden Kategorien bzw. Details zu den Kategorien hinzugefügt, welche noch Lücken aufweisen. Diese Schritte sind von Strauss und Corbin [18] der Darstellung des Zusammenhangs wegen schrittweise aufgeführt und sollten in der Praxis je nach Bedarf fließend und iterativ angewendet werden.

Dieses Vorgehen fand bei mir keine Anwendung, da ich nur ein einzelnes Phänomen, welches bei der Behebung von sicherheitsrelevanten Schwachstellen auftritt, betrachtet habe. Dieses Phänomen habe ich während des offenen Kodieren herausgearbeitet und mit dem axialem Kodieren näher untersucht. Somit steht am Ende meiner Arbeit keine Theorie, welche das Verhalten im Behebungsverlauf bei OSS beschreibt, sondern ein Reihe von Aussagen über Verhaltens- und Rollenkategorien, die Einfluss auf den Behebungsverlauf haben.

### 2.2.4 Werkzeug zur Unterstützung der Datenanalyse atlas.ti

Am Anfang meiner Arbeit habe ich eine Evaluierung bezüglich CAQDAS (Computer Assisted Qualitative Data Analysis) Software durchgeführt. Diese Softwareanwendungen dienen zur Unterstützung der qualitativen Datenanalyse. Dabei habe ich mir Freie

Software bzw, Demo Versionen kommerzieller Software angeschaut. Die Software sollte zu diesem Zeitpunkt folgende Funktionen unterstützen:

- Kodieren von Textdokumenten,
- visuelles Hervorheben codierter Textpassagen damit ich den Überblick habe, welche Stelle mit welchem Konzept versehen ist,
- schnelles Versehen von Textpassagen mit neuen Konzepten und und die Möglichkeit bestehende Konzepte auszuwählen,
- statistische Informationen von Codes sollten abrufbar sein, z.b. wie viel Konzepte in welchen Dokumenten verwenden,
- die Möglichkeit Konzepte mit Eigenschaften zu versehen und diese ggf. zu dimensionalisieren.

Ich habe mich im Endeffekt für die kommerzielle Anwendung atlas.ti entschieden. Die Gründe für diese Entscheidung sind:

- Ich habe vor der Evaluierung damit angefangen zu arbeiten.
- Diese Anwendung wird in der Arbeitsgruppe Software Engineering eingesetzt.
- Es ist die einzige kommerzielle Software, die ich gefunden habe, welche ohne zeitliche Begrenzung im Demobetrieb eingesetzt werden kann.
- Sie ermöglicht benutzerfreundlich den Einsatz der oben beschriebenen Funktionen.

Im Laufe der Arbeit hat sich Atlas.ti als leistungsfähiges und gut zu bedienendes Unterstützungswerkzeug für das Vorgehen des offenen Kodierens erwiesen.

In der Anlage der Arbeit habe ich meine Atlas.ti Daten und eine Demoversion des Programms, zum Betrachten der Daten, auf CD hinterlegt. Im Anhang A.5 ist eine Beschreibung der Atlas.ti Daten und eine Anleitung zur Installation der Demo zu finden.

## 2.3 Angewandte Praktiken bei der Datenanalyse

Nach Identifikation der Projekte und Erstellung erster Episoden aus diesen Projekten war eine große Datenmenge vorhanden. Mit großer Datenmenge ist hier auf der einen Seite gemeint, dass zu einer Episode viele Dokumente (im Durchschnitt 7-9 Dokumente je Episode) unterschiedlicher Größe vorliegen. Auf der anderen Seite meint große Datenmenge, dass viele Personen innerhalb eines Dokumentes, welches in Form eines Forumsbeitrags, Mailinglistenausschnitts oder Chatlogs vorliegt, auftreten und diese abgegrenzt voneinander betrachtet werden müssen.

Nach meinem erstem Herangehen an das offene Kodieren der ersten Wordpress Episode war eine Menge von 77 Konzepten vorhanden. Diese Konzepte konnten auf unterschiedlichen Granularitätsebenen angeordnet werden (z.B. die Konzepte „Behebung“, „,Keine Angabe weiterer Details trotz Recherche nach Anleitung“). Innerhalb der Konzeptmenge gab es eine Vermischung von Verhaltens- und Rollenkonzepten ohne klare sichtbare Abgrenzung dieser Konzepte.

Salinger et al. [17] schlagen Praktiken vor, um Abgrenzungen zwischen den Konzepten und die Beibehaltung der gleichen Granularitätsebene aller Konzepte zu gewährleisten. Darüber hinaus helfen die Praktiken eine übersichtliche Zahl an Konzepten für große Datenmengen zu entwickeln. Ich adaptierte diese Praktiken, da ich damit oben genannte Probleme in den Griff bekomme und weil sie mich beim Vorgehen des offenen Kodierens unterstützen. Diese Praktiken sind:

- Blickwinkel auf die Daten,
- Konzeptnamensschema,
- Analyse Meta-Modell,
- Paar Kodierung.

Da ich meine Arbeit ohne weitere Hilfe durchgeführt habe, entfällt Anwendung der Praxis „Paar Kodierung“, welche die Ausführung der Kodierarbeit als Paar an einem Computer vorschlägt. Die Beschreibung und meine Umsetzung der übrigen Praktiken beschreibe ich einzeln in den folgenden Abschnitten.

### 2.3.1 Blickwinkel

Salinger et al. [17] schlagen vor den Blickwinkel auf die Daten im Gegensatz zu Strauss und Corbin [18] schon vor dem Beginn des Kodierprozesses (offenes Kodieren) festzulegen. Der Blickwinkel soll anhand der folgenden drei Fragen festgelegt werden:

1. Unter welchen Gesichtspunkten erwarte ich in den Daten Erkenntnisse zu finden?
2. Welche Arten von Phänomene möchte ich bei der Betrachtung der Daten zulassen?
3. Zu welcher Art von Ergebnissen will ich gelangen?

Die erste Frage beantworte ich damit, dass die Ergebnisse mir dabei helfen sollen, zu verstehen welche Verhaltensweisen und Rollen bei Behebung von Sicherheitslücken in OSS relevant sind.

Die Antwort zur zweiten Frage ergibt sich durch die Form der mir zur Verfügung stehenden Daten. Da sie ausschließlich in Textform vorliegen, sollen nur die Phänomene, die direkt aus den Daten erhoben werden können, betrachtet werden. Weitere Ausführungen und

Interpretationen über die klar vorliegenden Daten hinaus führen zu subjektiv geprägten und nicht zwangsweise nachvollziehbaren Konzepten.

Das Ergebnis, welches in der letzten Frage gefordert wird, soll die Entwicklung einer Klassifikation von Verhaltensweisen und eine von Rollen, welche bei der Behebung von sicherheitsrelevanten Schwachstellen in OSS zum Tragen kommen, sein. Salinger et al. [17] bezeichnen solch eine Klassifikation als Kodierschema. Darüber hinaus sollen interessante Phänomene innerhalb dieser Klassifikation aufgedeckt werden und mir weitere und tiefer gehende Untersuchung ermöglichen. Ziel dieser weiterführenden Untersuchung ist es, das Phänomen besser zu verstehen und Aussagen über Einflüsse auf den Behebungsverlauf aufzustellen.

### 2.3.2 Konzeptnamensschema

Die zweite Praxis, welche die Datenanalyse unterstützt, ist, wie von Salinger et al. [17] vorgeschlagen, die Benutzung eines strukturierten Namensschemas für die entwickelten Konzepte. Ein Vorteil eines solchen Schemas ist, dass man sich bei der Vergabe des Konzeptnamens viel intensiver mit dem zu Grunde liegenden Phänomen beschäftigt, weil es in die vorgegebene Syntax passen muss. Erste Ideen bei der Vergabe von neuen Namen prüfte ich meist länger als bei Konzeptnamen, welche ich vor dem Einsatz dieser Praxis entwickelt habe. Der Vergleich von Konzepten, welche bei der Grounded Theory durch das immerwährende Vergleichen gefordert wird, ist bei Konzepten mit gleicher Syntax einfacher als bei Konzepten, die unterschiedliche Formen besitzen. Außerdem können strukturiert aufgebaute Konzeptnamen besser gruppiert werden, was bei entsprechendem Gebrauch zu einer Vereinfachung der Kategorienbildung führt. Und durch diese Gruppierung ist es einfacher den Überblick über eine große Menge von Konzepten zu behalten.

Die ursprüngliche Struktur der Konzeptnamens von Salinger et al. [17] ist folgende:

```
code = <actor>.<description>
```

Der Konzeptname besteht aus einem Akteur verknüpft mit der Beschreibung einer Aktion, die von dem Akteur ausgeführt wird. Bei mir besteht der Teil <actor> oder allgemein Akteur aus einer breiteren Auswahl an beteiligten Personen als bei Salinger et al. [17], welche aufgrund der Untersuchung von Paar Programmierern nur maximal zwei Personen betrachten. Jeden zu einem Akteur zuordnenbare Texteintrag in den Daten versehe ich mit dem entsprechenden <actor> Teilkonzept. Dieses Teilkonzept besteht aus drei Teilen:

```
actor = _<document_ID>_<actor_id>_<additional_document_IDs>
```

Die Dokument ID ergibt sich aus der Identifikationsnummer für Dokumente, welche von atlas.ti vergeben wird. Diese hilft mir sofort zu erkennen in welchem Dokument der

Akteur das erste mal auftaucht. Die Akteur ID ergibt sich aus den ersten sechs Zeichen einer Kennung, durch die der Akteur eindeutig bestimmt werden kann. Dies kann z.B. der Titel, der für das jeweilige Dokument zugrunde liegende Homepage oder Blogs, der Nick- bzw. Username des Autors oder seine Email-Adresse sein. Die zusätzlichen Dokument IDs geben die Identifikationsnummer der Dokumente an, in denen der Akteur noch zusätzlich auftaucht. Hierbei sollten die Dokumente, wenn möglich, bei der Kodierung in chronologischer Reihenfolge bearbeitet werden.

Die Vorteile der Akteur Syntax:

- Übersicht über die konkrete Zahl beteiligter Akteure und keine erhöhte durch Doppelungen,
- Übersicht über das Auftreten des jeweiligen Akteurs in Dokumenten und auch Episoden,
- Übersicht über zu einem Dokument gehörende Akteure anhand des vergebenen Namens,
- es ist sofort sichtbar wo der Akteur zum ersten Mal auftaucht,
- man vergibt bei Auftreten eines schon mal betrachteten Akteurs in einem neuen Dokument kein neues Teilkonzept, sondern erweitert nur den vorhandenen Namen.

Der einzige Nachteile ergibt sich aus der Beachtung der chronologischen Abarbeitung. Stellt sich nämlich im Nachhinein eine andere chronologische Reihenfolge der Dokumente heraus, so muss der vergebene Akteurname geändert werden.

Ein Beispiel eines solchen Akteurnamens ist „__P02__pishmi__P01__P03__P21“. Die erste Dokument ID deutet darauf hin, dass dieser Akteur zuerst in Dokument P02 aufgetreten ist. Zusätzlich besitzt er die Kennung „pishmi“ und kommt in drei weiteren Dokumenten als Autor vor. Die vorausgesetzte chronologische Bearbeitung der Dokumente führt zu der Aussage, dass der in diesem Beispiel betrachtete Akteur nach dem Dokument P02 in den Dokumenten P01, P03 und P21 nacheinander aufgetreten ist.

Der Teil <description> wird bei mir Verhaltenskonzept genannt und wie bei Salinger et al. [17] in folgender Form verwendet:

```
description = <verb>_<object>[_<criterion>]
```

Ich weise das Verhaltenskonzept direkt einem Texabschnitt zu, in dem das zu kodierende Phänomen auftritt. Da der Texteintrag, in dem sich der kodierte Textabschnitt befindet, einem <actor> Teilkonzept zugeordnet ist, wird das zugehörige Verhaltenskonzept automatisch diesem Akteur zugewiesen.

Der Teil <verb> steht in dritter Person Singular in deutscher Sprache und wird erweitert um sein Präfix und ggf. einer Präposition. Dieser Teil stellt eine Tätigkeit des jeweiligen

Akteurs dar. Das <object> ist ein Substantiv, welches den Gegenstand der Tätigkeit darstellt. Der optionale Teil <criterion> ist für Spezialisierungen und Darstellung von Ausprägungsmerkmalen, welche später die Dimensionen darstellen, gedacht. In diesem Teil findet auch die Negation eines Konzeptes statt.Denn durch die Darstellung der Ausprägungsmerkmale über den Teil <criterion>ist auch die duale Dimensionierung wie z.B. 0/1, ja/nein möglich. Dadurch werden keine unnötigen Erweiterungen oder Verkomplizierungen der Teile <verb> und <object> notwendig, sondern es ist möglich bestehende Konzeptnamen heranzuziehen und durch das <criterion> zu negieren.

Beispiele für Verhaltenskonzepte sind:

```
spezifiziert_Version (siehe Anhang A.1 ID 90)
bietet_Details_* (Gruppierung eines Verhaltenskonzepts siehe
Anhang A.1 IDs 37, 38 & 39)
beschreibt_Programmverhalten_ungewöhnliches (siehe Anhang A.1 ID 20)
```

Kodiere ich nun in einem Texteintrag, welcher dem Akteur aus dem obigem Beispiel (_P02_pishmi_ P01_P03_P21) zugeordnet ist, einen Textabschnitt, der das Phänomen Versionsspezifizierung beinhaltet, so ergibt sich als Konzeptname, bestehend aus Akteur und Verhaltenskonzept:

```
_P02_pishmi_P01_P03_P21.spezifiziert_Version
```

Dieses Konzept würde nun wie folgt umschrieben werden:„Akteur *pishmi* spezifiziert Version“. Die Angabe der Dokument IDs liefern weitere Aussagen, die ich im oberen Beispiel schon dargestellt habe.

Neben den Akteuren und Verhaltenskonzepten gibt es noch das Namensschema für die verschiedenen Rollenkonzepte:

```
Rolle: <Rollenkonzept>
```

Diese Konzepte leite ich aus den Verhaltenskonzepten ab (siehe Analyse-Modell im nächsten Abschnitt) und ordne sie direkt Akteuren zu. Das Beispiel „Rolle: Hersteller“ stellt das Rollenkonzept des Herstellers dar (mehr zu den Rollen im Abschnitt 3.2).

### 2.3.3 Analyse-Modell

Die dritte Praxis, welche Salinger et al. [17] vorschlagen, ist das Erstellen eines Analyse Meta-Modells. Dieses Modell soll die Struktur der Analyseergebnisse beschreiben und die Terminologie der verschiedenen Elemente des Ergebnis festlegen. In Anlehnung an das von Salinger et al. [17] erstellte Modell, in Form eines leicht abgewandelten UML Klassendiagramms[19], zeigt Abbildung 2.3 das von mir erstellte Analyse-Modell.

[19] UML Spezifikation 2.1.2 nach der Object Management Group http://www.omg.org/docs/formal/07-11-04.pdf überprüft am 14.08.2008

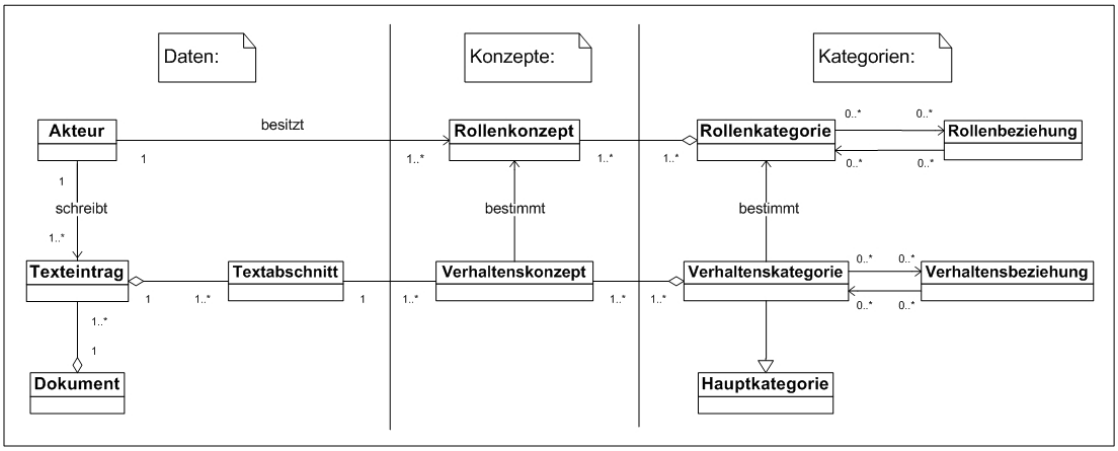


Abb. 2.3: Analyse Meta-Modell

Das Modell bettet sich in drei unterschiedliche Ebenen ein:

- Ebene der Daten,
- Ebene der Konzepte,
- Ebene der entwickelten Kategorien und ihrer Beziehungen.

Auf der Ebene der Daten liegen die Dokumente, welche die Daten für die Untersuchung bereitstellen. Innerhalb eines Dokumentes gibt es mindestens einen Texteintrag, der von genau einem Autor (Akteur) geschrieben wurde. Innerhalb eines Texteintrags gibt es beliebig viele Textabschnitte, in denen die zu betrachtenden Phänomene vorkommen. Die kleinste Menge, der in einem Texteintrag vorkommenden Textabschnitte, ist ein Textabschnitt der den gesamten Texteintrag umfasst.

Wird jetzt an einen Textabschnitt ein Verhaltenskonzept kodiert, findet der Wechsel auf die Konzeptebene statt. Zu einem Textabschnitt können mehrere Verhaltenskonzepte zugewiesen werden. Die Rollenkonzepte leiten sich aus den Verhaltenskonzepten ab. Diese Konzepte werden den zum Verhaltenskonzept gehörigen Akteur zugewiesen.

Ferner werden ähnliche bzw. zusammengehörige Verhaltenskonzepte auf der Kategorieebene zu Verhaltenskategorien gruppiert, aus denen analog zur Konzeptebene Rollenkategorien abgeleitet werden können. Rollenkategorien können aber auch aus Rollenkonzepten aggregiert werden. Die Verhaltenskategorien können in Beziehung untereinander stehen, ebenso wie die Rollenkategorien. Ich gehe auf diese Beziehungen speziell bei der Betrachtung des Phänomens „Umgang mit Schwachstellenveröffentlichung“ im Abschnitt 3.3 ein.

Eine Besonderheit auf der Kategorieebene besteht bei den Verhaltenskategorien. Diese werden zu Hauptkategorien gruppiert, zu denen dann alle Konzepte ihrer zugeordneten Verhaltenskategorien gehören. Auf diese Hauptkategorien gehe ich in meiner Verhaltensklassifikation im Abschnitt 3.1 näher ein.

## 2.4 Zusammenfassung

Am Anfang dieses Kapitels habe ich die Bestimmung möglicher Projekte vorgestellt. Dies geschah über die Schwachstellendatenbank der NVD, über Neuigkeiten zu sicherheitsrelevanten Themen aus dem IT-Bereich oder aufgrund persönlicher Vorlieben. Dabei habe ich die Projekte Drupal, Horde, Wordpress und Joomla identifiziert. Die ersten beiden Projekte habe ich verworfen, da sie zu wenig Daten für die Analyse bereitgestellt haben. Die Projekte Wordpress und Joomla habe ich für meine Datenanalyse ausgewählt und die konkrete Auswahl von Daten zu diesen Projekten beschrieben. Neben der Auswahl der Projekte habe ich auch die Eigenschaften der Daten, dass diese in Form von Textdokumenten vorliegen, beschrieben und den Begriff der Episode, die Sammlung aller Daten zu einer Schwachstelle, eingeführt.

Nach der Beschreibung der Datenerhebung habe ich die Forschungsmethode Grounded Theory vorgestellt. Mit Hilfe dieser Forschungsmethode habe ich meine Daten analysiert, weil sie Verfahren beinhaltet, die methodisches, wissenschaftliches und schrittweises Vorgehen zum systematischen Erheben und Analysieren von Daten beschreiben. Diese Verfahren sind offenes Kodieren, axiales Kodieren und selektives Kodieren.

Darüber hinaus habe ich Praktiken adaptiert, welche mich beim offenen Kodieren unterstützt haben. Diese Praktiken sind einen Blickwinkel auf die Daten festlegen, ein Konzeptnamensschema entwickeln, welches die Syntax der Konzeptnamen festlegt, und ein Analyse Meta-Modell erstellen, welches der Struktur der Analyseergebnisse beschreibt.

# 3 Analyse und Ergebnisse

In diesem Kapitel stelle ich im Abschnitt 3.1 das Vorgehen und das Ergebnis des offenen Kodierens in Form einer Klassifikation der Verhaltensweisen vor. Auf die Beschreibung der entwickelten Rollenklassifikation gehe ich in Abschnitt 3.2 ein. Anschließend stelle ich im Abschnitt 3.3 das Vorgehen und die Ergebnisse der Untersuchung zum Phänomen „Umgang mit Schwachstellenveröffentlichung“ vor. Im Abschnitt 3.4 gebe ich einen Ausblick zum Phänomen „Drängen nach Schwachstellenbehebung“, welches sich als weiterer Untersuchungsgegenstand herausgestellt hat, aber auf den ich nicht mehr genauer eingehen konnte. Abschließend stelle ich Praktiken zur Vermeidung von sicherheitsrelevanten Schwachstellen, welche ich im Laufe meiner Untersuchungen gefunden habe, im Abschnitt 3.5 vor.

## 3.1 Klassifikation der Verhaltenskonzepte und -kategorien

Ich habe die Klassifikation des Verhaltens (oder auch Verhaltensklassifikation), welche Verhaltensweisen, die ich im Verlauf der Behebung einer sicherheitsrelevanten Schwachstelle entdeckt habe, durch die Anwendung des offenen Kodierens entwickelt. Die Klassifikation soll in erster Linie einen Blick auf mögliche Verhaltenskonzepte und -kategorien des Behebungsverlaufs geben und dabei helfen diese Konzepte und Kategorien hierarchisch anzuordnen.

Ich habe für die Entwicklung der Konzepte, Kategorien und dem Aufbau der Klassifikation die Daten aller fünf Wordpress Episoden (siehe Anhang A.4.1) analysiert und die Konzepte aus der Klassifikation für die Kodierung eines Teils der Daten der Joomla Episode 1 (siehe Anhang A.4.2 Episode 1) herangezogen. Ich habe mich dafür entschieden das offene Kodieren nach der Analyse dieser Daten einzustellen, da ich zu diesem Zeitpunkt auf die beiden Phänomene „Umgang mit Schwachstellenveröffentlichung“ und „Drängen nach Schwachstellenbehebung“ aufmerksam geworden bin und diese noch genauer untersuchen wollte (siehe Abschnitt 3.3 und 3.4). Daher erhebe ich keinen Anspruch darauf, dass die Verhaltensklassifikation vollständig ist, d.h. alle möglichen Phänomene abdeckt sind, die innerhalb des Behebungsverlauf auftreten könnten. Durch offenes Kodieren weiterer Daten könnten neue Konzepte und Kategorien herausgearbeitet werden.

Einige Kategorien aus der Klassifikation wurden im späteren Verlauf meiner Arbeit ausgewählt und während des axialien Kodierens des Phänomens „Umgang mit Schwach-

stellenveröffentlichung“ genauer untersucht und in Beziehung zueinander gesetzt. Auf diesen Teil meiner Arbeit werde ich im Abschnitt 3.3 näher eingehen.

### 3.1.1 Die Konzeptebene

Auf Grundlage der im vorangegangenem Kapitel beschriebenen Praktiken habe ich die Episoden des Projekts Wordpress kodiert. Dabei habe ich nach dem Analyse-Modell die Verhaltenskonzepte an Textabschnitte kodiert und sie dem Namensschema entsprechend benannt. Zu Phänomenen des Verhaltens, welche zu Verhaltenskonzepten kodiert wurden, gehören die typische Aktivitäten beteiligter Akteure am Behebungsverlauf. Diese Aktivitäten können Einfluss auf den Behebungsverlauf und auf die Aktivitäten der anderen beteiligten Akteure haben.

Nach dem Kodieren der ersten Episoden ist die Liste der Verhaltenskonzepte auf 122 Konzepte angewachsen, wobei fast die Hälfte der Konzepte nur einen Textabschnitt kodieren. Um diese Menge zu reduzieren und mögliche redundant vergebene Konzepte zusammenzufassen, bin ich wie folgt vorgegangen:

1. Ich habe die Konzepte nach ihren Aktivitäten (vorangestellter Teil <verb>) sortiert.
2. Synonym zu sehende Aktivitäten habe ich zu einer Aktivität zusammengefasst, z. B. die Aktivitäten *Erklären* und *Schildern* zur Aktivität *Schildern* und die Aktivitäten *Angeben* und *Aufführen* zur Aktivität *Angeben* zusammengefasst.
3. Ich habe Phänomene aufgrund des zugrunde liegenden Phänomens oder des beinhalteten Objekts (mittlerer Teil <object>), auf welches die Aktivität zielt, umbenannt bzw. umkodiert.
   a) Umkodieren bedeutet, einen speziellen Texteintrag zu einem anderen bereits bestehenden oder neuem Konzept zuzuordnen, z. B. ist das Konzept bedanktSichBei_Berichterstatter(n) an der Textstelle (Quote in Atlas.ti) 2 im Dokument mit der ID 26 durch das Konzept bedanktSichBei_Hilfeleistendem ersetzt worden.
   b) Umbenennen bedeutet, einen neuen Namen für ein bestehende Konzept zu verwenden, z. B. das Konzept zeigtAuf_Missverständnis in stelltFest_ Missverständnis zu ändern.

Nach der Überarbeitung der Konzeptnamen reduzierte sich deren Anzahl von 122 auf 110. Die Zahl der in den Konzepten verwendeten unterschiedlichen Aktivitäten verringerte sich von 57 auf 39.

### 3.1.2 Verlauf der Behebung

Zu Beginn der Datenanalyse jeder Episode habe ich die Dokumente einer Episode in eine chronologische Abfolge gebracht. Dabei halfen mir die Veröffentlichungsdaten einzelner Texteinträge, wie das Datum des Blog-, Foren-, Schwachstellenveröffentlichungs- oder Mailinglisteneintrags aber auch die Änderungshistorie in Wikis, Bugtrackereinträgen und Versionsverwaltungssystemen. Weiterhin gaben auch Verweise auf andere Dokumente Hinweise zur chronologischen Abfolge.

Durch diese chronologische Ordnung der einzelnen Dokumente konnte ich den Ablauf des Behebungsverlaufs einzelner Episoden nachvollziehen und skizzieren. Während der Analyse der Daten und der Entwicklung der Verhaltenskonzepte habe ich Etappen des Behebungsverlaufs gefunden, welche in den untersuchten Episoden auftraten. Abbildung 3.1 zeigt die zeitliche Reihenfolge der gefundenen Etappen innerhalb des Behebungsverlaufs.

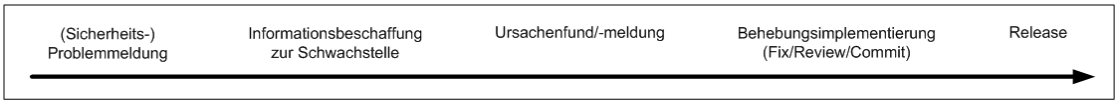


Abb. 3.1: Behebungsverlauf

Der Behebungsverlauf beginnt mit einer Meldung eines Akteurs zu einem Problem. Als Problem definiere ich hier das Verhalten der Anwendung, welches bei der Ausführung der Anwendung auftritt und nicht vom Benutzer erwartet wird. Je nach Informationsmenge und Beschreibung des Problems ist schon bei der Meldung ersichtlich, dass es sich um ein sicherheitsrelevantes Problem handelt. Ein sicherheitsrelevantes Problem entsteht durch das Ausnutzen einer Schwachstelle bzw. auch als unbeabsichtigter Nebeneffekt bei der Ausnutzung einer Schwachstelle.

Nachdem das Problem bekannt geworden ist, versuchen beteiligte Akteure mehr über die dem Problem zugrunde liegenden Schwachstelle herauszufinden. Es kommt schließlich zum Fund der Schwachstellenursache. Aufgrund dieser Information erfolgt die Behebungsimplementierung. Der Hersteller (Rollenkategorie „der Verantwortliche des Herstellers“ siehe Abschnitt 3.2) behebt den Defekt in der Software, welcher die Schwachstellenursache darstellt. Außerdem überprüft und bewertet er die Behebung und checkt sie anschließend in die Codebasis ein. Nachdem die Behebungsimplementierung abgeschlossen ist, erfolgt der offizielle Release mit der Behebung, d.h. die Veröffentlichung einer stabilen Version, welche die Schwachstelle nicht mehr beinhaltet.

Arbaugh et al. [1] und die Organization for Internet Safety [13] geben neben den Phasen Veröffentlichung ((Sicherheits-)Problemmeldung, Informationsbeschaffung zur Schwachstelle und Ursachenfund/-meldung) und Behebung (Behebungsimplementierung) noch die Entdeckung der Schwachstelle innerhalb der von ihnen beschriebenen Modellen an. Da

die Episoden den Behebungsverlauf von der Meldung bis zum Release abdecken, entfällt diese Phase bei meiner Betrachtung des Behebungsverlaufs.

Bei einigen Episoden habe ich Abwandlungen des oben beschriebenen Verlaufs festgestellt:

- Ein Akteur behob einen Defekt in der Software und veröffentlichte die Behebung (Behebungsimplementierung). Aus den durchgeführten Änderungen im Code rekonstruiert ein zweiter Akteur eine Schwachstelle (Informationsbeschaffung zur Schwachstelle) und veröffentlichte sie (Ursachenfund/-meldung). Aufgrund dieser Schwachstellenveröffentlichung gab der Hersteller den Release mit der Behebung raus.
- Auf eine Schwachstellenveröffentlichung ((Sicherheits-)Problemmeldung, Informationsbeschaffung zur Schwachstelle und Ursachenfund/-meldung) folgten die Behebungsimplementierung und der Release.
- Der Behebungsverlauf begann mit einer Schwachstellenmeldung an eine Securitymailingliste bzw. an ein Securityteam. Es findet eine interne Behebungsimplementierung statt, welcher ein Release mit der zugehörigen Schwachstellenveröffentlichung folgte.

Aufgrund dieser drei Varianten kam ich zu der Erkenntnis, dass es die oben beschriebenen Etappen in allen Varianten auftauchen bzw. zu einer neuen Etappe zusammengefasst werden können. Nur die Reihenfolge, in der die Etappen innerhalb des Behebungsverlaufs auftreten, verändert sich.

Die ersten beiden Varianten beinhalten eine explizite Schwachstellenveröffentlichung innerhalb des Verlaufs der Behebung. Dies ist einer der Gründe warum ich das Phänomen „Umgang mit Schwachstellenveröffentlichung“ ausführlich untersucht habe. Auf diese Untersuchung gehe ich im Abschnitt 3.3 genauer ein.

### 3.1.3 Die Verhaltenskategorien

Nachdem ich die Menge der Verhaltenskonzepte auf eine übersichtlichere Zahl verkleinert und zu einer übersichtlicheren Menge an Konzeptaktivitäten reduziert habe, begann ich mit der Bildung der Verhaltenskategorien, also die Gruppierung verschiedener Konzepte, welche zu einem Phänomen gehören. Bei der Bildung der einzelnen Kategorien habe ich eine der drei folgenden möglichen Bildungsmöglichkeiten oder einer Vermischung dieser angewandt.

Die Kategorien entstanden:

1. aufgrund des im Konzeptnamen auftretenden Objekts (Teil <object> des Konzeptnamens).

2. aufgrund der im Konzeptnamen auftretenden Aktivität (Teil <verb> des Konzeptnamens).
3. aufgrund des Auftretens in gleichen Etappen des Behebungsverlaufs.

In der ersten Möglichkeit habe ich Konzepte, welche denselben Objektnamen haben, zu einer Kategorie zusammengefasst. So habe ich z.B. die verschiedenen Konzepte mit dem Objekt *Ursache*, wie unter anderem „begründet_Ursache“ (siehe Anhang A.1 ID 9) und „vermutet_Ursache_weitere“ (Anhang A.1 ID 103), in die Verhaltenskategorie *Ursachenidentifikation* eingeordnet.

Auch Konzepte deren Objekte als zusammengehörig oder synonym gesehen werden können, habe ich in einer Kategorie gruppiert, wie z.B. *Hilfe* („erhofftSich_Hilfe“, Anhang A.1 ID 50) und *Hilfeleistung* („bittetUm_Hilfeleistung“, Anhang A.1 ID 41) zur Kategorie *Unterstützung erbeten.*

Neben der Zusammengehörigkeit der Objekte ist auch der gemeinsame Kontext, in denen Objekte verschiedener Konzepte auftreten können, für die Bildung von Kategorien möglich, da ein gemeinsamer Kontext auch auf ähnliche Phänomene hindeutet. Beispielsweise habe ich die Konzepte „beantwortet_Verständnisfrage“ (Anhang A.1 ID 2) und „fragtNach_Verständnisklärung“ (Anhang A.1 ID 61) zur Verhaltenskategorie *Verständnisklärung*, welche auch den Kontext beider Konzepte bildet, zusammengefasst.

Die zweite Möglichkeit zur Kategorienbildung lässt sich analog zur Objektgruppierung anwenden und stellt eine Verfeinerung der im vorangegangenem Abschnitt beschriebenen, durch Aktivitäten erzeugte Konzeptgruppierung dar. Dabei wurden Konzepte, deren Aktivitäten gleich sind oder ähnliche Bedeutung haben, zu Kategorien zusammengefasst. Beispiele für solche Aktivitäten sind *Bestätigung*, wie in den Konzepten „bestätigt_Problem“ (Anhang A.1 ID 25) und „bestätigt_Schwachstelle“ (Anhang A.1 ID 28), welche zur Kategorie *Informationsbestätigung* zusammengefasst wurden, oder *Bewertung,* bei der u.a. die Konzepte „bewertet_Schadensbegrenzung“ (Anhang A.1 ID 33) und „bewertet_Schweregrad“ (Anhang A.1 ID 35) in die Kategorie *Informationsbewertung* eingeordnet wurden.

Die dritte Bildungsmöglichkeit lehnt sich an den Behebungsverlauf an. Dabei habe ich Konzepte zu Kategorien zusammengefasst, die in einer bestimmten Etappe innerhalb des Verlaufs der Behebung auftreten. Ein Beispiel dafür ist die Kategorie *Review*, welche Konzepte wie „bewertet_Lösung_positiv“ (Anhang A.1 ID 31) „gibtAn_Verbesserungsvorschlag“ (Anhang A.1 ID 71) zusammenfasst, die während des Reviews in der Etappe Behebungsimplementierung auftreten.

Diese drei Möglichkeiten habe ich nicht ausschließlich getrennt von aneinander angewandt, sondern für die Bildung einiger Verhaltenskategorien vermischt. Ein Beispiel für diese Art der Kategorienbildung ist die *Schwachstellendarstellung*, die einerseits Konzepte gruppiert, deren Objekt die *Schwachstelle* selbst ist oder damit im Zusammenhang

steht, wie *Schwachstellenvoraussetzung*. Andererseits beinhaltet diese Kategorie Konzepte, welche im Behebungsverlauf, während der Etappe Schwachstellenveröffentlichung (Zusammenfassung der Etappen (Sicherheits-)Problemmeldung, Informationsbeschaffung zur Schwachstelle und Ursachenfund/-meldung), auftreten.

Im folgenden Teil dieses Abschnitts werde ich die entwickelten Verhaltenskategorien vorstellen und sie kurz beschreiben. Eine vollständige Übersicht über die Verhaltenskategorien und der ihnen zugeordneten Konzepte habe ich im Anhang A.2 angefügt.

**Funktionsweisendarstellung** umfasst die Verhaltenskonzepte der detaillierten Beschreibung von Funktionen der Software bzw. Änderungen an der Software, durch welche die Schwachstellen behoben werden soll. Dabei kann die Quellcode innerhalb der Beschreibung mit einbezogen werden.

**Herstellerinformationenbereitstellung** ist die Kategorie für die Verhaltenskonzepte, welche die Bereitstellung von Informationen über oder vom Hersteller der betrachteten OSS umfassen.

**Problemdarstellung** gruppiert alle Aspekte, die Informationen zu einem Problem liefern.

**Schwachstellendarstellung** stellt eine umfangreiche Kategorie dar. In ihr werden alle Eigenschaften zusammengefasst, welche innerhalb eines Schwachstellenberichts bzw. einer Schwachstellenveröffentlichung oder in deren Kontext auftreten.

**Zwischenstatusbereitstellung** beschreibt das Phänomen der Frage und Angabe eines Zwischenstatus.

**Informationsbestätigung** umfasst das Bestätigen oder das Ablehnen von gegebenen Informationen.

**Informationsbewertung** beinhaltet die Bewertung und Einschätzung von Informationen und die Anfrage, welche zu diesen Bewertungen führt.

**Informationskorrektur** stellt die Verhaltensweisen zur Richtigstellung und Verbesserung von gegebenen Informationen dar.

**Review** ist eine der Behebungsimplemtierung zugeordnete Kategorie. Diese Kategorie beinhaltet Aktivitäten die zur Begutachtung des Quellcodes gehören, nach dem die Schwachstelle behoben wurde.

**Ursachenidentifikation** sind die Verhaltensweisen, welche die Ursache für ein Problem oder eine Schwachstelle suchen und bewerten.

**Verständnisklärung** umfasst das Stellen und Beantworten von Verständnisfragen.

**Drängen nach Lösung** ist die Kategorie, welche das Drängen durch Akteure beinhaltet. Diese Aktivität kann allgemein formuliert oder gezielt auf bestimmte Akteure gerichtet sein.

**Missverständnisbehandlung** beinhaltet die Verhaltenskonzepte, welche das Auftreten und Klären von Missverständnissen behandeln.

**Umgang mit Beleidigungen** umfasst Verhaltensweisen zum Umgang mit Beleidigungen und unsachlicher Diskussionsführung.

**Umgang mit voller Schwachstellenveröffentlichung** stellt die Kategorie dar, welche alle Aktionen und Reaktionen auf eine Schwachstellenveröffentlichung von Akteuren beinhaltet.

**Empfehlung** beschreibt das geben und versprechen der Einhaltung und Durchführung von Empfehlungen.

**Motivation** beinhaltet das Begründen, Bestätigen und Vermuten von Motivation der Akteure, welche zur Schwachstellenbehebung führt.

**Unterstützung erbeten** beschreibt Verhaltensweisen, mit welchen Hilfe, Unterstützung oder Informationen erbeten werden.

**Unterstützung geben** umfasst Verhaltensweisen, welche Informationen, Unterstützung und Hilfe liefern.

**konkreten Schutz liefern** beinhaltet als Kategorie die Angabe bzw. Bereitstellung von Maßnahmen durch Akteure, welche Sicherheit gewährleisten oder wiederherstellen sollen.

**Lösungs_Informationsbereitstellung** stellt die Verhaltensweisen zur Erfragung und Angabe einer Lösung sowie der Auflistung der Lösungsversuche dar.

**Bugtracker** umfasst die Konzepte, die spezifisch bei der Verwendung eines Bugtrackers auftreten.

**Kommunikationswerkzeug** gruppiert die Konzepte, die beim Umgang mit Foren, Mailinglisten, Chats und weiteren Kommunikationswerkzeugen auftreten.

**Versionsverwaltung** beinhaltet die Verhaltensweisen, die konkret bei der Verwendung einer Versionsverwaltung auftreten.

### 3.1.4 Die Hauptkategorien

Nach der Verhaltenskategoriebildung habe ich diese zu Hauptkategorien zusammengefasst. Zum einen liefern die Hauptkategorien einen speziellen Blick auf die Daten und die darin beherbergten Phänomene. Zum anderen ermöglichen sie eine weitere Abgrenzung der Verhaltenskategorien voneinander. Dadurch bieten sie eine bessere Übersicht bei der Einordnung neu entwickelter Konzepte oder gruppieren Verhaltenskategorien, die ich während des axialen Kodierens genauer betrachtet habe (siehe Beschreibung zu *Umgang mit Konflikten* unten).

Bei der Betrachtung der Verhaltenskategorien habe ich verschiedene Gemeinsamkeiten zwischen ihnen gefunden, welche die Grundlage für die Entwicklung der folgenden Hauptkategorien darstellen[1]. Die Beschreibung dieser Gemeinsamkeiten stelle ich im Anschluss an die Nennung der Hauptkategorien vor.

- Informationsbereitstellung
- Informationsverwertung/-verarbeitung
- Einleitung konkreter Maßnahmen
- Umgang mit Konflikten
- Unterstützung
- Einsatz von Werkzeugen

**Die Informationsbereitstellung** fasst die Kategorien zusammen, welche die Aktionen der Informationslieferung beinhalten. Diese Informationen können ein konkretes Phänomen beschreiben (Problem, Schwachstelle etc.), von einer bestimmten Rolle (Hersteller etc.) geliefert werden oder zu einem bestimmten Zeitpunkt auftreten (Zwischenstatus etc.). Des weiteren liegen die Informationen in verschiedenen Formen vor, z.B. als einzelne Datenstichpunkte wie Datum oder Versionsnummer, als Zusammenfassung verschiedener Datenstichpunkte in Form eines Berichts oder als ausführliche Schilderung. Bei einer Schilderung fließen Begründungen, Beschreibungen, aber auch Gedanken und Vermutungen in den Bericht mit ein.

**Die Informationsverwertung/-verarbeitung** umfasst die Kategorien, in welchen die bereitgestellten Informationen für die Behebung der Schwachstelle angenommen und verarbeitet werden. Dabei können die Informationen bewertet, bestätigt, korrigiert, hinterfragt oder/und vervollständigt werden. Als Ergebnis der Verarbeitung ergeben sich neue Informationen.

**Die Kategorie Einleitung konkreter Maßnahmen** beinhaltet die Kategorien, welche aufgrund ihrer Aktionen zur Sicherheit im Allgemeinen und speziell zur Behebung der Schwachstelle beitragen. Diese Aktionen äußern sich in Maßnahmen und ergeben sich aus den verarbeiteten Informationen. Diese unterteilen sich in technische und informative Maßnahmen. Neben dem Ziel der Behebung der Schwachstelle kann der Einsatz konkreter Maßnahmen auch Zwischenlösungen vorsehen. Diese Zwischenlösungen werden unter anderem durch die Maßnahmen Symptombekämpfung, Workarounds, Schadensbegrenzung oder Vermeidungsregeln erreicht.

**Der Umgang mit Konflikten** gruppiert die Kategorien, deren zugrunde liegenden Phänomene sich zum größten Teil als interessante Gegenstände für nähere Untersuchung

[1] Eine Zuordnung der Verhaltenskategorien zu den entsprechenden Hauptkategorien habe ich im Anhang A.2 eingefügt.

erwiesen haben. Diese Konflikte können während des Verlaufs der Behebung auftreten. Sie beruhen meist auf zwischenmenschlicher Kommunikation und haben einen Einfluss auf die Behebung der Schwachstelle und auf anderen Kategorien. Die Verhaltenskategorie *Umgang mit voller Schwachstellenveröffentlichung*, welche das Phänomen „Umgang mit Schwachstellenveröffentlichung“ repräsentiert, ist dieser Hauptkategorie zugeordnet. Dieses Phänomen habe ich im späteren Verlauf der Arbeit untersucht und ich beschreibe es im Abschnitt 3.3.

**Unter Unterstützung** sind die Kategorien zusammengefasst, welche die sozialen Aspekte darstellen, die zur Behebung der Schwachstelle beitragen. Darunter fallen das *Geben und Nehmen* zwischen den Akteuren und die Motivation, welche dazu beiträgt. Diese Aspekte haben Einfluss auf den Behebungsverlauf und die anderen Kategorien, wobei der Grad des Einflusses von der Intensität der Unterstützung abhängig ist.

**Der Einsatz von Werkzeugen** umfasst die Kategorien, deren zugrunde liegenden Konzepte bei der Verwendung von Werkzeugen, die im Verlauf der Behebung eingesetzt werden, auftreten. Beispiele für diese Werkzeuge sind Bugtracker, Foren, Mailinglisten, Chats, Wikis und Versionsverwaltungen. Diese Hauptkategorie dient dazu, die Konzepte und Kategorien einzusortieren, die übergreifend im Kontext der anderen Hauptkategorien und bei der Benutzung von Werkzeugen auftreten. Darüber hinaus stellt diese Hauptkategorie den Einsatz von Werkzeugen während des Behebungsverlaufs dar.

## 3.2 Klassifikation der Rollen

Neben den Verhaltensweisen habe ich auch die Rollen, welche die Akteure während des Behebungsverlauf annehmen, untersucht. Im Prozess des offenen Kodierens habe ich die Rollenkonzepte aufgrund der Verhaltenskonzepte des Akteurs, Hinweise im Text und aus Metadaten zu dem Projekt entwickelt. Metadaten kommen in Datenquellen vor, welche zu keiner Episode gehören, sondern Informationen zu bestimmten Akteuren, dem Behebungsverlauf im Allgemeinen oder zum Projekt liefern. Folgende Rollenkonzepte sind dabei herausgekommen[2]:

- Rolle: Beitragender
- Rolle: Dokumentator
- Rolle: Ehemaliger
- Rolle: Externer
- Rolle: Forumsmoderator

[2]Eine Beschreibung dieser Rollenkonzepte ist im Anhang A.3 zu finden.

- Rolle: Hilfeleistender
- Rolle: Interner
- Rolle: Kernteammitglied
- Rolle: OSS Benutzer
- Rolle: Problem-Berichterstatter
- Rolle: Problemlöser
- Rolle: Releaseankündiger
- Rolle: Schwachstellen-Berichterstatter
- Rolle: Supporter

Aus Verhaltenskonzepten entstandene Rollen sind z.B. *Problem-Berichterstatter* aus dem Konzept *berichtet_Problem* (siehe Anhang A.1 ID 15) und *Releaseankündiger* aus dem Konzept *kündigtAn_ Release* (siehe Anhang A.1 ID 77). Hinweise im Text, welche Rollenkonzepte ermöglichten, sind z.B. für das Rollenkonzept *OSS Benutzer* „Damn, I am having same problem. Just realized this in my source code“, „I thing my site got one also [..]“ oder „Yup. The same thing happened to me. Running 2.3.“ (alle Textabschnitte gehören zum Verhaltenskonzept *bestätigt_Problem* Anhang A.1 ID 25). Die Rollenkonzepte *Ehemaliger, Supporter, Dokumentator, Forumsmoderator, Kernteammitglied* sind aus den Metadaten zum Projekt entstanden. Dabei sind die Konzepte *Supporter* und *Kernteammitglied* In-vivo-Codes, d.h. der Name für das Konzept kommt in dieser Form in den Daten vor und bezeichnet die Rolle sehr passend.

Die Rollenkonzepte *Externer* und *Interner* stellen keine eigenen Konzepte dar, sondern dienen als Eigenschaften für andere Rollenkonzepte bzw. -kategorien. Diese beiden Rollenkonzepte habe ich ebenso aus den Metadaten der jeweiligen Projekte entwickelt. Denn innerhalb dieser Daten, werden die Akteure aufgeführt, die zum offiziellen Team des Projekts gehören (Interne). Alle anderen am Behebungsverlauf beteiligten Akteure sind *Externe*.

### 3.2.1 Rollenkategorien

Während des offenen Kodierens habe ich entdeckt, dass einem Akteur während des Behebungsverlaufs verschiedene Rollenkonzepte zugeordnet werden können. In Abbildung 3.2 auf der nächsten Seite ist ein Beispiel einer zeitlichen Veränderung der Rollenzuordnung eines Akteurs dargestellt. So kann ein Akteur, welcher zu Beginn des Behebungsverlaufs als *Problem-Berichterstatter* agiert, durch genügend technisches Grundwissen die Ursache des Problems herausfinden, diese als Schwachstelle offenbaren und gleichzeitig eine Lösung zu dem Problem bzw. der Schwachstelle anbieten.

Abb. 3.2: Veränderung der Rollenzuordnung über Zeit

Für die weitere Untersuchung habe ich drei Rollenkategorien entwickelt, welche ihre Zuordnung zu einem Akteur auch bei einer Veränderung der Konzeptzuordnung innerhalb des Behebungsverlaufs beibehält.

1. der betroffene/besorgte Nutzer,
2. der externe Helfer,
3. der Verantwortliche des Herstellers.

Neben der Gruppierung von Rollenkonzepten besitzen diese Rollenkategorien entweder die Eigenschaft *Externer* oder *Interner* und stehen mit bestimmten Verhaltenskategorien in Beziehung. Die Beziehung entsteht über die Verbindung zwischen den Verhaltenskonzepten, welche im Texteintrag des Akteurs und seiner zugehörigen Rollenkategorie kodiert wurden, und ihrer zugehörigen Verhaltenskategorie.

Die erste Rollenkategorie *der betroffene/besorgte Nutzer* ist ein externer Akteur, welcher durch die Verhaltenskategorien *Unterstützung erbeten*, *Problemdarstellung* oder *Drängen nach Lösung* erkannt werden kann. Die Kategorie umfasst Rollenkonzepte wie *Problem-Berichterstatter*, *OSS Benutzer.*

Die zweite Rollenkategorie *der externe Helfer* ist ebenfalls ein externer Akteur. Typische Verhaltenskategorien, welche diese Rollenkategorie auszeichnen sind *Unterstützung geben, konkreten Schutz liefern, Umgang mit voller Schwachstellenveröffentlichung, Schwachstellendarstellung, Drängen nach Lösung.* Beispiele für zugehörige Rollenkonzepte sind *Supporter, Ehemaliger, Hilfeleistender, Problemlöser.*

Die letzte Rollenkategorie ist *der Verantwortliche des Herstellers*, welcher die Eigenschaft Intern besitzt. Diese Kategorie ist mit den Verhaltenskategorien *Unterstützung geben, konkreten Schutz liefern, Umgang mit voller Schwachstellenveröffentlichung, Herstellerinformationenbereitstellung, Lösungs_Informationsbereitstellung, Review* und *Funktionsweisendarstellung* verknüpft. Rollenkonzepte, die zu dieser Kategorie gehören, sind z.B. *Kernteamentwickler, Releaseankündiger, Problemlöser.*

Das letzte Rollenkonzept wird auch der Kategorie *der externe Helfer* zugeordnet. Bei den Rollenkategorien habe ich auf eine strenge Abgrenzung der untergeordneten Konzepte verzichtet, da zum einen die Kategorieeigenschaften *Externer*, *Interner* und zum anderen die dem Akteur zugeordneten Verhaltenskategorien eine Abgrenzung bieten. Letzteres

wird z.B. anhand der Haupkategorie Unterstützung deutlich. *Der betroffene/besorgte Nutzer* erbittet Hilfe während *der externe Helfer* diese anbietet.

Eine komplette Übersicht über die Beziehungen der Rollenkategorien zu ihren Rollenkonzepten und Verhaltenskategorien sind im Anhang A.3.2 zu finden. Die hier vorgestellten Rollenkategorien werden bei der Untersuchung des Phänomens Umgang mit Schwachstellenveröffentlichung im nächsten Abschnitt eingebunden.

## 3.3 Phänomen „Umgang mit Schwachstellenveröffentlichung“

In diesem Abschnitt stelle ich das Phänomen „Umgang mit Schwachstellenveröffentlichung“ und dessen Auswirkungen auf den Verlauf der Schwachstellenbehebung dar. Zuerst erkläre ich im Abschnitt 3.3.1 den Begriff der Schwachstellenveröffentlichung. Danach stelle ich im Abschnitt 3.3.2 Gründe vor, warum ich mich für die Untersuchung dieses Phänomens entschieden habe und welche Erkenntnisse ich zu diesem Phänomen in der Literatur gefunden habe. Im Abschnitt 3.3.3 beschreibe ich das Ziel, welches ich durch die Untersuchung des Phänomens erreichen will, und die mit Hilfe des paradigmatischen Modells in Beziehung gesetzten Kategorien. Die Analyse und die dabei gewonnenen Erkenntnisse über das Phänomen stelle ich im Abschnitt 3.3.4 vor. Im Abschnitt 3.3.5 fasse ich alle Aussagen, welche ich bei der Analyse der Daten zu dem Phänomen „Umgang mit Schwachstellenveröffentlichung“ gefunden habe, zusammen.

### 3.3.1 Begriff Schwachstellenveröffentlichung

Die Schwachstellenveröffentlichung (engl. Disclosure) beinhaltet die Veröffentlichung aller Schwachstellendetails. Darunter fallen folgende Informationen, welche aber nicht zwingend alle bei Veröffentlichung angegeben werden:

- eine generelle Beschreibung der Schwachstelle [*beschreibt_ Schwachstelle* Anhang A.1 ID 22],
- Ursache der Schwachstelle im Code [*schildert_ Schwachstellenursache* Anhang A.1 ID 86],
- die betroffenen Versionen der Software [*spezifiziert_ Version* Anhang A.1 ID 90],
- Darstellung des Sicherheitsrisikos [*bewertet_ Schweregrad* Anhang A.1 ID 35],
- Folgen der Schwachstellenausnutzung [*gibtAn_Schwachstellenfolge(n)_mögliche* Anhang A.1 ID 69],
- Beweis der Schwachstelle (engl. Proof of Concept) [*beschreibt_ Schwachstellenausnutzung* Anhang A.1 ID 23],

- explizite Beispiele der Schwachstellenausnutzung (engl. Exploit) [*stelltBereit_ Exploit* Anhang A.1 ID 94],
- Voraussetzung bzw. Einschränkungen zur Schwachstellenausnutzung [*grenztEin_ Schwachstellenvoraussetzung* Anhang A.1 ID 74],
- Angabe der Wiederherstellung bzw. Gewährleistung der Sicherheit [*gibtAn_ Workaround* Anhang A.1 ID 72, *stelltBereit_Patch* Anhang A.1 ID 95].

Ich habe in eckigen Klammern die Verhaltenskonzepte hinzugefügt, durch welche der Akteur die jeweilige Information der Schwachstellenveröffentlichung bereitstellt bzw. näher darstellt.

Es gibt verschiedene Arten der Veröffentlichung einer Schwachstelle. Die volle Schwachstellenveröffentlichung (engl. full disclosure) ist die Veröffentlichung so vieler Informationen wie möglich über die sicherheitsrelevante Schwachstelle, inklusive der Darstellung der Schwachstellenausnutzung. Diese Art der Veröffentlichung soll nach Ford und Gordon [6] dem Benutzer des Systems dabei helfen, richtige Entscheidungen auf Grundlage dieser Informationen zu treffen und angemessene Aktionen einzuleiten.

Im Gegensatz zur vollen Veröffentlichung unterlässt die partielle Veröffentlichung (engl. partial disclosure) nach Arora et al. [2] die Angabe von expliziten Informationen und technischen Details zur Schwachstellenausnutzung.

Eine weitere Unterscheidung der verschiedenen Veröffentlichungsarten betrifft den Veröffentlichungszeitpunkt der Schwachstelle. Ozment [15] unterscheidet dabei zwischen der unmittelbaren Veröffentlichung (engl. instantaneous disclosure), bei welcher die Veröffentlichung vor dem Release der Behebung durch den Hersteller erfolgt, und der verantwortlichen Veröffentlichung (engl. responsible disclosure), bei welcher die Schwachstellenveröffentlichung bis nach der Behebung aufgeschoben wird.

### 3.3.2 Festlegung des Untersuchungsgegenstands

Die volle Veröffentlichung von Schwachstellendetails taucht in meinen Daten meist gefolgt von einer kontrovers geführte Diskussion zu diesem Thema auf. Bei der Entwicklung meiner Verhaltensklassifikation bin ich in vielen Episoden auf dieses Phänomen und die dazugehörige Diskussion gestoßen. Deswegen habe ich die zu diesem Phänomen gehörenden Verhaltensweisen als Verhaltenskategorie „Umgang mit voller Schwachstellenveröffentlichung“ unter der Hauptkategorie „Umgang mit Konflikten“ eingeordnet (vgl. Anhang A.2). Wie im Abschnitt 3.1.4 beschrieben haben alle zur Hauptkategorie „Umgang mit Konflikten“ gehörenden Verhaltenskategorien Einfluss auf anderen Kategorien und damit auch auf den Behebungsverlauf, was mich darin bestärkte das Phänomen „Umgang mit Schwachstellenveröffentlichung“ als Untersuchungsgegenstand festzulegen.

### Literatur zum Phänomen

Neben meinen Beobachtungen dieses Untersuchungsgegenstands in den Daten ist dieser auch in der Literatur Gegenstand der Betrachtung. Dabei wird zum einen die richtige Vorgehensweise bei der Veröffentlichung von Schwachstellen diskutiert. Zum anderen finden sich in der Literatur Erkenntnisse zu den Auswirkungen der Schwachstellenveröffentlichung.

Levy [11] zeigt die Argumente beider Seiten, welche sich für und gegen die volle Veröffentlichung von sicherheitsrelevanten Schwachstellen aussprechen. Das wichtigste Argument für diese Art der Veröffentlichung ist die schnelle Behebung der Schwachstelle und die Bereitstellung der Informationen zur Schwachstelle für jeden. Das wichtigste Gegenargument äußert sich in der Aussage, dass die volle Schwachstellenveröffentlichung zu Angriffen und Schwachstellenausnutzung führt, da sie potentiellen Angreifern Informationen zur Schwachstelle bereitstellt.

Levy [11] stellt außerdem eine Alternative zur vollen Veröffentlichung in Form eines geschlossenen Konsortiums dar, innerhalb dessen die Schwachstellendetails berichtet und geteilt werden. Dies soll verhindern, dass die Schwachstellendetails potentiellen Angreifern in die Hände fallen. Open Source Software nimmt bei seiner Betrachtung eine Sonderrolle ein, da Schwachstellendetails nach einem Patch der Software in den Codeänderungen sichtbar sind. Somit müsste Open Source Software aus der geschlossenen Gruppe verbannt werden, da sie Informationen nach Außen zulässt.

Canellopoulou-Bottis [3] fasst für beide Seiten noch weitere Argumente zusammen. Durch die Veröffentlichung von Schwachstellen erhalten Forscher Informationen, welche dabei helfen können die Sicherheit durch ihre Forschung zu verbessern. Außerdem sind Informationen über geheimgehaltene Schwachstellen meist nur für den Angreifer von Nutzen.

Als weitere Gegenargumente zur Veröffentlichung von Schwachstellen nennt Canellopoulou-Bottis [3] zum einen, dass der Großteil der Nutzer einer Software auch bei Veröffentlichung einer Schwachstelle und eines Patches zur Behebung nichts zu ihrer Sicherheit unternehmen würden. Zum anderen führt der Druck durch eine Veröffentlichung zu geringerer Qualität in der Software.

Arora et al. [2] testeten empirisch den Einfluss der Schwachstellenveröffentlichung und die dabei vorhandene Verfügbarkeit von Patches. Diese Untersuchung unterteilt sich in zwei Bereiche. Zum einen wird der Einfluss der Veröffentlichung auf das Angriffsverhalten untersucht. Zum anderen wird die Auswirkung der Veröffentlichung einer Schwachstelle auf das Behebungsverhalten untersucht.

Im ersten Fall wurden Daten zu Angriffen auf honeypots[3], Systeme mit verschiedenen Betriebssystemen, aber ohne reale Anwendungen und mit einer Anbindung ans Internet,

[3] http://www.honeynet.org/ überprüft am 15.08.2008

extrahiert und diesen Daten zu Schwachstellen, welche sie aus Schwachstellendatenbanken erhoben haben, zugeordnet. Über diese Zuordnung konnten sie Aussagen über Zusammenhänge zwischen der Veröffentlichung einer Schwachstelle, der Bereitstellung eines Behebungspatches und Angriffen, welche diese Schwachstelle ausnutzten, treffen. Dabei ergaben sich folgende Erkenntnisse:

- eine Veröffentlichung erhöht die Zahl der Angriffe auf diese Schwachstelle,
- gibt es eine Veröffentlichung ohne eine gleichzeitige Bereitstellung einer Behebung ist die Zahl der Angriffe noch höher,
- je älter eine Schwachstelle wird, desto weniger oft wird sie ausgenutzt im Gegensatz zu aktuelleren Veröffentlichungen,
- die Patchbereitstellung, welche eine bereits veröffentlichte Schwachstelle behebt, führt zu weniger Angriffen,
- die Patchveröffentlichung einer geheimen Schwachstelle erhöht die Zahl der Angriffe.

Über die Auswirkungen und dem tatsächlichen Schaden der Angriffe konnten Arora et al. [2] keine Aussagen treffen. Sie konnten nur die Zahl der Angriffe zu der Veröffentlichung in Beziehung setzen.

Im zweiten Bereich ihrer Untersuchung verglichen sie den Zeitpunkt von Schwachstellenveröffentlichungen in Schwachstellendatenbanken mit dem Zeitpunkt der Patchbereitstellung durch den Hersteller. Dabei zeigten sie, dass OSS Hersteller veröffentlichte Schwachstellen schneller beheben. Größere Hersteller reagieren intensiver auf Veröffentlichungen. Ebenso wird auf schwerwiegendere Schwachstellen intensiver reagiert. Und eine unmittelbare und volle Veröffentlichung erhöht die Behebungswahrscheinlichkeit und -bereitschaft des Herstellers.

In ihrer Beschreibung des Schwachstellenlebenszyklus zeigen Arbaugh et al. [1], dass nicht die Veröffentlichung die eigentliche Ursache für die vielen Angriffe ist, sondern die Automatisierung der Angriffe, welche durch die Veröffentlichung ermöglicht wird.

### 3.3.3 Ziel und Vorgehen bei der Phänomenuntersuchung

Für meine Arbeit sind die Aussagen aus der Literatur über die Auswirkungen einer Schwachstellenveröffentlichung interessant, wobei dort vor allem auf die endgültigen Auswirkungen eingegangen wurde. Es fehlt die detaillierte Betrachtungen des Phänomens „Umgang mit Schwachstellenveröffentlichung“, wie z.B. die Beschreibung genauerer Umstände und Bedingungen (Kontext), die Art der Veröffentlichung oder welche Art von Akteuren vorrangig involviert sind.

Darüber hinaus betrachten die Autoren, der oben dargestellten Quellen, Open Source Software nur am Rande. Sie beschreiben mögliche Folgen von geschlossenen Gruppen für

die Schwachstellenveröffentlichungen im Zusammenhang mit OSS. Und sie formulieren die Aussage, dass bei Open Source Software die Behebungsdauer nach einer Veröffentlichung kürzer ist als bei Closed Source Software, wobei die Gründe dafür offen bleiben. Außerdem betrachten sie den Nutzer der Software nicht als möglichen Problemlöser, Hilfeleistenden oder Beheber von Schwachstellen, obwohl er in der Open Source Softwareentwicklung das alles sein kann.

Aus dieser Betrachtung heraus ergibt sich für meine Untersuchung des Phänomens „Umgang mit Schwachstellenveröffentlichung“ folgendes Ziel:

> Muster in meinen Daten zu finden, welche den Einfluss auf die Behebungsdauer, den Behebungsverlauf und den Einfluss der verschiedenen Rollen beschreiben.

Um dieses Ziel zu erreichen bin ich in Anlehnung an das axiale Kodieren wie folgt vorgegangen:

1. Hypothetisches In-Beziehung-Setzen von Verhaltenskategorien zu der Kategorie, welche das zu untersuchende Phänomen repräsentiert, mit Hilfe des in Abschnitt 2.2.2 vorgestellten paradigmatischen Modells. Diese in Beziehung zueinander gesetzten Kategorien beschreibe ich unten.
2. Verifikation dieser Beziehung anhand der Daten und die Suche nach weiteren Eigenschaften der in Beziehung gesetzten Kategorien. Diese Kategorien, ihre Eigenschaften und dimensionale Bandbreite (Wertebereich der Eigenschaften), beschreibe ich ab dem nächsten Abschnitt innerhalb der Vorstellung der einzelnen Bestandteile des paradigmatischen Modells. Dabei stelle ich auch die Verankerung der Eigenschaften in den Daten dar.
3. Auf Grundlage dieser in Beziehung gesetzten Kategorien untersuche ich die Ausprägung konkreter Eigenschaften in den verschiedenen Episoden und gruppiere sie, wenn ich Ähnlichkeiten feststelle. Das führt mich zu Mustern, welche ich in Form von Aussagen festhalte. In Abschnitt 3.3.4 stelle ich die Ausprägung konkreter Eigenschaften von verschiedenen Episoden dar, gehe auf Ähnlichkeiten zwischen diesen verschiedenen Episoden ein und stelle Aussagen über gefundene Muster auf, in welchen sich der Einfluss auf die Behebungsdauer, den Behebungsverlauf und der Einfluss der verschiedenen Rollen widerspiegelt.

Mit Hilfe des paradigmatischen Modells habe ich Verhaltenskategorien zur neuen Kategorie „Umgang mit der Schwachstelle nach ihrer Veröffentlichung“, welche das Phänomen „Umgang mit Schwachstellenveröffentlichung“ repräsentiert, wie folgt in Beziehung gesetzt:

- (Die ursächliche Bedingung) **Schwachstellenveröffentlichung** führt zum
- (Phänomen) **Umgang mit der Schwachstelle nach ihrer Veröffentlichung**.

- (Kontext) Der spezifische Satz von Eigenschaften, die zu diesem Phänomen gehören, wird **unter dem Aspekt der verschiedenen Rollenkategorien** betrachtet.
- (Die angewandten Strategien) zur **Schwachstellenbehebung bzw. zur Minimierung potentieller Schwachstellenfolgen** führen
  - beeinflusst durch die intervenierenden Bedingungen wie z.B. **Umgang mit Konflikten**
- zur (Konsequenz) **Behebung der Schwachstelle**.

In den folgenden Abschnitten stelle ich die Kategorien, welche ich den verschiedenen Bestandteilen des paradigmatischen Modells zugeordnet habe, sowie deren Eigenschaften und dimensionale Bandbreite vor.

### Ursächliche Bedingung

Als ursächliche Bedingung definiere ich eine neue Kategorie mit Namen „Schwachstellenveröffentlichung“, welche eine Teilmenge der Verhaltenskategorie „Schwachstellendarstellung“ darstellt. Die Eigenschaften der neuen Kategorie bestehen aus ausgewählten Verhaltenskonzepten der Schwachstellendarstellung. Viele dieser Eigenschaften sind binär, also ist diese Eigenschaft bei der Schwachstellenveröffentlichung vorhanden oder nicht. Die Eigenschaften der Kategorie „Schwachstellenveröffentlichung“ unterteilen sich in Basiseigenschaften, optionale Eigenschaften und Metadaten.

Die Basiseigenschaften stellen die benötigte Grundmenge an Eigenschaften für eine Schwachstellenveröffentlichung dar. Folgende Basiseigenschaften gehören zur ursächlichen Bedingung:

- beschreibt_Schwachstelle (siehe Anhang A.1 ID 22)
- bewertet_Schweregrad (siehe Anhang A.1 ID 35)
- schildert_Schwachstellenursache (siehe Anhang A.1 ID 86)
- spezifiziert_Version (siehe Anhang A.1 ID 90)

Wenn ich von einer Schwachstellenveröffentlichung spreche, dann müssen diese Eigenschaften vorhanden sein, da sie in allen von mir betrachteten Episoden aufgetreten sind.

Die optionalen Eigenschaften, also diejenigen, welche bei einer Schwachstellenveröffentlichung nicht unbedingt vorhanden sein müssen, sind:

- grenztEin_Schwachstellenvoraussetzung (siehe Anhang A.1 ID 74)
- gibtAn_Schwachstellenfolge(n)_mögliche (siehe Anhang A.1 ID 69)

- beschreibt_Schwachstellenausnutzung (siehe Anhang A.1 ID 23)
- stelltBereit_Exploit (siehe Anhang A.1 ID 94)
- stelltBereit_Patch (siehe Anhang A.1 ID 95)
- gibtAn_Workaround (siehe Anhang A.1 ID 72)

Die Konzepte *beschreibt_Schwachstellenausnutzung* und *stelltBereit_Exploit* stellen beide die Beschreibung der Ausnutzung einer sicherheitsrelevanten Schwachstelle dar. Ebenso beschreiben die Konzepte *stelltBereit_Patch* und *gibtAn_Workaround* die Angabe einer (Ab-)Sicherung gegen die Ausnutzung der Schwachstelle. Somit reduzieren sich die optionalen Eigenschaften auf folgende vier Punkte:

- Angabe der
  - Ausnutzung
  - (Ab-)Sicherung
  - Folgen
  - Eingrenzung

Mit Metadaten ist der Rahmen in denen die Veröffentlichung stattfindet gemeint. In diesen Eigenschaften halte ich unter anderem auch die Rolle des Melders fest. Folgende Ausprägungen an Metadaten habe ich in den Daten gefunden:

- Melder - zugeordnete Rollenkategorie (betroffene/besorgte Nutzer externe Helfer Verantwortliche des Herstellers ),
- Art der Veröffentlichung - direkte Veröffentlichung beim Hersteller, Verlinkung einer gefundenen Veröffentlichung, über Sicherheitsteam,
- Ort der Veröffentlichung - Kommunikationswerkzeug, Bugtracker, Wiki.

### Phänomen

Das Phänomen, welches ich untersuche, ist konkret formuliert der Umgang mit der Schwachstelle nach ihrer Veröffentlichung. Tabelle 3.1 auf der nächsten Seite stellt die Eigenschaften des Phänomens und ihre dimensionale Bandbreite dar. Da ich das Phänomen unter dem Aspekt der verschiedenen Rollenkategorien untersuche, beziehen sich die ausgewählten Eigenschaften auf die jeweilige Rollenkategorie.

| Eigenschaften | Dimensionale Bandbreite |
|---|---|
| Verzögerungszeit der Reaktion auf Veröffentlichung | kurz<br>mittel<br>lang |
| Bewertung des angegebenen konkreten Schutzes (Veröffentlichung) | unvollständig<br>inakzeptabel<br>übernehmend<br>nicht vorhanden |
| Konkrete Reaktion | drängend<br>Veröffentlichung kritisierend<br>bedankend/lobend |
| Subjektive Einstufung des Sicherheitsrisikos | hoch<br>moderat<br>niedrig |
| Umgang mit Details zur Schwachstellenausnutzung | Zensur<br>Offen<br>Sicherheitsteam |
| Unterstützung | anbieten<br>annehmen |

Tab. 3.1: Eigenschaften des Phänomens

### Kontext

Der Kontext stellt die konkrete dimensionale Ausprägung des Phänomens der einzelnen Episode, bezogen auf die jeweilige Rollenkategorie, dar. Die einzelnen Eigenschaften sind:

**Verzögerungszeit der Reaktion auf Veröffentlichung** bezeichnet die Zeit, welche der erste Akteur einer Rollenkategorie benötigt, um auf die Schwachstellenveröffentlichung zu reagieren. Die Dimension kurz bedeutet dabei, dass er innerhalb von drei Tagen reagiert. Diese Zeitspanne habe ich an die Sicherheitsmeldung des TWiki Sicherheitsteams angelehnt. Das Team veröffentlicht die Meldung in der Regel innerhalb von zwei Tagen. Aufgrund der Berücksichtigung des Wochenendes habe ich diese Zeitspanne um einen Tag erhöht. Die Dimension Mittel bezeichnet den Zeitraum zwischen vier Tagen und einem Monat. Die Dimension Lang umfasst den Zeitraum ab einem Monat. Die Entscheidung für den Monat als Grenze habe ich aufgrund der untersuchten Behebungsdauer getroffen (siehe Abschnitt 3.3.4).

**Bewertung des angegebenen konkreten Schutzes** bedeutet, dass ein Verantwortlicher des Herstellers bzw. ein anderer Hilfeleistender oder Problemlösers die Maßnahme zum Schutz, sofern diese bei der Veröffentlichung vorhanden ist, bewertet (in

den Daten durch Konzepte oder Verahltenskategorien unter der Hauptkategorie „Informationsbewertung“ ersichtlich). Diese Maßnahme kann der Verantwortlicher des Herstellers ablehnen, als unvollständig beurteilen oder für die eigene Maßnahme der Behebung heranziehen und übernehmen.

**Konkrete Reaktion** umfasst verschiedene Verhaltensweisen als Dimensionen. Dies ist zum einen „Drängen nach Lösung“ aus der Hauptkategorie „Umgang mit Konflikten“. Zum anderen sind es die Aktivitäten Bedanken bzw. Loben, welche zu Konzepten aus „Unterstützung geben“ unter der Hauptkategorie „Unterstützung“ gehören. Als weitere Dimension habe ich „Veröffentlichung kritisierend“ gefunden, da diese Verhaltensweise ausschließlich innerhalb der Diskussionen über die Schwachstellenveröffentlichung in den betrachteten Episoden auftritt.

**Subjektive Einstufung des Sicherheitsrisikos** stellt die subjektive Einschätzung des Sicherheitsrisikos durch die verschiedenen Rollen dar. Ich hab die Dimensionen „niedrig“, „moderat“ und „hoch“ direkt aus den Daten, z.B. in der Schwachstelleveröffentlichung, im Bugtracker, Beschreibung beim Codecheckin, entnommen.

**Umgang mit Details zur Schwachstellenausnutzung** beinhaltet die Einschränkung des Zugangs zu Informationen der Schwachstellenausnutzung durch eine Rolle. Die Dimensionen sind „Offen“ für vollen Zugang, „Zensur“ für nachträgliche Löschung dieser Informationen und „Sicherheitsteam“, welches den Zugang zu diesen Informationen regelt.

**Unterstützung** bietet der Akteur einer Rollenkategorie an und kann im Gegenzug von einem Akteur einer anderen Rollenkategorie angenommen werden.

#### Handlungs- und interaktionale Strategien

Die angewandten Strategien zur Bewältigung des Phänomens haben zum kurzfristigen Ziel die Minimierung von potentiellen Schwachstellenfolgen und zum langfristigen Ziel die Behebung der Schwachstelle. Für diese Strategien habe ich die Konzepte, die zur Verhaltenskategorie „konkreten Schutz liefern” gehören, herangezogen. Somit ergeben sich die folgenden eingesetzten Strategien:

- beschreibt_Schadensbegrenzung (siehe Anhang A.1 ID 21)
- gibtAn_Workaround (siehe Anhang A.1 ID 72)
- listetAuf_Sicherheitsmaßnahmen (siehe Anhang A.1 ID 80)
- stelltBereit_Patch (siehe Anhang A.1 ID 95)
- stelltBereit_Release (siehe Anhang A.1 ID 96)

Zu diesen Konzepten habe ich noch die Strategie der „Über- bzw. Annahme eines angegebenen konkreten Schutzes“ hinzugefügt. Dabei übernimmt der Verantwortliche des Herstellers vom externen Helfer Maßnahmen zur Behebung, die bei der Veröffentlichung oder während des Behebungsverlaufs angegeben werden.

### Intervenierende Bedingungen

Zu den intervenierenden Bedingungen habe ich zum einen die Verhaltenskategorien „Umgang mit Beleidigungen“ und „Missverständnisbehandlung“ der Hauptkategorien „Umgang mit Konflikten“ gefunden. Zum anderen ergaben sich zwei neue Bedingungen aufgrund der in den Daten kodierten Konzepte.

Die erste Bedingung ist die Qualitätssicherung, welche Reviews und Tests umfasst. Review ist eine Verhaltenskategorie der Hauptkategorie „Informationsverwertung/-verarbeitung“. Die Unterbedingung Testen kann durch das Konzept „gibtAn_Zwischenmeldung“ (siehe Anhang A.1 ID 73) in den Daten beschrieben, aber auch direkt benannt werden („testing of the SVN release of 1.0 is being organised at the moment“, „Anyone help with testing of this RC would be appreciated“, „please test the SVN copy of 1.0 and see if the exploit is still available“, „It has been sent to testing teams for final approval for release.“)

Die zweite Bedingung ist die Releaseerstellung mit den Unterbedingungen Versionseinschränkung und Releasepolitik. Bei der Versionseinschränkung handelt es sich um eine Prüfung mit dem Ziel, die Versionszweige herauszufinden, welche noch von der Schwachstelle betroffen sind. Denn diese müssen auch behoben werden. Das Vorkommen der Konzepte „spezifiziert_Version“ (siehe Anhang A.1 ID 90) und „spezifiziert_Version_nicht“ (siehe Anhang A.1 ID 91), außerhalb der Schwachstellenveröffentlichung in den Daten, ist ebenso ein Hinweis auf diese Unterbedingung.

Die Releasepolitik äußert sich in der Veröffentlichung der Behebung, d.h. der Hersteller veröffentlicht die Behebung innerhalb des nächsten Release oder so schnell es geht in Form eines Zwischenpatches. Diese Unterbedingung ist durch das Konzept „kündigtAn_Release“ (siehe Anhang A.1 ID 77) in den Daten erkenntlich. Darüber hinaus ist sie auch durch die Begriffe Release und Release Candidate (RC), eine abschließende Testversion, zu finden („It has been sent to testing teams for final approval for release.“, „deploy a release“, „testing of the SVN release of 1.0 is being organised at the moment“, „1.0.14 is very close. Details of the release candidate 1.0.14RC1 can be found [..]“).

### Konsequenzen

Die einzige Konsequenz, welche in allen Episoden auftritt, ist die Behebung der Schwachstelle. In Tabelle 3.2 auf der nächsten Seite sind die Eigenschaften der Konsequenz vorgestellt.

| Eigenschaften | Dimensionale Bandbreite |
|---|---|
| Dauer | innerhalb eines Monats<br>ab einem Monat |
| Bekanntgabe der Behebung | direkt<br>indirekt |

Tab. 3.2: Eigenschaften der Konsequenz Schwachstellenbehebung

Die erste Eigenschaft ist die Behebungsdauer nach der schon vorgestellten Metrik von Witten et al. [23]. Diese Dauer habe ich nach zwei Aspekten unterschieden:

1. **Erste Behebung** Dauer vom Zeitpunkt der Veröffentlichung bis erste offizielle Behebung in Form eines Patches oder Release vorliegt.
2. **Releasebehebung** Dauer vom Zeitpunkt der Veröffentlichung bis zur Bekanntgabe der Behebung in einer offiziellen Releaseankündigung, sofern die Schwachstellenbehebung einem Release zuordnenbar ist.

Die dimensionalen Ausprägungen der Behebungsdauer ist zum einen der Zeitraum innerhalb eines Monats, zum anderen der Zeitraum ab einem Monat. Die Entscheidung für den Monat als Grenze habe ich auch hier aufgrund der in den untersuchten Episoden vorkommenden Behebungsdauer getroffen (siehe Abschnitt 3.3.4).

Unter der Eigenschaft Bekanntgabe der Behebung ist der Umgang des Herstellers mit der Veröffentlichung der Behebung gemeint. D.h. stellt der Hersteller in einem Release die Behebung der Schwachstelle explizit heraus, oder ist es die Aufgabe der anderen Rollen, dies aufgrund von Changelogs oder Codeänderungen selber herauszufinden, denn die Veröffentlichung einer Schwachstelle kann nicht jeden erreichen, den die Meldung eines Releases erreicht. Die dimensionale Ausprägungen umfasst:

**direkt** Die Behebung wird durch eine offizielle Releasankündigung, einen Mailinglisteneintrag oder innerhalb der Diskussion zur Schwachstelle bekannt gegeben.

**indirekt** Eine Behebung durch Codeänderung erfolgt, aber es gibt keine explizite Angabe der Behebung innerhalb der betreffenden Releaseankündigung.

### Zusätzliches Projekt TWiki

Für die Untersuchung des Phänomens Umgang mit Schwachstellenveröffentlichung habe ich mir ein weiteres Projekt gesucht, in welchem der Behebungsverlauf dokumentiert wird und Schwachstellenveröffentlichungen auftreten. Dabei bin ich auf das Projekt TWiki gestoßen, welches diese beiden Kriterien erfüllt und zusätzlich ein Sicherheitsteam für

die Behebung von sicherheitsrelevanten Schwachstellen einsetzt. Durch den letzte Punkt versprach ich mir den Einfluss eines Sicherheitsteam in die Untersuchung mit einfließen zu lassen.

TWiki ist eine Open Source Wiki-Software und ebenfalls unter der GPL. Sie ist größtenteils in der Programmiersprache Perl geschrieben und benutzt für die Datenhaltung Textdateien und keine externe Datenbank.

Der Entdecker einer Schwachstelle soll das Sicherheitsteam von TWiki über die Sicherheitsmailingliste des Projekts kontaktieren[4]. Dieses Team bewertet das Sicherheitsrisiko dieser Schwachstelle und behebt die Schwachstelle aufgrund dieser Einstufung.

Das Sicherheitsteam dokumentiert nachträglich und ausführlich die Schwachstellenbehebung. Die Schwachstellen für meine Episoden zu diesem Projekt habe ich über diese Dokumentationen gefunden. Als Einstieg dazu diente mir die Übersicht über alle Schwachstellendokumentationen[5]. Dabei entdeckte ich eine Schwachstelle, welche vor der Behebung veröffentlicht wurde. Diese Tatsache führte mich zu der Suche nach weiteren Veröffentlichungen[6], welche aber erst immer nach der Behebung durch das Sicherheitsteam stattfanden. Ich habe weitere vier Schwachstellendokumentationen aus dieser Suchergebnisliste für meine Untersuchung nach absteigendem Revisionsdatum ausgesucht.

### 3.3.4 Untersuchungsergebnisse

In diesem Abschnitt gehe ich auf die Ausprägung der Eigenschaften von den in Beziehung gesetzten Kategorien innerhalb verschiedener Episoden ein. Dabei untersuche ich Gemeinsamkeiten und Unterschiede in der dimensionalen Ausprägung der Episoden im Hinblick auf die Erkennung von Mustern und stelle Aussagen über gefundene Muster auf. Dabei möchte ich herausarbeiten, was Einfluss auf die Behebungsdauer und den Behebungsverlauf hat und welchen Einfluss die verschiedenen Rollenkategorien haben.

Ich habe im Rahmen dieser Untersuchung die Episoden W E2, W E4[7], alle Joomla und alle TWiki Episoden (Beschreibung und Inhalt siehe Anhang A.4) betrachtet.

Abbildung 3.3 auf der nächsten Seite stellt die Behebungsdauer (Erste Behebung) einzelner Episoden in absteigender Sortierung dar. Dabei werden die beiden Aspekten der Behebungsdauer, sofern vorhanden, unterschiedlich dargestellt (blau = „Erste Behebung“, rot = „Releasebehebung“).

[4] http://twiki.org/cgi-bin/view/Codev/TWikiSecurityAlertProcess überprüft am 15.08.2008
[5] http://twiki.org/cgi-bin/view/Codev/TWikiSecurityAlerts überprüft am 15.08.2008
[6] http://twiki.org/cgi-bin/view/Codev/WebSearch?search=discloses überprüft am 15.08.2008
[7] Die Episoden werden im Folgenden in der Kurzschreibweise <Erster Buchstabe Projektnamen> E<Episodenidentifikator> formuliert, so bedeutet z.b. W E2, dass es sich hierbei um Episode 2 aus dem Projekt Wordpress handelt.

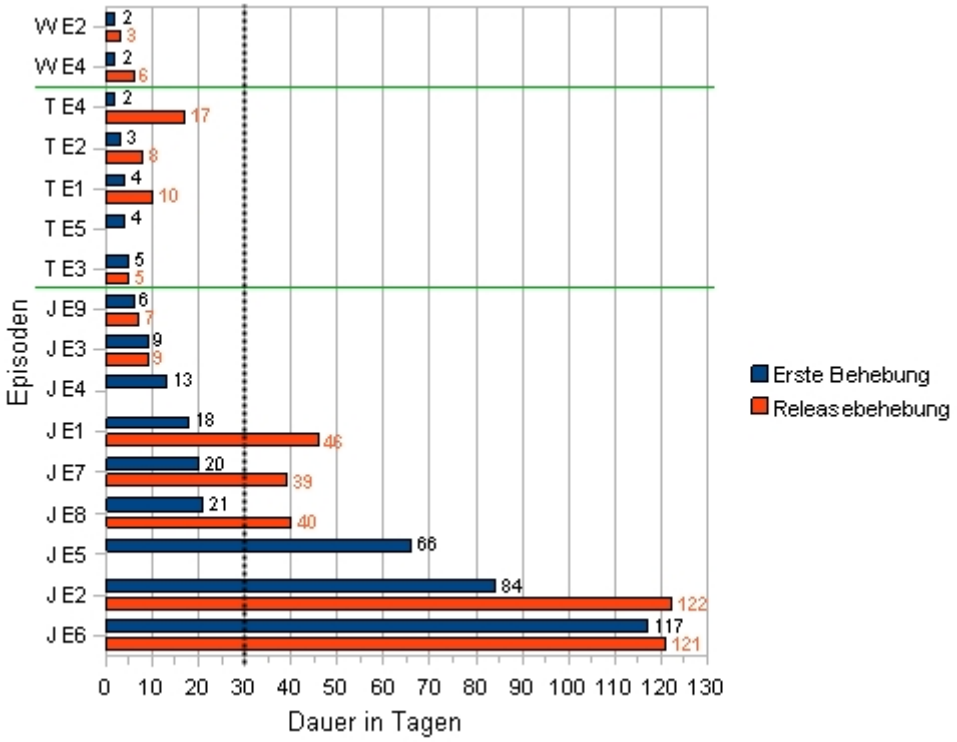


Abb. 3.3: Dauer von Veröffentlichung bis zur Behebung der einzelnen Episoden

Aus der Abbildung wird deutlich, dass nicht alle Schwachstellenbehebungen explizit einem Release zugeordnet werden können und dass der zu einer Schwachstellenbehebung gehörende Release zum eigentlichen Behebungszeitpunkt oder erst danach erscheint.

Die gestrichelte vertikale Linie markiert eine Behebungsdauer von einem Monat (30 Tage). Nur die Joomla Episoden 2, 5 und 6 übersteigen deutlich diesen Zeitraum für die erste Behebung. Die Behebungsdauer bei Joomla Episode 5 beträgt 2,2 Monate, bei Joomla Episode 2 sind es 2,8 Monate und bei Joomla Episode 6 sogar 3,9 Monate. Alle anderen Episoden sind in weniger als einem Monat (die längste Behebungsdauer betrug in diesem Fall sogar nur drei Wochen) behoben worden.

Die grünen Linien trennen die verschiedenen Projekte von einander. Es ist zu sehen, dass die Wordpress Episoden am schnellsten behoben werden, danach kommen die Twiki Episoden und erst dann die Episoden des Projekts Joomla. Ich gehe später auf diese projektspezifischen Unterschiede näher ein.

### Ursächliche Bedingungen

Den Grund für diese deutlich längere Behebungsdauer vermutete ich zuerst bei den Eigenschaften der ursächlichen Bedingung „Schwachstellenveröffentlichung“. Dabei betrachtete ich zunächst die optionalen Eigenschaften Angabe *der Ausnutzung*, *der (Ab-)Sicherung*, *der Folgen* und *der Eingrenzung*, da sie nicht notwendigerweise bei einer Schwachstellenveröffentlichung vorhanden sein müssen.

Aus der Betrachtung ergab sich folgende Aussage:

**Aussage 1:** Die Angabe einer Schwachstellenausnutzung in Form einer Beschreibung oder als Angabe eines Exploits bei einer Schwachstellenveröffentlichung **kann** zu einer schnellen Behebung (innerhalb eines Monats) dieser Schwachstelle durch den Hersteller führen, sofern seine Einstufung des Sicherheitsrisikos durch die Schwachstelle eine schnelle Behebung rechtfertigt.

Der erste Teil dieser Aussage – die Angabe einer Schwachstellenausnutzung in Form einer Beschreibung oder als Angabe eines Exploits bei einer Schwachstellenveröffentlichung kann zu einer schnellen Behebung (innerhalb eines Monats) dieser Schwachstelle führen – ergibt sich daher, dass in den meisten Episoden, welche innerhalb eines Monats behoben wurden, eine Ausnutzungsmöglichkeit innerhalb der Schwachstellenveröffentlichung angegeben wurde. In Tabelle 3.3 auf der nächsten Seite, welche die optionalen Eigenschaften und die Behebungsdauer in Beziehung setzt, ist diese Beobachtung hervorgehoben.

In Tabelle 3.4 auf der nächsten Seite habe ich alle Episoden aufgrund ihrer Behebungsdauer und, ob eine Ausnutzung innerhalb der Schwachstelle angegeben wird, aufgeteilt. Dabei ist zu sehen, dass die Joomla Episoden 2 und 6 die Angabe einer Ausnutzung besitzen. Und zwar in Form eines Exploits:

- Episode 2 - „The following proof-of-concept URI is available: /data/vulnerabilities/exploits/26031.html“ (Anhang A.4.2 Episode 2 Dokument ID 25),
- Episode 6 - „if not replaced, the following will pass easily: <img src="somepic.gif" oNclick="alert('hi')">“ (Anhang A.4.2 Episode Dokument ID 19).

Dennoch weisen beide Episoden eine Behebungsdauer von mehr als einem Monat auf. Aus diesem Grund habe ich den ersten Teil obiger Aussage um „sofern seine Einstufung des Sicherheitsrisikos durch die Schwachstelle eine schnelle Behebung rechtfertigt“ erweitert. Die Gründe dafür stelle ich im nächsten Abschnitt vor.

Die Schwachstellenveröffentlichung der Joomla Episode 5 besitzt keine Angabe einer Ausnutzung und bekräftigt damit den ersten Teil der obigen Aussage. Auf Joomla Episode 1, welche ebenfalls keine Angabe einer Ausnutzung besitzt, aber eine kurze Behebungsdauer hat, gehe ich später nochmal ein.

| Behebungsdauer | Angabe von Eingrenzung | **Angabe von Ausnutzung** | Angabe von (Ab-)Sicherung | Angabe von Folgen |
|---|---|---|---|---|
| innerhalb eines Monats | W E2<br>J E1<br>J E4<br>J E7<br>T E1<br>T E2<br>T E3<br>T E4<br>T E5 | W E2<br>W E4<br>J E3<br>J E4<br>J E7<br>J E8<br>J E9<br>T E1<br>T E2<br>T E3<br>T E4<br>T E5 | W E4<br>J E1<br>J E9<br>T E2<br>T E3<br>T E4<br>T E5 | W E4<br>J E1<br>J E3<br>J E7<br>J E8<br>J E9<br>T E1<br>T E2<br>T E3<br>T E4<br>T E5 |
| ab einem Monat | J E2<br>J E5 | J E2<br>J E6 | J E5<br>J E6 | J E2<br>J E5<br>J E6 |

Tab. 3.3: Beziehung von Dauer und optionalen Eigenschaften der ursächlichen Bedingung

| Behebungsdauer | Mit Angabe der Ausnutzung | Ohne Angabe der Ausnutzung |
|---|---|---|
| innerhalb eines Monats | W E2, W E4, J E3, J E4, J E7, J E8, J E9, T E1, T E2, T E3, T E4, T E5 | J E1 |
| ab einem Monat | J E2, J E6 | J E5 |

Tab. 3.4: Beziehung der Ausnutzungsangabe und der Behebungsdauer

### Einstufung des Sicherheitsrisikos einer Schwachstelle

Bei der Betrachtung der konkreten Ausprägungen der verschiedenen Kategorien der Joomla Episoden 2 und 6 bin ich zu folgender Aussage gekommen, die von der Erkenntnis von Arora et al. [2], dass auf schwerwiegendere Schwachstellen intensiver vom Hersteller reagiert wird, bekräftigt wird:

**Aussage 2:** Auf eine von ihm niedrig oder moderat eingestufte Schwachstelle wird der Hersteller nicht unbedingt so schnell reagieren, wie auf eine von ihm als schwerwiegender eingestufte Schwachstelle.

Eine schwerwiegende Einstufung bedeutet eine höhere Einschätzung des Sicherheitsrisikos durch eine Schwachstelle, welche durch die Akteure verschiedener Rollen oder durch Instanzen wie Schwachstellendatenbanken subjektiv oder aufgrund von standardisierten Einstufungen, vorgenommen wird.

Aussage 2 begründet sich darin, dass in diesen beiden Episoden aus der Sicht des Hersteller (Akteure der Rollenkategorie Verantwortlicher des Herstellers) das jeweilige Sicherheitsrisiko durch die veröffentlichte Schwachstellen als niedrig eingestuft wurde:

- Episode 2 - „SECURITY [LOW] Fixed XSS issue in Search Component.“ (Anhang A.4.2 Episode 2 Dokument ID 13)
- Episode 6 - „SECURITY [LOW] Fixed case-sensitive flaw in Input Filter.“ (Anhang A.4.2 Episode 6 Dokument ID 13)

Analog zur Dauer der Behebung bleibt vorerst auch eine Reaktion des Herstellers auf die Schwachstellenveröffentlichung aus, so dass man die obige Aussage auch auf die sichtbare Herstellerreaktion ausweiten könnte:

- in Episode 2 reagiert zwar ein Verantwortlicher des Herstellers und gibt Zwischenmeldung zum Behebungsverlauf, nur wird diese von einem externen Helfer widerlegt (Anhang A.4.2 Episode 2 Dokument ID 6), erste konkrete Reaktion auf die Schwachstelle des Herstellers taucht am 03.01.2008 (nach 83 Tagen) auf
- in Episode 6 ist die erste Reaktion am 04.01.2008 (nach 83 Tagen) sichtbar (Anhang A.4.2 Episode 6 Dokument ID 19)

Betrachte ich nun Joomla Episode 5, bei welcher der Hersteller das Sichereitsrisiko als moderat eingestuft hat („Priorität: Mittel Hoch“ AnhangA.4.2 Episode 5 Dokument ID 18), so finde ich hier eine weitere Bestätigung der obigen Aussage. Darüber hinaus führte Aussage 2 zu der Einschränkung in Aussage 1.

Tabelle 3.5 auf der nächsten Seite (Zeile hervorgehoben) zeigt, dass auch bei niedrig und moderat eingestuften Schwachstellen eine Behebung innerhalb eines Monats stattfindet, was eine Abschwächung der Aussage 2 bedeuten könnte. Das veranlasste mich zur Untersuchung weiterer Einflüsse für eine schnelle Behebung, auf welche ich im nächsten Abschnitt eingehe.

| Behebungsdauer | Einstufung des Sicherheitsrisikos | |
|---|---|---|
| | moderat | niedrig |
| **innerhalb eines Monats** | **T E1, T E2** | **J E7, J E8, T E5** |
| ab einem Monat | J E5 | J E2, J E6 |

Tab. 3.5: Beziehung zwischen Behebungsdauer und niedrig bzw. moderat eingestufter Schwachstellen

### Art und Ort der Veröffentlichung

Durch die Betrachtung der dimensionalen Ausprägung der in Beziehung gesetzten Kategorien in den Episoden T E1, T E2 und T E5 aus dem TWiki Projekt und die Episoden J E7 und J E8 bin ich zu folgender Aussage gelangt:

**Aussage 3:** Die Art des vorgegebenen Veröffentlichungskanals hat Einfluss auf die Behebungsdauer.

Im Folgenden betrachte ich die Episoden T E1, T E2 und T E5 aus dem TWiki Projekt und die Episoden J E7 und J E8 aus dem Joomla Projekt. Mir ist aufgefallen, dass es zwischen den beiden Projekten TWiki und Joomla Unterschiede in der Behebungsdauer gibt. Die Behebung der Schwachstellen im Projekt TWiki findet in jeder Episode innerhalb einer Woche statt (T E1 4 Tage, T E2 3 Tage und T E5 4 Tage), während die Behebung in Joomla fast drei Wochen dauern (J E7 20 Tage und J E8 21 Tage). Folgende Unterschiede habe ich bei den Episoden zwischen den Projekten in den Daten gefunden:

1. Art und Ort der Veröffentlichung
   - TWiki benutzt ein Sicherheitsteam, an welches Schwachstellen gemeldet werden und das sich um sicherheitsrelevante Probleme kümmert.
   - Joomla benutzt für den Umgang mit Schwachstellenmeldungen einen Bugtracker, es gibt aber keine Möglichkeit Schwachstellenmeldungen explizit als solche innerhalb des Tracker zu kennzeichnen.
2. Einstufung des Sicherheitsrisikos
   - TWikis Sicherheitsteam bewertet das Sicherheitsrisiko eigenständig, nachdem die Schwachstelle dem Team gemeldet wurde („The SecurityTeam triages the seriousness of the issue“[8]).
   - Bei Joomla legt der Melder der Schwachstelle deren initiale Einstufung bzgl. des Sicherheitrisikos im Bugtracker fest.

[8] http://twiki.org/cgi-bin/view/Codev/TWikiSecurityAlertProcess überprüft am 15.08.2008

3. Veröffentlichung der Behebung
    - Das Sicherheitsteam von TWiki veröffentlichte die Behebung der Schwachstellen in den drei untersuchten Episoden in Form eines Patches bevor die Behebung regulär mit dem nächsten Release ausgeliefert wurde.
    - Bei den betrachteten Joomla Episoden fand die Behebung in der Codebasis statt und wurde erst mit dem Release veröffentlicht (Releaseerstellung), d.h. ein Nutzer kann vor dem Release zur Behebung der Schwachstelle die Änderungen nur aus der Codebasis übernehmen.

Daraus ergibt sich die Zwischenerkenntniss, dass die Behebung niedrig oder moderat eingestufter Schwachstellen in einem Projekt mit Sicherheitsteam, an welches die Schwachstellen gemeldet wurden und welches dann das Sicherheitsrisiko durch die Schwachstelle bewertet, schneller ist als bei einem Projekt ohne festes Sicherheitsteam und einem Bugtracker als Meldewerkzeug von Schwachstellen.

Aus dem Diagramm 3.3 auf Seite 56 geht hervor, dass bei dem Projekt TWiki alle Schwachstellen schneller behoben wurden als die betrachteten Schwachsellen im Projekt Joomla, was die Vermutung nahe legt, obige Zwischenerkenntnis zu verallgemeinern. Um aber eine Verallgemeinerung zuzulassen, musste ich die Zwischenerkenntnis noch für das Projekt Wordpress überprüfen, da alle Wordpress Episoden noch schneller behoben wurden als die Episoden von TWiki.

Bei beiden untersuchten Wordpress Episoden wurden die Schwachstellen durch den jeweiligen Schwachstellen-Berichterstatter (gehört zu Rollenkategorie „externer Helfer“, siehe Anhang A.3.2) entdeckt und direkt beim Hersteller veröffentlicht (W E2 im Kommunikationswerkzeug von Wordpress und W E4 im Bugtracker). Sowohl der Schwachstellen-Berichterstatter als auch die Verantwortlichen des Herstellers haben beide Schwachstellen als schwerwiegend eingestuft. In beiden Fällen hat ein Verantwortlicher des Herstellers die Veröffentlichung kritisiert, da die direkte Schwachstellenveröffentlichung nicht dem üblichen Weg der Schwachstellenmeldung über die dafür vorgesehene Mailingliste in diesem Projekt entsprach.

Dadurch ergibt sich, dass die Art der Meldung und der dafür vom Projekt vorgesehene Veröffentlichungskanals ein weiteres Einflusskriterium für die Behebungsdauer ist. Aus der obigen Zwischenerkenntnis, welche den Einfluss eines Sicherheitsteams auf die Behebungsdauer beschreibt, und dem kritischen Umgang mit Veröffentlichungen bei Wordpress ergibt sich für mich die oben formulierte Aussage 3:

> Die Art des vorgegebenen Veröffentlichungskanals hat Einfluss auf die Behebungsdauer.

Diese Aussage wandte ich auf die konkreten Daten meiner drei untersuchten Projekte an und das führte mich zu folgenden Teilaussagen, welche sich auf meine betrachteten Episoden beziehen:

1. Das Projekt mit dem Sicherheitsteam, welches die Veröffentlichungen von Schwachstellen über eine eigens dafür eingerichtete Schwachstellenmeldungsmailingliste entgegen nimmt und daraufhin das Sicherheitsrisiko selber einschätzt, behob die Schwachstellen innerhalb einer Woche.
2. Das Projekt, bei welchem Schwachstellen über einen Bugtracker gemeldet werden sollen ohne dass die Schwachstellen als solche gekennzeichnet werden können, benötigte im Schnitt eine längere Behebungsdauer als das Projekt aus Punkt 1. Ich vermute das liegt daran, dass aufgrund der fehlenden Kennzeichnung der Bugmeldung als sicherheitsrelevante Schwachstelle und aufgrund einer niedrigen Einstufen des Sichereitsrisikos, die Bugmeldung als Schwachstellenveröffentlichung zu spät erkannt wurde und dadurch die späte Reaktion zustande kam.
3. Eine direkte Schwachstellenveröffentlichung beim Hersteller (z.B. in einem Kommunikationswerkzeug oder einem Bugtracker), welche nicht über dem vom Hersteller vorgesehenen Weg der Schwachstellenmeldung geschah und welche sowohl der Schwachstellen-Berichterstatter als auch der Hersteller als schwerwiegend einstufte, wurde innerhalb einer Woche behoben.

Neben den beiden Wordpress Episoden zeigt sich Teilaussage 3 auch in der Joomla Episode 3, in welcher ein betroffener/besorgter Nutzer die Schwachstellenveröffentlichung anstatt im Bugtracker im Kommunikationswerkzeug verlinkte. Diese hat der Hersteller innerhalb von sieben Tagen behoben. Die Abweichung dieser Dauer von der im Diagramm 3.3 auf Seite 56 liegt darin, dass die Veröffentlichung zu einem anderen Projekt, aus dem Joomla hervorgegangen ist, gehörte und diese erst für Joomla bestätigt werden musste (siehe Anhang A.4.2 Episode 3 Dokument ID 14). Von dieser Bestätigung der Schwachstelle bis zum offiziellen Release, welcher die Behebung beinhaltet, dauerte es genau sieben Tage.

Aufgrund der angeführten Erkenntnisse erweitere ich Aussage 3 zu:

**Aussage 3a:** Die Art des vorgegebenen Veröffentlichungskanals hat Einfluss auf die Behebungsdauer. Projekte mit einem Veröffentlichungskanal, welcher direkt für Schwachstellenmeldungen vorgesehen ist, und bei denen Schwachstellen auch explizit gekennzeichnet werden, beheben eine Schwachstelle schneller als Projekte, die einen Veröffentlichungskanal benutzen, über den auch nicht sicherheitsrelevante Defekte in der Software gemeldet werden.

**Aussage 3b:** Auf der anderen Seite wird scheinbar eine Schwachstelle, welche dem Hersteller offen, aber ohne Nutzung des vorgegebenen Veröffentlichungskanals, gemeldet wird, schnell behoben.

Eine Ausnahme der Aussage 3b stellt auf dem ersten Blick die Episode 1 von Joomla dar. Bei dieser Episode beträgt die Behebungsdauer 18 Tage und damit ist sie die dritt längste Episode innerhalb derer, die in weniger als einem Monats behoben wurden, obwohl dort

der Melder nicht den vorgegebenen Veröffentlichungskanal für die Schwachstellenveröffentlichung benutzt hat. Aus der Untersuchung dieser Episode ergaben sich weitere Erkenntnisse zu Einflüssen auf die Behebungsdauer. Diese stelle ich im nächsten Abschnitt vor.

### Weitere Einflüsse auf die Behebungsdauer

Aus der Untersuchung der Joomla Episode 1 ergaben sich folgende weitere Aussagen, in denen Eigenschaften beschrieben werden, welche anscheinend Einfluss auf die Behebungsdauer haben können:

1. Versionsspezifizierung. Diese Schwachstelle hat ein externer Helfer für den jüngeren Versionszweig über den vorgegebenen Veröffentlichungskanal gemeldet, bevor diese Schwachstelle für den älteren Versionszweig veröffentlicht wurde. Der Hersteller hat die Schwachstelle in diesem Versionszweig innerhalb von sechs Tagen behoben. Eine Prüfung, ob die Schwachstelle auch im anderen Versionszweig auftritt blieb aus. (siehe Anhang A.4.2 Episode 1 Dokument ID 5)
2. Releaseerstellung. Die Priorität der Entwicklung innerhalb des Projekts lag vor dem Bekanntwerden der im älteren Versionszweig befindende Schwachstelle bei der Fertigstellung des jüngeren Versionszweiges. Dadurch und aufgrund der Releasepolitik von Joomla[9], welche besagt, dass verschiedene Behebungen von Schwachstellen und Defekten, die in einem gemeinsamen Zeitraum auftreten, als ein Release veröffentlicht werden, dauerte es bis zur Veröffentlichung der endgültigen offiziellen Behebung 46 Tage.
3. Qualitätssicherung. Bis zur ersten Veröffentlichung einer Behebung in Form eines Release Candidate durch den Hersteller dauerte es 18 Tage aufgrund von Tests und Reviews der Behebung in der Codebasis. Bis zum endgültigen Release führte der Hersteller noch weitere Tests und Anpassung an der Codebasis durch, was die Veröffentlichung des Release noch weiter verschob.
4. Drängen. In dieser Episode hat neben den drängenden Anfragen nach einer offiziellen Behebung durch betroffene/besorgte Nutzer ein externer Helfer effektivere Maßnahmen zur Gewinnung der Aufmerksamkeit des Herstellers eingesetzt. Dazu zähle ich die direkte Kontaktierung einzelner Verantwortliche des Herstellers („I have sent an open letter to [..]“ siehe Anhang A.4.2 Episode 1 Dokument ID 1) und die Darstellung des Sicherheitsrisikos der veröffentlichten Schwachstelle durch die Ausnutzung dieser Schwachstelle („I just managed to create a super admin account

[9] http://forum.joomla.org/viewtopic.php?f=267&t=70117&st=0&sk=t&sd=a&start=30#p373406 überprüft am 14.08.2008. Innerhalb dieses Forenbeitrags gibt ein Verantwortlicher des Herstellers Aussagen zur Releasepolitik und zur Veröffentlichung von Schwachstellenbehebungen in Joomla ab.

on his new live site“ siehe Anhang A.4.2 Episode 1 Dokument ID 1). Neben dem typischen Drängen (siehe Abschnitt 3.4) der Nutzer, stellten die externen Helfer Argumente dar, die für eine schnelle Behebung und eine schnelle Veröffentlichung eines Release sprachen.

Nachdem ich diese Eigenschaften in allen anderen Episoden untersucht habe, konnte ich nur Versionsspezifizierung und Releasepolitik als weitere Einflüsse auf die Behebungsdauer innerhalb meiner Episoden ausmachen:

Die Versionsspezifizierung hat bei der korrekten Angabe aller betroffenen Versionszweige eine schnelle Reaktion des Herstellers auf die Schwachstellen zur Folge. Der Hersteller übersieht dadurch nicht aus Versehen oder aufgrund von gesetzten Prioritäten ebenfalls betroffene Versionszweige. Dennoch sollte es zu seinen Aufgaben gehören, dafür zu sorgen, dass alle unterstützten und offiziell veröffentlichten Versionszweige bei einer Schwachstelle geprüft werden, wie es im Wordpress Projekt geschieht (siehe Anhang A.4.1 Episode 4 Dokument ID 17).

Bei einer Releasepolitik, welche die Behebungsveröffentlichung an die Veröffentlichung größerer Releases koppelt, dauert die dazugehörige Behebung länger als bei einer Behebungsveröffentlichung in Form von Zwischenpatches. Das zeigt sich an der kürzeren Behebungsdauer des Projekts TWiki gegenüber dem Projekt Joomla. Das Sicherheitsteam von TWiki[10] veröffentlicht je nach Einstufung des Sicherheitsrisikos eine Schwachstellenbehebung in Form eines Patches oder eines Release innerhalb kurzer Zeit. Die Releasepolitik von Joomla[11] sieht vor, dass Schwachstellenbehebungen jeder Sicherheitsrisikoeinstufung in einem bestimmten Zeitraum zu einem Release zusammengefasst und veröffentlicht werden.

### Einflüsse auf den Behebungsverlauf

Qualitätssicherung, Drängen und der externe Helfer haben Einfluss auf den Behebungsverlauf, was sich nicht zwingend auf die Behebungsdauer auswirkt:

Die Qualitätssicherung der Software nach der Behebung wird in meinen Daten größtenteils durch Tests und Reviews gewährleistet. Das Weglassen dieser Praktiken würde zwar zu einer schnelleren Behebung führen, aber Sicherheit ist ein Qualitätsaspekt, welcher durch eben solche Praktiken gewährleistet wird.

Ich habe den Einfluss des Phänomens Drängen im Rahmen der Untersuchung des Phänomens Umgang mit Schwachstellenveröffentlichung nur dahingehend untersucht, dass ich festgestellt habe, dass das Drängen in direkter oder unkonventioneller Form zu

[10] http://twiki.org/cgi-bin/view/Codev/TWikiSecurityAlertProcess überprüft am 15.08.2008
[11] http://forum.joomla.org/viewtopic.php?f=267&t=70117&st=0&sk=t&sd=a&start=30#p373406 überprüft am 14.08.2008

einer Reaktion durch den Hersteller führt. Im Abschnitt 3.4 gebe ich einen Ausblick auf mögliche Vermutungen und Fragen zum Phänomen Drängen, welche in zukünftigen Untersuchungen geklärt werden könnten.

Neben diesen beiden Verhaltensweisen gehört die Rolle des externen Helfers zu den Faktoren, welche Einfluss auf den Behebungsverlauf haben. Er benutzt im Vergleich zum Drängen nach Schwachstellenbehebung ohne Informationsbereitstellung effektivere Möglichkeiten, um die Aufmerksamkeit des Herstellers zu erlangen und er schafft es durch sinnvolle Argumente den Hersteller von der Wichtigkeit bestimmter Aktionen zu überzeugen.

Joomla Episode 1 stellt ein gutes Beispiel dafür dar. So führen in dieser Episode externe Helfer die Begründung für die Dringlichkeit der Schwachstellenbehebung an, dass der ältere Versionszweig viel häufiger eingesetzt wird als der Jüngere. Darüber hinaus versorgt der externe Helfer den betroffenen Nutzer mit Informationen und inoffiziellen Maßnahmen zur Minimierung potentieller Schwachstellenfolgen bzw. zur Schwachstellenbehebung. Tabelle 3.6 zeigt die konkreten Maßnahmen, welche der externe Helfer in den verschiedenen Episoden liefert.

| | Episoden |
|---|---|
| beschreibt_Schadensbegrenzung | J E1 |
| gibtAn_Workaround | W E2, J E1, J E3, J E9 |
| listetAuf_Sicherheitsmaßnahmen | J E1, J E3 |
| stelltBereit_Patch | J E1, J E2, J E3, J E5, J E6, J E7 |

Tab. 3.6: Schutzmaßnahmen durch den externen Helfer in den verschiedenen Episoden

Diese Maßnahmen können schon während der Veröffentlichung mit angegeben werden, nur habe ich für diesen speziellen Fall keine aussagekräftigen Informationen zum Einfluss auf die Behebungsdauer bzw. den Behebungsverlauf gefunden. Weitere Eigenschaften deren Einfluss auf den Behebungsverlauf nicht feststellbar waren, stelle ich im nächsten Abschnitt vor.

### Eigenschaften ohne feststellbaren Einfluss auf den Behebungsverlauf

Bei folgenden Eigenschaften konnte ich keinen konkreten Einfluss durch meine Untersuchungen bestimmen:

Die Angabe von möglichen Folgen (optionale Eigenschaft der ursächlichen Bedingung), welche sich aus der Ausnutzung einer Schwachstelle ergeben, scheint die Informationen zur Ausnutzung und die Bewertung des Sicherheitsrisikos, welche in der Schwachstellenveröffentlichung angegeben werden, zu unterstreichen und kann dem Hersteller bei der eigenen Einstufung der Schwachstelle dienen.

Die Verhaltenskategorien „Umgang mit Beleidigungen“ und „Missverständnisbehandlung“ (intervenierende Bedingungen) aus der Hauptkategorie „Umgang mit Konflikten“ habe ich in einigen der untersuchten Episoden zum Phänomen „Umgang mit Schwachstellenveröffentlichung“ gefunden. Diese Konflikte haben keine sichtbaren Auswirkungen, da sie fast immer gelöst werden. Ausnahme bildet die Joomla Episode 2. In dieser Episode führt die Diskussion vom Thema ab und nimmt danach an Beleidigungen zu. Dies geschieht nachdem die Schwachstellenbehebung eingecheckt worden ist und endet damit, dass die Diskussion durch Schließung des Themas von Seiten des Herstellers beendet wird.

Zu Verhaltensweisen, welche in die Hauptkategorie „Unterstützung“ eingeordnet werden können, habe ich keine für mich aussagekräftigen Einflüsse erkennen können, obwohl diese Verhaltensweisen innerhalb der betrachteten Episoden vorkommen. Diese Verhaltensweisen sollten in zukünftigen Untersuchungen betrachtet werden.

### 3.3.5 Kernaussagen über das Phänomen

Die Erkenntnisse meiner Analyse der Episoden W E2, W E4, sowie aller Joomla und aller TWiki Episoden beschreiben nur die Verhaltensweisen, Rollen und deren Beziehungen, die ich in den zugrunde liegenden Daten gefunden habe. Zusammenfassend lassen sich aufgrund der Untersuchung des Phänomens „Umgang mit Schwachstellenveröffentlichung“ folgende Aussagen formulieren:

1. Auf eine von ihm niedrig oder moderat eingestufte Schwachstelle hat der Hersteller nicht unbedingt immer so schnell reagiert, wie auf eine von ihm als schwerwiegender eingestufte Schwachstelle.
2. Die Angabe einer Schwachstellenausnutzung innerhalb einer Schwachstellenveröffentlichung führte in den untersuchten Daten zu einer schnellen Schwachstellenbehebung durch den Hersteller, sofern dies nicht durch Aussage 1 eingeschränkt wurde.
3. Die Projekte mit einem Veröffentlichungskanal, welcher direkt für Schwachstellenmeldungen vorgesehen ist und dadurch Schwachstellen explizit kennzeichnet, haben Schwachstellen schneller behoben als Projekte, deren Veröffentlichungskanal auch zur Meldung anderer Defekte dient.
4. Schwachstellen, welche dem Hersteller offen, aber ohne Nutzung des vorgegebenen Veröffentlichungskanals, gemeldet wurden, hatten eine kurze Behebungsdauer.
5. Das Fehlen der konkreten Angabe aller betroffenen Versionen führte zu einer längeren Behebungsdauer eines Versionszweigs verglichen mit einem anderen. Durch Prüfung aller Versionszweige, wie in einem anderen Projekt geschehen, kann dies verhindert werden.

6. Die Behebung in einem Projekt, welches die Behebungsveröffentlichung immer an einen Release koppelt, dauerte länger als in einem Projekt, welches Behebungen in Form von Zwischenpatches veröffentlicht.
7. Die Durchführung von qualitätssichernden Praktiken erhöhte sich die Behebungsdauer.
8. Das Drängen in direkter (z.B. E-Mail) oder unkonventioneller (z.B. Demonstration eines Angriffs) Form führte zur Reaktion des Hersteller auf die Schwachstelle.
9. Die Akteure, welche zur Rollenkategorie des externen Helfers gehören, versorgten den betroffenen/besorgten Nutzer mit Informationen und inoffiziellen Maßnahmen zur Minimierung potentieller Schwachstellenfolgen bzw. zur Schwachstellenbehebung. Durch sinnvolle und nachvollziehbare Argumente leisteten sie erfolgreich Überzeugungsarbeit gegenüber dem Hersteller.
10. Zu den optionalen Eigenschaften bei der Schwachstellenveröffentlichung Angabe von „(Ab-)Sicherung“ und Angabe von „Folgen“ konnte ich innerhalb der Untersuchung keine aussagekräftigen Erkenntnisse gewinnen.
11. Zu den Verhaltenskategorien „Umgang mit Beleidigungen“, „Missverständnisbehandlung“ und „Unterstützung“ hatte ich zu wenig Daten und Anhaltspunkte, um Aussagen über ihren möglichen Einfluss auf den Behebungsverlauf geben zu können.

Es gibt Verhaltensweisen, welche Akteure der Rollen externer Helfer oder betroffener/-besorgter Nutzer aktiv eingesetzt haben und wodurch die Behebungsdauer beeinflusst wurde. Dazu zähle ich die Angabe der Schwachstellenausnutzung, die Veröffentlichung der Schwachstelle ohne Nutzung des vorgegebenen Veröffentlichungskanals und die Versionsspezifizierung verschiedener betroffener Versionszweige.

Auf der anderen Seite hat der Verantwortliche des Herstellers durch die Bereitstellung eines Veröffentlichungskanals für Schwachstellen, die Releasepolitik, welche die Veröffentlichung von Schwachstellenbehebungen regelt, und die Prüfung aller Versionszweige auf die veröffentlichte Schwachstelle die Behebungsdauer ebenfalls beeinflusst.

Die Einstufung des Sicherheitsrisikos als hoch, sorgte zwar für eine schnellere Behebung, der Schweregrad der Schwachstelle selbst bleibt von den Akteuren unbeeinflusst. Hierbei konnten nachvollziehbare Argumente der externen Helfer oder direkte Formen des Drängen zu einer schnellere Reaktion des Herstellers auf die Schwachstelle führen.

Neben der Behebungsdauer hatte der externe Akteur Einfluss auf den Behebungsverlauf, indem Informationen für den betroffenen/besorgten Nutzer bereitstellte und konkreten Schutz lieferte. Ebenfalls Einfluss auf den Behebungsverlauf hat Qualitätssicherung in Form von Tests und Reviews. Das Weglassen dieser Praktiken würde zwar die Behebungsdauer verkürzen, aber der Sicherheit nicht vollständig gewährleisten.

Die Eigenschaften und Verhaltenskategorien Angabe von „(Ab-)Sicherung“, Angabe von „Folgen“, „Umgang mit Beleidigungen“, „Missverständnisbehandlung“ oder „Unterstützung“ brachten aufgrund mangelnder Anzeichen in den Daten keine aussagekräftigen Erkenntnisse.

## 3.4 Ausblick: Drängen nach Schwachstellenbehebung

Das Phänomen „Drängen nach Schwachstellenbehebung“ hat sich neben dem Phänomen „Umgang mit Schwachstellenveröffentlichung“ als ein weiteres interessantes Phänomen während der Entwicklung der Verhaltensklassifikation herausgestellt. Zu diesem Phänomen gehörende Konzepte habe ich zu einer eigenen Verhaltenskategorie zusammengefasst und unter der Hauptkategorie „Umgang mit Konflikten“ angesiedelt.

Drängen ist eine Verhaltensweise eines Akteurs (in meinen Daten ist das in der Regel der betroffene/besorgte Nutzer), welche im Verlauf der Behebung stattfindet. Sie äußert sich als Nachfrage nach einem Ergebnis einer Aktivität, in meinen Episoden war das die Behebung der Schwachstelle bzw. die Veröffentlichung eines Release, welcher die Behebung beinhalten soll. Diese Nachfrage findet ohne weitere Informationsbereitstellung statt, d.h. innerhalb der Diskussion zum Behebungsverlauf werden beim Drängen keine bis kaum Informationen geliefert. Sollten Informationen geliefert werden, besitzen diese selbst kaum Relevanz zur aktuellen Aufgabe und tragen nicht zur Schwachstellenbehebung bei.

Obwohl das Phänomen in den untersuchten Episoden häufig aufgetreten ist und scheinbar Einfluss auf den Behebungsverlauf hat, habe ich mich aus Zeitgründen dafür entschieden nur den Aspekt des Phänomens zu betrachten, der beim Umgang mit Schwachstellenveröffentlichung auftritt. Und zwar wie im vorigen Abschnitt beschrieben, das Drängen in direkter oder unkonventioneller Form, damit der Hersteller auf die Schwachstelle reagiert.

In einer Untersuchung, wie ich sie beim Phänomen „Umgang mit Schwachstellenveröffentlichung“ durchgeführt habe, könnten die folgenden Fragen und Vermutungen zu diesem Phänomen geklärt werden:

- Was für eine Verhaltensweise stellt Drängen dar?
  - eine Folge eines intransparenten Behebungsverlaufs - Folge?
  - ein Symptom eines intransparenten Behebungsverlaufs - Ausprägung?
  - ein Mittel mehr Transparenz im Behebungsverlauf zu erzielen - intervenierende Bedingung?
- Äußert der Akteur durch Drängen Unzufriedenheit aus? Wenn ja, worüber ist er unzufrieden?

  - keine Behebung bisher,
  - kein Release bisher,
  - kein Ursachenfund bisher,
  - immer noch dem Problem bzw. der Schwachstelle ausgeliefert sein ohne was dagegen unternehmen zu können.

- Vielleicht möchte der Akteur durch Drängen Druck auf den Hersteller ausüben, um schneller an Ergebnisse/Informationen zu gelangen. Wie erfolgversprechend ist dieser Einsatz durch Drängen?
- Drängen könnte darauf hindeuten, dass der Behebungsverlauf nicht wie gewünscht verläuft. Gründe für solch einen Behebungsverlauf könnten sein:
  - Releasepolitik - Behebung wird erst nach vorgegebenen Workflows und Kriterien veröffentlicht.
  - Prioritätenvergabe - andere Aspekte bei der Entwicklung der Anwendung werden vorgezogen.
  - Unfähigkeit bestimmte Informationen zu erlangen, welche für den Behebungsverlauf relevant sind.
  - Personalmangel.

Antworten auf diese Fragen bzw. Bestätigungen dieser Vermutungen könnten helfen diese Verhaltensweise besser zu verstehen. Stellen sich einige Vermutung als wahr heraus, so könnte dass unter Umständen bedeuten, dass das Vorkommen von Drängen während des Behebungsverlaufs Hinweis auf Probleme oder sogar Intransparenz innerhalb des Behebungsverlaufs geben kann. Woraus sich weitere Fragen ableiten würden:

- Welche Auswirkungen hätten diese Probleme bzw. Intransparenz auf den Behebungsverlauf oder seine Dauer?
- Und wenn Drängen auf Probleme hinweisen könnte, gäbe es dann Verhaltensweisen, welche das Gegenteil darstellen?

Auf der anderen Seite sollte auch untersucht werden, ob Drängen selbst zu Problemen oder negativen Auswirkungen im Behebungsverlauf führen kann.

## 3.5 Vermeidung

In diesem Abschnitt stelle ich einige Verhaltensweisen und Praktiken zur Vermeidung von Schwachstellen vor, auf welche ich bei der Analyse meiner Daten gestoßen bin. Wobei

ich an dieser Stelle nochmal betonen muss, dass diese Aufgabe nie im Fokus meiner Arbeit lag und ich nur die Vermeidungsaspekte berücksichtigt habe, welche mir bei der Untersuchung der Schwachstellenbehebung aufgefallen sind.

Die erste Praktik, welche ich in den Daten gefunden habe, ist der Review des Quellcodes nachdem eine Schwachstelle behoben wurde, um Defekte im Code zu ermitteln, welche ähnliche Schwachstellen ermöglichen. Im Projekt Joomla wird dies wie folgt beschrieben:

> „ The usual case when a security vulnerability is found - of any nature, is that an analysis is done to secure the codebase from future attacks of similar nature and additional hardening is taken.“[12]

Eine andere Verhaltensweise zur Vermeidung besteht darin aus anderen Open Source Anwendungen mit ähnlichen Funktionen Praktiken zur Vermeidung zu übernehmen. In der ersten Joomla Episode äußert ein Verantwortlicher des Herstellers diese Maßnahme („Basically we've done as much as, for example, Drupal or SMF do in there forms.“ siehe Anhang A.4.2 Episode 1 Dokument ID 1).

Praktiken zur Vermeidung von Schwachstellen können neben Änderungen im Code auch durch Sicherheitsmaßnahmen auf Seiten des Nutzers durchgeführt werden. Dabei wird z.B. in Joomla Episode 1 vorgeschlagen, um sich vor den durch die Schwachstelle möglichen CSRF Angriffen zu schützen, besteht die Möglichkeit Prism[13] für die Administrierung zu nutzen. Die Sitzung des Administrators bleibt innerhalb der Prism Instanz, während auf CSRF vorbereitete Anfragen im eigentlichen Webbrowser verbleiben.

Neben diesen Sicherheitsmaßnahmen gehört auch das sichere Konfigurieren der Anwendung zur Vermeidung von Schwachstellen. Nach der Fortify Taxonomie fallen diese Art von Schwachstellen in die Kategorie Environment. Alle untersuchten Projekte bieten für die sichere Konfigurationen Anleitungen und Dokumentationen[14] an.

Das Projekt Drupal stellt allgemeine Praktiken[15] vor, welche den Code in Snippets oder in den eigenen Modulen sicherer machen soll.

Bei den vorgestellten Schwachstellen für Webanwendungen habe ich festgestellt, dass ein Großteil dieser Schwachstellen aufgrund fehlender oder ungenügender Eingabeprüfung ermöglicht wird. Diese Schwachstellenursache war auch bei den Episoden häufig vertreten (11 mal in 19 Episoden, siehe Anhang A.4). Beim CLASP Projekt[16] oder im Online

[12] http://forum.joomla.org/viewtopic.php?f=267&t=70117&st=0&sk=t&sd=a&start=30#p373406 überprüft am 14.08.2008

[13] http://developer.mozilla.org/en/docs/Prism überprüft am 15.08.2008

[14] Wordpress http://codex.wordpress.org/Hardening_WordPress
Joomla http://docs.joomla.org/Joomla_Administrators_Security_Checklist
TWiki http://twiki.org/cgi-bin/view/Codev/SecuringYourTWiki
alle überprüft am 15.08.2008

[15] http://drupal.org/writing-secure-code überprüft am 15.08.2008

[16] http://www.owasp.org/index.php/CLASP_Security_Principles überprüft am 14.08.2008

Buch „Secure Programming for Linux and Unix HOWTO“[17] von David Wheeler sind Praktiken zur Vermeidung von Schwachstellen aufgrund fehlender oder ungenügender Eingabeprüfung zu finden.

## 3.6 Zusammenfassung

Am Anfang dieses Kapitels habe ich die Klassifikation des Verhaltens vorgestellt. Darin sind die Konzepte und Kategorien des Verhaltens hierarchisch angeordnet. Diese Klassifikation soll einen Überblick auf mögliche Verhaltenskonzepte und -kategorien des Behebungsverlaufs geben. Die Verhaltenskategorien werden nochmal zu Hauptkategorien gruppiert, um neu entwickelte Konzepte in die Klassifikation besser einordnen zu können. Darüber hinaus gruppieren sie Verhaltenskategorien, welche interessante Phänomene für weitere Untersuchungen repräsentieren.

Daneben habe ich eine Klassifikation der Rollen dargestellt, wobei ich die drei wesentliche Rollenkategorien *der betroffene/besorgte Nutzer, der externe Helfer* und *der Verantwortliche des Herstellers* identifiziert habe.

Einige Kategorien aus den Klassifikationen habe ich mit Hilfe des paradigmatischen Modells in Beziehung zum Phänomen „Umgang mit Schwachstellenveröffentlichung“ gesetzt. Durch die Verifikation dieser Beziehung anhand der Daten und der Suche nach weiteren Eigenschaften kam ich zu der ausführlichen Beschreibung der Kategorien, deren Eigenschaften und dimensionalen Bandbreite. Auf Grundlage dieser in Beziehung gesetzten Kategorien habe ich die Ausprägung konkreter Eigenschaften in den verschiedenen Episoden untersucht und sie gruppiert, wenn ich Ähnlichkeiten feststellte. Das führte mich zu Mustern, welche den Einfluss auf die Behebungsdauer, den Behebungsverlauf und den Einfluss der verschiedenen Rollen beschreiben.

Neben der Untersuchung des Phänomens „Umgang mit Schwachstellenveröffentlichung“ habe ich Vermutungen und Fragen zum Phänomen „Drängen nach Schwachstellenbehebung“ für spätere Untersuchungen formuliert.

Darüber hinaus habe ich Verhaltensweisen und Praktiken zur Vermeidung von Schwachstellen vorgestellt, welche ich während der Untersuchung der Schwachstellenbehebung gefunden habe.

[17] http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/input.html überprüft am 14.08.2008

# 4 Fazit und Ausblick

Im Abschnitt 4.1 stelle ich Herausforderungen, welche in meiner Arbeit auftraten vor und beschreibe, wie ich damit umgegangen bin. Im danach folgenden Abschnitt 4.2 fasse ich die Ergebnisse zusammen und stelle persönliche Erkenntnisse vor. Zum Schluss gebe ich in Abschnitt 4.3 einen Ausblick darüber, wie weitere Untersuchungen zu diesem Thema aussehen könnten.

## 4.1 Herausforderungen in der Arbeit

Am Anfang der Arbeit wusste ich nicht, was mich aufgrund der Aufgabenstellung erwartet. Ich habe angefangen Daten zu suchen und mir zu überlegen, wie ich die Analyse durchführen sollte, um an die Beschreibung von sicherheitsrelevanten Verhalten zu gelangen. Durch Rückmeldungen aus der Arbeitsgruppe Software Engineering habe ich mich dazu entschlossen die Forschungsmethode Grounded Theory für die Datenanalyse anzuwenden. Gerade das offene Kodieren half mir den Untersuchungsraum besser kennenzulernen und interessante Phänomene zu identifizieren.

Eine weitere Herausforderung, auf die ich gestoßen bin, ergab sich bei der Datenerhebung. Bei der Betrachtung des Projekts Drupal bin ich auf deren Einsatz eines Sicherheitsteams gestoßen, welches für den Umgang mit sicherheitsrelevanten Schwachstellen zuständig ist. Der Behebungsverlauf findet innerhalb dieses Teams statt und wird anscheinend in einer internen Mailingliste besprochen. Erst nach der Behebung veröffentlicht das Sicherheitsteam Informationen zur Schwachstelle. Eine direkte Nachfrage mit der Bitte zur Möglichkeit Zugriff auf einen Teil der älteren Daten in dieser Mailingliste zu erhalten wurde nicht beantwortet. Daraufhin habe ich mir Projekte gesucht, welche genügend Anhaltspunkte für eine qualitative Datenanalyse lieferten.

Nach den ersten Schritten des offenen Kodierens hatte ich Konzepte entwickelt, welche verschiedene Granularitätsebenen abdeckten oder eine Vermischung von Verhaltensweisen und Rollen zu ließen. Durch die Adaption der von Salinger et al. [17] vorgestellten Praktiken und der Erstellung eines Namensschema für Konzepte habe ich diese Probleme in den Griff bekommen.

Die Verhaltens- und die Rollenklassifikation, welche vor ihrer Erstellung als eine Beschreibung von Verhaltensweisen, Rollen und ihren Beziehungen geplant war, stellten sich nach

ihrer Entwicklung als allgemeine Klassifikation für Konzepte und Kategorien heraus. Sie war einfach zu groß und zu allgemein gehalten, um konkrete Beziehungen und Rahmenbedingungen darin festzumachen. Erst durch die Untersuchung konkreter Phänomene bin ich zu einer Beschreibung von Verhaltensweisen, Rollen, sowie ihre Beziehungen, die während der Schwachstellenbehebung auftreten können, gelangt.

Die konkreten Phänomene habe ich schon während der Erstellung der Verhaltensklassifikation identifiziert. Den Zeitpunkt des Übergangs vom offenen Kodieren zum axialen Kodieren festzulegen fiel schwer, da auf der einen Seite durch offenes Kodieren die Verhaltensklassifikation immer hätte weiter entwickelt und neue interessante Phänomene entdeckt werden können, aber auf der anderen Seite würde ich dadurch nicht zu Aussagen über Verhaltensweisen, Rollen und ihren Beziehungen gelangen. Ich entschied mich also nach dem offenen Kodieren der Episoden des ersten Projekts Wordpress und der Anwendung der Konzepte auf Daten der ersten Joomla Episode dafür, das Phänomen „Umgang mit Schwachstellenveröffentlichung“ genauer zu untersuchen.

Als ich mit der Untersuchung interessanter Phänomene angefangen habe, hatte ich zunächst vor mehrere davon zu untersuchen. Ich stellte Vermutungen und Fragen über diese Phänomene auf und begann anschließend mit der Datenanalyse des Phänomens „Umgang mit Schwachstellenveröffentlichung“ in Anlehnung an das axiale Kodieren. Diese Untersuchung beschäftigte mich bis zum Ende der Arbeit, so dass ich die Fragen und Vermutungen zum weiteren Phänomen „Drängen nach Schwachstellenbehebung“ als Ausblick für weitere Untersuchungen darstellen konnte. Weiterhin habe ich Teilaspekte dieses zweiten Phänomens in der Untersuchung zum Phänomen „Umgang mit Schwachstellenveröffentlichung“ betrachtet.

## 4.2 Fazit

Ich habe in meiner Arbeit zum ersten Mal wissenschaftliche qualitative Forschung betrieben. Dafür habe ich mich in die Forschungsmethode Grounded Theory eingearbeitet. Durch diese Methode war es mir möglich aus betrachteten Daten heraus eine Beschreibung zu sicherheitsrelevantem Verhalten in der Open Source Softwareentwicklung zu entwickeln. Diese Beschreibung ist nicht allgemeingültig, bietet aber Aussagen, welche auf Teilaspekte der Schwachstellenbehebung eingehen.

Zuerst habe ich eine Verhaltensklassifikation aufgestellt, welche Konzepte und Kategorien des Verhaltens umfasst und Hauptkategorien zuordnet. Diese Verhaltensklassifikation soll beim Kodieren neuerer Daten helfen, neue Konzepte schneller einordnen und interessante Phänomene, welche noch untersucht werden sollen zu gruppieren. Daneben habe ich eine Klassifikation der Rollen dargestellt, wobei ich die drei wesentliche Rollenkategorien *der betroffene/besorgte Nutzer, der externe Helfer* und *der Verantwortliche des Herstellers* identifiziert habe.

Durch die Untersuchung der Beziehungen zwischen dem Phänomen „Umgang mit Schwachstellenveröffentlichung“ und anderen Verhaltenskategorien, sowie die bestimmten Eigenschaften dieser Kategorien, habe ich Muster gefunden, welche den Einfluss auf die Behebungsdauer, den Behebungsverlauf und den Einfluss der verschiedenen Rollen beschreiben.

Dabei habe ich Verhaltensweisen von Akteuren der verschiedenen Rollen gefunden, welche aktiv eingesetzt wurden und wodurch die Behebungsdauer beeinflusst wurde. Dazu gehören die Veröffentlichung der Schwachstelle ohne Nutzung des vorgegebenen Veröffentlichungskanals, Angabe der Schwachstellenausnutzung, aber auch die Bereitstellung eines Veröffentlichungskanals für Schwachstellen und die Prüfung aller Versionszweige auf die veröffentlichte Schwachstelle. Die Einstufung des Sicherheitsrisikos hat ebenfalls Einfluss auf die Behebungsdauer.

Einfluss auf den Behebungsverlauf, und nicht zwingend auf die Behebungsdauer, haben die Verhaltensweisen Qualitätssicherung in Form von Testen und Review, Drängen und die Rolle des externen Helfers. Es gab aber auch Verhaltensweisen und Eigenschaften der in Beziehung gesetzter Kategorie, welche aufgrund geringen Vorkommens in den Daten keine Aussagen über ihren Einfluss auf den Behebungsverlauf zuließen. Dazu gehören z.B. Angabe von Schwachstellenfolgen, Missverständnisbehandlung oder Unterstützung.

Neben der Untersuchung des Phänomens „Umgang mit Schwachstellenveröffentlichung“ habe ich Vermutungen und Fragen zum Phänomen „Drängen“ für spätere Untersuchungen formuliert und Verhaltensweisen und Praktiken zur Vermeidung von Schwachstellen vorgestellt, welche ich während der Untersuchung der Schwachstellenbehebung gefunden habe.

Während dieser Arbeit habe ich viel über den Begriff Sicherheit gelernt und wie sich Schwachstellen, in Auftreten, Ursache und Folgen äußern. Aus dieser Arbeit heraus nehme ich die Erkenntnis, gewissenhafter mit Sicherheit bei der Benutzung von (Open Source) Webanwendungen umzugehen und bei häufig eingesetzten Anwendungen die zugehörigen Sicherheitsmeldungen im Auge zu behalten. Auch die gefundenen Vermeidungspraktiken und Hinweise, die ich dazu gefundenen habe, werde ich bei zukünftiger Entwicklung von Webanwendungen einsetzen.

Neben der Sicherheit habe ich Verhaltensweisen von Akteuren gesehen, die ich zum einen selber schon erlebt habe und zum anderen habe ich welche kennen gelernt, deren Auftreten mich ernsthaft überrascht haben und deren Ursache zum Teil für mich nicht nachvollziehbar waren.

Bei der Betrachtung der Open Source Projekte habe ich gelernt, welche Werkzeuge zur Entwicklung, Kommunikation und Diskussion eingesetzt werden und wie ich diese benutzen kann, um an bestimmte Informationen zu gelangen. Ich weiß jetzt auf welche Informationen und Anzeichen ich achten sollte, um ein Gefühl für die Sicherheit eines Open Source Projekts zu entwickeln.

## 4.3 Ausblick und weiteres Vorgehen

Um weitere Erkenntnisse über die Verhaltensweisen und Rollen, welche bei der Schwachstellenbehung involviert sind, zu erlangen, könnten folgende Untersuchungen durchgeführt werden.

Zunächst wäre die weitere Untersuchung des Phänomens „Umgang mit Schwachstellenveröffentlichung“ zu nennen. Die Bestimmung der Dauer ist durch die Angabe der Zeitspanne zwischen Veröffentlichung und Behebung schon ein erster Hinweis auf sichere Software, aber er birgt dennoch Probleme, gerade wenn sie in Beziehung zueinander gesetzt werden, um verschiedene Behebungsverläufe oder gar Projekte miteinander zu vergleichen. Witten et al. [23] stellen ein vereinfachtes statistisches Modell vor, welches die Häufigkeit der Schwachstellenentdeckung durch verschiedene Akteure beschreibt. Dadurch ließen sich auch Aussagen über den Vergleich der Sicherheit unterschiedlicher Anwendungen aufstellen.

Daneben könnten auch folgende Eigenschaften bei der Behebung untersucht werden, um den Behebungsverlauf zu bewerten:

- Transparenz
- Nutzerzufriedenheit
- Umfang

Bei diesen drei Faktoren bietet sich der Weg der Betrachtung an, dass dabei nicht untersucht wird, ob dieser Faktor vorhanden ist, sondern geschaut wird, ob die Negation des Faktors vorhanden ist. Konkret bedeutet das:

- Transparenz - Das Fehlen der Transparenz, wird durch das Auftreten bestimmter Verhaltensweisen (ich vermute „Drängen“ dabei als Beispiel) ersichtlich. Eine Untersuchung, welche dimensionale Ausprägungen der in Beziehung gesetzten Kategorien zu einer fehlenden Transparenz während des Behebungsverlaufs führen, würden Aussagen zur Transparenz ermöglichen.
- Nutzerzufriedenheit - Unzufriedenheit wird durch Verhaltensweisen wie Drängen oder durch auftretende Beleidigungen der Akteure sichtbar. Verhaltensweisen nach denen Unzufriedenheit auftritt bzw. welche nach denen keine Unzufriedenheit auftritt, können zu Aussagen über die Nutzerzufriedenheit führen.
- Umfang - Treten nach dem Patch bzw. -release, welcher die Behebung einer Schwachstelle beinhaltet noch weitere Maßnahmen zur Behebung dieser Schwachstelle auf, dann ist dies ein Zeichen dafür, dass die erste Behebung unzureichend war. Die konkreten Schritte und Verhaltensweisen, die in der Behebungsimplemntierung auftraten, können zu weiteren Aussagen über den Behebungsverlauf führen.

Zur weiteren Untersuchung des Phänomens „Umgang mit Schwachstellenveröffentlichung“ könnten zusätzliche Datenquellen beschafft werden, die auch anderen Arten von Open Source Software umfassen, wie z.B. Bibliotheken (Libraries), Betriebssysteme oder Desktop Anwendungen. Auch bei der Erhebung der Daten könnte man neben Dokumenten aus dem Internet auch andere Quellen einbeziehen, z.B. Ergebnisse von Interviews, Umfragen oder Fallstudien.

Diese verschiedenen Arten von Open Source Software könnte man aber auch für das Vorgehen des offenen Kodierens und der Identifizierung weiterer interessanter Phänomene heranziehen. Neben dem untersuchten Phänomen und dem Phänomen „Drängen nach Schwachstellenbehebung“, welches ich schon im Abschnitt 3.4 behandelt habe, könnten weitere Phänomene zur Untersuchung herangezogen werden. Aus meiner Untersuchung haben ich noch „Umgang mit Beleidigungen“, „Missverständnisbehandlung“ und die Hauptkategorie „Unterstützung“ als mögliche interessante Phänomene im Hinterkopf behalten.

Würden genügend Phänomene untersucht werden, könnte damit begonnen werden, eine Theorie bei der Behebung von sicherheitsrelevanten Schwachstelle in Open Source Software mit Hilfe des selektives Kodieren zu bilden. Diese Theorie, aber auch Erkenntnisse über die einzelnen untersuchten Phänomene könnten bei der Schwachstellenbehebung innerhalb der Closed Source Softwareentwicklung betrachtet werden, um gewonnene Erkenntnisse auszuweiten oder abzugrenzen.

# Literaturverzeichnis


[1] William A. Arbaugh, William L. Fithen, und John McHugh. Windows of vulnerability: A case study analysis. *Computer*, 33(12):52–59, 2000. ISSN 0018-9162. doi: http://doi.ieeecomputersociety.org/10.1109/2.889093. 7, 35, 47

[2] Ashish Arora, Ramayya Krishnan, Anand Nandkumar, Rahul Telang, und Yubao Yang. Impact of vulnerability disclosure and patch availability - an empirical analysis. *Third Workshop on the Economics of Information Security*, 2004. URL https://www.dtc.umn.edu/weis2004/telang.pdf. 45, 46, 47, 58

[3] Maria Canellopoulou-Bottis. Disclosing software vulnerabilities. In Kenneth Einar Himma, editor, *Internet Security: Hacking, Counterhacking and Society*, pages 255–268. Jones & Bartlett Publishers, Inc., 1. Ausgabe, 2007. URL http://ssrn.com/abstract=952955. 46

[4] Crispin Cowan. Software security for open-source systems. *Security & Privacy Magazine, IEEE*, 1(1):38–45, 2003. doi: http://dx.doi.org/10.1109/MSECP.2003.1176994. Vorstellung dreier Methoden zur Verbesserung der Sicherheit beim Einsatz von Open Source Software (Systeme) und Tools, die für diese Methoden genutzt werden können Software auditing (Software Revision/Prüfung - Direkt im Code) Vulnerability mitigation (Schwachstellen Verringerung - Fehler zur Laufzeit beheben) Behavior management (Verhaltenssteuerung - Schadensbegrenzung bei Verahlten, welches nicht erwünscht ist). 10, 11

[5] Claudia Eckert. *IT-Sicherheit : Konzepte - Verfahren - Protokolle.* Oldenbourg, München [u.a.], 4. Ausgabe, 2006. ISBN 3-486-20000-3. iii, 2, 3, 4

[6] Richard Ford und Sarah Gordon. When worlds collide: Information sharing for the security and anti-virus communities. *Virus Bulletin Conference.*, 1999. URL http://vx.netlux.org/lib/asg06.html. Überprüft am 27.07.2008. 45

[7] Jaap-Henk Hoepman und Bart Jacobs. Increased security through open source. *Communications of the ACM*, 50(1):79–83, 2007. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1188913.1188921. 11

[8] ISO. Information processing systems - open systems interconnection - basis reference model - part 2: Security architecture, international standards organization. Technical Report 7498-2, ISO, 1989. 2

[9] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX: 5–83, 1883. URL http://www.petitcolas.net/fabien/kerckhoffs/. 10

[10] George Lawton. Open source security: Opportunity or oxymoron? *Computer*, 35(3):18–21, 2002. ISSN 0018-9162. doi: http://doi.ieeecomputersociety.org/10.1109/2.989921. Darstellung von OS Sicherheitssoftware/werkzeuge, Allgemeine Informationen, Vor- und Nachteile von OSS Security Tools, Vorstellung wichtiger OSS Security Tools, Darstellung möglicher Barrieren/Problemen aufgrund der Natur von OSS, Trendabschätzung gestützt durch Zitate/Meinungen von Entwicklern und Sec.-Firmen. 11

[11] Elias Levy. Full disclosure is a necessary evil, 2001. URL http://www.securityfocus.com/news/238. Überprüft am 05.07.2008. 46

[12] Steve Lipner und Michael Howard. The trustworthy computing security development lifecycle. In *Proceedings of the 20th Annual Computer Security Applications Conference 2004 (IEEE ACSAC)*., 2004. URL http://msdn.microsoft.com/en-us/library/ms995349.aspx. Überprüft am 01.08.2008. 8

[13] oisafety.org. Guidelines for security vulnerability reporting and response, September 2004. URL http://www.oisafety.org/guidelines/Guidelines%20for%20Security%20Vulnerability%20Reporting%20and%20Response%20V2.0.pdf. Überprüft am 05.07.2008. 7, 35

[14] OpenSourceInitiative. The open source definition, July 2006. URL http://www.opensource.org/docs/osd. Überprüft am 02.08.2008. 9

[15] Andy Ozment. The likelihood of vulnerability rediscovery and the social utility of vulnerability hunting. *In the proceedings of the Fourth Workshop on the Economics of Information Security (WEIS)*, 2005. URL http://www.andyozment.com/papers/weis05-ozment-vulnrediscovery.pdf. 45

[16] Eric S. Raymond. *The Cathedral and the Bazaar*. O'Reilly & Associates Inc., Sebastopol, CA, USA, 1999. ISBN 1565927249. 10

[17] Stephan Salinger, Laura Plonka, und Lutz Prechelt. A coding scheme development methodology using grounded theory for qualitative analysis of pair programming. In *Proceedings of the 19th Annual Workshop of the Psychology of Programming Interest Group (PPIG '07)*, pages 144–157, Joensuu, Finland, July 2007. University of Joensuu. URL http://www.ppig.org/workshops/19th-programme.html. www.ppig.org. 27, 28, 29, 30, 72

[18] Anselm Strauss und Juliet Corbin. *Grounded Theory: Grundlagen Qualitativer Sozialforschung*. Beltz Psychologie Verlags Union, 1996. 22, 24, 25, 27

[19] Katrina Tsipenyuk, Brian Chess, und Gary McGraw. Seven pernicious kingdoms: A taxonomy of software security errors. *IEEE Security and Privacy*, 3(6):81–84, 2005. ISSN 1540-7993. doi: http://doi.ieeecomputersociety.org/10.1109/MSP.2005.159. 4

[20] John Viega. The myth of open source security, May 2000. URL http://www.developer.com/security/article.php/11580_621851_1. Überprüft am 02.08.2008. 11

[21] H. Wang und C. Wang. Taxonomy of security considerations and software quality. *Communications of the ACM*, 46(6):75–78, 2003. doi: http://doi.acm.org/10.1145/777313.777315. 7

[22] David A. Wheeler. Chapter 2.4: Is open source good for security? *Secure Programming for Linux and Unix HOWTO*, 2003. URL http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/open-source-security.html. 11

[23] Brian Witten, Carl Landwehr, und Michael Caloyannides. Does open source improve system security? *IEEE Software*, 18(5):57–61, 2001. ISSN 0740-7459. doi: http://doi.ieeecomputersociety.org/10.1109/52.951496. 11, 54, 75

# A Anhang

## A.1 Konzepte

Die folgende Tabelle enthält die Liste aller von mir entwickelten Konzepte, ein Textbeispiel, welches diese Konzept beschreibt und wie häufig das Konzept innerhalb meiner kodierten Episoden auftritt.

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 1 | ändert_ Ticketbeschreibung | * priority changed from high to highest omg bbq.<br>* summary changed from Apply [5570] int cast to 2.2 branch to 2.2 remote SQL injection exploit, user registration, xmlrpc.php..<br>* description changed.<br>* severity changed from major to blocker. | 7 |
| 2 | beantwortet_ Verständnisfrage | > If I read this correctly - isn't the exploit only viable if you have a valid username/password combo to use as there is a login check.<br>Yes, you are correct, you must have at least one valid user/pass combo. | 3 |
| 3 | bedanktSichBei_ Berichterstatter(n) | Many thanks to Sumit Siddharth and Alex Concha for their help with reporting issues in this release. | 3 |
| 4 | bedanktSichBei_ Hilfeleistendem/n | Otto42 - thankyou for your help and advice also! | 9 |
| 5 | begründet_ Eingrenzung | It says as much in the exploit code (after running it through Google Translate). | 2 |
| 6 | begründet_ Massnahme | I think thats a great idea, given the current rash of troubles. | 3 |
| 7 | begründet_ Motivation | it's my job/fun to find bugs... | 1 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 8 | begründet_ Nachfrage | It is hard to work out what is happening here without knowing where the problem is coming from. | 1 |
| 9 | begründet_ Ursache | Manipulation of $_SERVER['REQUEST_URI'] has led to many xss flaws. | 1 |
| 10 | begründet_ Ursache_ unzulässig | As such, this function does exactly what it is supposed to do. | 4 |
| 11 | begründet_ Verhalten | Ah, heck. I think this bug is being actively exploited, so I may as well, so you can protect yourself. Here's the post I sent to the xmlrpc list.<br>I'm a little hesitant to post this here as it's a publically available list, but I think I just found a security hole in xmlrpc that is being actively exploited, and as such, may as well sound the alarm. | 6 |
| 12 | bekräftigt_ Erfolgslosigkeit_ Lösungsfindung | but a wee alter and the problem still remains | 1 |
| 13 | bekräftigt_ Erfolgslosigkeit_ Ursachensuche | I've just done a little digging in my logs but haven't spotted anything yet. I'll keep looking. | 3 |
| 14 | belehrt_ Diskussionspartner | In the future you shouldn't attack someone who is trying to help. This is a complex and irregular issue so I totally understand why it was difficult to see. In the future you shouldn't dismiss something you do not understand, instead i urge you to ask questions and learn more. | 2 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 15 | berichtet_ Problem | Hi,<br>I've searched around for a resolution to my problem but the closet thread I can find is this: <http://wordpress.org/support/topic/89912?replies=4><br>Basically about a week ago my site began experiencing problems whenever I tried to access the home page. The screen just freezes for about 10 minutes..sometimes it also throws me out (closes the browser). In the browser footer it displays the following:<br>waiting for <http://xx.xx.xx.xx./iframe/wp-stats.php><br>(the 'x' is an IP address which I don't recognise)<br>At first I suspected that it was a problem with the wp-stats plugin which I had just installed prior to this problem surfacing. So I removed the plugin (and other plugins)..I also tried other themes and browsers, but a wee alter and the problem still remains.<br>So I contacted my host (as one of the threads here suggested I do) and they have reported to me the following:<br>"Your site was most likely injected with a 1px iframe due to a vulnerability in WordPress – which is why 2.2.3 was rushed out and pushed out to everyone. A number of sites have the same link which leads one to believe it was due to an exploit in either Wordpress itself or the theme you're using (which has also been called into question as of late)."<br>So now i'm wondering whether anyone can corroborate that this is the likely reason..and whether they is anything I can do to resolve the problem. I would of course like to upgrade to 2.3 asap, but I doubt this will solve the issue in itself..or will it?<br>Any advise would be much appreciated. | 2 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 16 | berichtet_ Problem_ erneut | Hi all,<br>Apparently there is some disagreement about this issue. I am providing more information to build a greater understanding about what is happening. This problem is entirely contained within the query.php file. At the comment header of query.php it says: "The Big Query." Yes indeed this file produces a large query. This file is very disorganized and it was difficult to go though with a fine tooth comb, but I did and i found a flaw because of it. I was looking for SQL Injection, but broken access control will get me a CVE number. Perhaps this URL provides more information: http://localhost/wordpress/index.php/'wp-admin/ <http://localhost/wordpress/index.php/%27wp-admin/><br>I urge everyone to make this get request and to print the $_SERVER['REQUEST_URI'] and $_SERVER['PHP_SELF'] variables. You will see that wp-admin/ is at the end of these variables.<br>I should have provided the exact point in which the flawed query is being built. I thought that my PoC was enough, my bad. [..]<br>This url: htttp://localhost/wordpress/index.php/'wp-admin/ will cause the is_admin() function to return TRUE. [..]<br>Peace | 1 |
| 17 | berichtigt_ Berichterstatter | The is_admin() function is not supposed to tell whether a user is an administrator or not, it tells whether the user is looking at one of the administration pages. As such, this function does exactly what it is supposed to do.<br>As for the rest, there is no flaw. To view a draft, the user must authenticate and have the correct capability set. There is no way to view drafts without being logged in and having that capability set on the user's role level. This "vulnerability" is non-existent. | 1 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 18 | berichtigt_ Diskussionspartner | Hyper-Links: <contradicts> 3:81 I can see a problem at AzzQim'.. [Content for linked quote "3:81"] I can see a problem at AzzQim's website. The permission on 'themes' folder is possibly set to 755, which makes folder content viewable and downloadable anyone. I see 'almost-spring' folder, fh-freedom-blue-01.zip and so forth. <contradicts> 3:83 'wp-includes' folder is open, .. [Content for linked quote "3:83"] 'wp-includes' folder is open, too. macsoft, you dont have much of a grip on permissions. Directories have to be 755 for the files inside of them to be accessed by the web server. Ive stated this in another thread you made waves about this in, but I will state it again here – there is NO extra security risk with directories being chmod 755. Furthermore, that a given directory is browsable, merely means that Options All -Indexes has not been enabled in one's .htacces AND that there is no index file in the given directory. | 1 |
| 19 | beschreibt_ Patchinhalt | Attached patch replaces is_admin() check with current_user_can('level_10'). Perhaps we could explicitly check the user's capabilities but I wasn't sure from the documentation which capabilities we should be looking at. Instead I've just checked if the user is the administrator or not. | 5 |
| 20 | beschreibt_ Programmverhalten_ ungewöhnliches | Basically about a week ago my site began experiencing problems whenever I tried to access the home page. The screen just freezes for about 10 minutes..sometimes it also throws me out (closes the browser). | 1 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 21 | beschreibt_ Schadensbegrenzung | - ALWAYS click LOGOUT in Joomla Admin when you finish<br>- NEVER browse other websites while logged in to Joomla Admin<br>- If you allow users to upload/modify your site (E.g. Community Builder) then don't browse/or limit your surfing of your own site while logged in to Joomla Admin<br>- NEVER click on links to "Upgrade this component" in 3rd Party Components or even the Joomla Core check<br>- NEVER browse forums while logged into Joomla Admin | 4 |
| 22 | beschreibt_ Schwachstelle | 2. SQL Injection:<br>This is only exploitable by authenticated users.<br>The post_id parameter is not properly sanitized before passing its value to the backend database which results in a Sql injection. Exploiting this is pretty trivial. As, it is an integer based injection, it works irrespective of the setting "magic quote". I wrote a Simple Proof Of Concept for this.<br>Download Exploit<br>———————<br>Successful Exploitation of this will give you usernames and md5 hash of password of all users including admin user. Before you run mdcrack on this hash, read my previous post on wordpress cookies as this will save your time.<br>Once you have the admin user hash needless to say you can create a php backdoor and that essentialy is game over.<br>About Poc:<br>The poc demonstrates how critical SQL injection vulnerabilities can be. In this example,the poc goes beyond obtaining admin hashes. It also returns the username and encrypted password of the mysql user(s). If the database is running as privilidged user, this will also try to fetch the /etc/passwd file, or any other file for that matter. As this injection is in an integer field it works irrespective of the setting magic quote | 6 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 23 | beschreibt_ Schwachstellenausnutzung | 1. Create a draft post 2. Log out 3. Visit <http://yourblog.com/index.php/wp-admin/> * is_admin() spots the wp-admin in the request and returns true * query.php uses is_admin() to decide to return future, draft or pending posts 4. Future, draft and pending posts are displayed. | 6 |
| 24 | bestätigt_ Motivation[1] | | 0 |
| 25 | bestätigt_ Problem | Damn, I am having same problem. Just realized this in my source code: <!– Traffic Statistics –> <iframe src=http://www.wp-stats-php.info/iframe/wp-stats.php width=1 height=1 frameborder=0></iframe> <!– End Traffic Statistics –> | 18 |
| 26 | bestätigt_ Problem_ Fremdquellen | Although I am still unable to reproduce the problem on any sort of setup, a few other people have claimed to make it work as well | 1 |
| 27 | bestätigt_ Problem_ nicht | Not found the source yet, or confirmed that it's a current problem so there's no fix. | 2 |
| 28 | bestätigt_ Schwachstelle | tested this, and this does work. however my tester says your code is incorrect. the other notable thing is that the edited posts are changed to drafts. | 6 |
| 29 | bestätigt_ Schwachstelle_ nicht | There is no way to view drafts without being logged in and having that capability set on the user's role level. | 1 |
| 30 | bewertet_ Lösung_ mögliche | whoami, your fix does not. I would rather not have people think they're safe and really not be | 6 |
| 31 | bewertet_ Lösung_ positiv | Looks good to me. | 2 |

[1] Es gibt in der Verhaltenskategorie „Motivation“ (siehe Anlage A.2) noch die zwei Konzepte „vermutet_Motivation“ und „begründet_ Motivation“, welche die Annahmen zulassen, dass die vermutete Motivation bestätigt werden kann bzw. einer Begründung eine Bestätigung vorausgeht.

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 32 | bewertet_ Lösung_ unvollständig | I don't think the current patch addresses all the issues. | 3 |
| 33 | bewertet_ Schadensbegrenzung | Keith, bad news for you: the guys doing this sort of thing use proxies and servers located in the US to cause the infection - the destination of the traffic is china (and then actually Russia... but that a longer story). So you can end up with these things all over your blog even if you have all of the eastern block and communist world blocked. | 1 |
| 34 | bewertet_ Schwach-stellenmeldung | There are many (most) around here who are more knowledgeable than I about the inner workings of Joomla, but could this be a hoax? It sure looks like something's fishy to me... | 1 |
| 35 | bewertet_ Schweregrad | Severity: HIGH - Attack needs no authentication. | 4 |
| 36 | bewertet_ Situation | At least we're now on equal footing and know how to work around it.<br>These kinds of things are always tricky. | 1 |
| 37 | bietet_ Details_ keine | I didn't find any reference to 1.x. | 3 |
| 38 | bietet_ Details_ unmöglich | No mysql logs, I can't access to them on my server :( | 5 |
| 39 | bietet_ Details_ weitere | Plugins:<br>Edit Comments 0.3 beta<br>Filosofo Comments Preview 0.7<br>Spam Karma 2 2.2 r3<br>Theme:<br>extensively hacked version of kubrick | 30 |
| 40 | bietet_ Hilfe_ weitere | Anyone that is interested in knowing how they too, can join the experiment, and get the code, can email me at whoo AT whoo.org | 1 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 41 | bittetUm_ Hilfeleistung | Any advise would be much appreciated. | 4 |
| 42 | dementiert_ Schwachstelle | This "vulnerability" is non-existent. | 2 |
| 43 | drängt_ | Any solution yet!? | 4 |
| 44 | drängt_ Releaseersteller | IF THIS is true, this should be patched for 1.xx immediately. | 2 |
| 45 | empfiehlt_ Update | Upgrade to the current latest WordPress version (2.2.3) | 2 |
| 46 | empfiehlt_ Update_ dringlich | This update is highly recommend for all users of both branches. | 2 |
| 47 | empfiehlt_ Vorgehen_ weiteres | Please open a new ticket with causal details, if you experience a similar issue. | 3 |
| 48 | entschuldigtSichBei_ Hersteller | UPDATE: I apologize to the wordpress guys for handling it how I did. This is my first security bug I found and it kind of caught me off guard. Please rest assured I meant no harm. Next time (I hope there isn't one) I'll follow the right process. | 1 |
| 49 | erhofftSich_ Erkenntnis_ zusätzliche | While I hope to learn something, I might not since mod_security seems to catch everything that is malicious, but you never know. | 1 |
| 50 | erhofftSich_ Hilfe | I hope Wordpress will do more for security issues. | 1 |
| 51 | eröffnet_ ticket | Opened a ticket: <http://trac.wordpress.org/ticket/5313> | 3 |
| 52 | eröffnet_ ticket_ wieder | § status changed from closed to reopened. | 2 |
| 53 | fässtZusammen_ Lösung | So a patch has been created and applied to eliminate the dependancy on "wp-admin/". | 1 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 54 | fässtZusammen_ Releaseinhalt | These releases include fixes for several publicly known minor XSS issues, one major XML-RPC issue, and a proactive full sweep of the WordPress codebase to protect against future problems. | 3 |
| 55 | fragtNach_ Details | Are you able to determine when that code was inserted? | 7 |
| 56 | fragtNach_ Details_ direkt | If someone has mysql logs hanging around (binary or otherwise) it may be helpful if they are able to pick out when the code was inserted into the database. Something along the lines of<br># mysqlbinlog mysql-bin.* \| grep iframe<br>would do the trick. | 9 |
| 57 | fragtNach_ Erfolgsaussicht | I would of course like to upgrade to 2.3 asap, but I doubt this will solve the issue in itself..or will it? | 1 |
| 58 | fragtNach_ Hinweisfundorte_ mögliche | In the interim, is there a way of determining when your particular wordpress installation was upgraded? Maybe looking at file dates? Or do database entries contain a version stamp of the WP that wrote it, etc? | 1 |
| 59 | fragtNach_ Lösung_ endgültige | When can we expect a patch? | 2 |
| 60 | fragtNach_ Schwachstellenursache | Is there any definitive answer yet as to how/why/only 2.2 was vulnerable, etc? | 2 |
| 61 | fragtNach_ Verständnisklärung | If I read this correctly - isn't the exploit only viable if you have a valid username/password combo to use as there is a login check. | 5 |
| 62 | gibtAn_ Behebungszeitpunkt | Fixed over a week ago. | 1 |
| 63 | gibtAn_ Exploitfundort(e) | Note: Exploit code for this (fixed) bug is in the wild: http://wordpress.org/support/topic/120857 | 3 |
| 64 | gibtAn_ Herstellerbe-nachrichtigung_ Zeitpunkt | 1. vendor notified on 22nd March 2007. | 1 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 65 | gibtAn_ Herstellerreaktion_ ausgebliebene | I have personally asked 3 core team members to comment in this thread, up to now, not one of them has. | 1 |
| 66 | gibtAn_ Herstellerreaktion_ chronologisch | 1. vendor notified on 22nd March 2007.<br>2. New Version released on 2nd April 2007.<br>3. Advisory released on 2nd April 2007 | 1 |
| 67 | gibtAn_ Lösung | The bug tracking and patch for this issue can be found here: <http://trac.wordpress.org/ticket/5487> | 1 |
| 68 | gibtAn_ Schwachstellenausnutzer_ mögliche | Inserting an iframe of that style is the common injection by at least one black hat seo ring | 1 |
| 69 | gibtAn_ Schwachstellenfolge(n)_ mögliche | WARNING: This issue may lead to a full blog compromise. | 12 |
| 70 | gibtAn_ Schwachstellenfolge(n)_ reale | One of them, Maomium <http://www.maomium.com>, was hacked last night. Someone created a user account on it then stole my admin identity to post content. | 5 |
| 71 | gibtAn_ Verbesserungsvorschlag | Fix should be a putiing in a missing attribute_escape instead of the given regex (in my opinion). | 2 |
| 72 | gibtAn_ Workaround | If anyone is scared and wants a fix NOW, they should either turn off registration (which is off by default) or delete xmlrpc.php. | 11 |
| 73 | gibtAn_ Zwischenmeldung | I'd just update the install as you're a few versions behind. It's up to 2.3.1 now. | 4 |
| 74 | grenztEin_ Schwachstellenvoraussetzung | This is only exploitable by authenticated users. | 1 |
| 75 | grenztEin_ Schwachstellenvoraussetzung_ unbegrenzt | This flaw works regardless of register_globas or magic_quotes_gpc. | 6 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 76 | klärtAuf_ Missverständnis | I'm sorry, that wasn't clear. What I meant: since I wrote the post Nov 26 and didn't discover the injected iframe until Dec 10, the access to admin-ajax.php could have been involved. | 4 |
| 77 | kündigtAn_ Release | there is a release coming shortly anyway. | 5 |
| 78 | lenktEinZum_ Diskussionsfrieden | Nobody was attacking you. Calm down and try to be a bit more professional, please. | 1 |
| 79 | listetAuf_ Lösungsversuche | At first I suspected that it was a problem with the wp-stats plugin which I had just installed prior to this problem surfacing. So I removed the plugin (and other plugins)..I also tried other themes and browsers | 2 |
| 80 | listetAuf_ Sicherheitsmaßnahmen | Step 1: Find where the code is being inserted. From what whooami is saying, it's likely inside the content of one specific post. So look through that post and find and remove it.<br>Step 2: Upgrade to the current latest WordPress version (2.2.3). This has no known security issues at this time.<br>Step 3: Keep up to date on WordPress releases. On the main dashboard, you'll always see new release information. Also, in WordPress 2.3 and up, WordPress itself will start telling you when your version is out of date and give you info on how to upgrade. So that will be good. | 1 |
| 81 | lobt_ Hilfeleistenden | @duskglow: excellent work isolating the problem and identifying an instant workaround/fix. | 1 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 82 | offenbart_ Schwachstelle(n) | 7. Wordpress /upload.php XSS Vulnerability<br>Severity: HIGH - Attack needs no authentication.<br>Affected Wordpress Version: 2.2.1 - the latest version (This is a 0day Vulnerability <http://en.wikipedia.org/wiki/Zero_day>)<br>WARNING: This issue may lead to a full blog compromise.<br>In /upload.php the parameter style is prone to XSS Attacks when editing Temporary Uploads (they usually have a negative ID). An attack could look like this:<br>/upload.php?style=%22%3E{shellcode}&tab=upload&post_id=-1337<br>As you can see this is a very basic reflective XSS vulnerability <http://en.wikipedia.org/wiki/Cross-site_scripting>. This shouldn't happen within a Web Application. | 9 |
| 83 | offenbart_ Schwachstellenursachefund_ möglichen | Actually, I think I just found the exploit. I don't want to publish it here. Can someone from wordpress please tell me where to post the info? | 1 |
| 84 | schildert_ Funktionsweise | These blocks of code relate to the checking of a token that is held/generated by the session object. This token is passed around by forms and checked before operations | 8 |
| 85 | schildert_ Reproduktion | *Logged into wp-admin with Firefox.<br>*Created a new post called "DRAFT", with text "DRAFT"<br>*I saved it (but did not publish it)<br>*I opened another browser (Opera).<br>*I tried using the URL you had above (modified for my site) and it does not show me drafts.<br>*I tried adding the p=<post number> get argument, but I just get a blank page. | 2 |
| 86 | schildert_ Schwachstellenursache | The post_id parameter is not properly sanitized before passing its value to the backend database which results in a Sql injection. | 8 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 87 | schildert_ Symptombekämpfung | Step 1: Find where the code is being inserted. From what whooami is saying, it's likely inside the content of one specific post. So look through that post and find and remove it. | 1 |
| 88 | schildert_ Symptom-bekämpfung_ erfolgreich | I have now removed it from that post. | 3 |
| 89 | schließt_ Ticket | § status changed from assigned to closed.<br>§ resolution set to fixed. | 7 |
| 90 | spezifiziert_ Version | Affected Wordpress Version: 2.2.1 - the latest version | 27 |
| 91 | spezifiziert_ Version_ nicht | but I can't tell if this came from the currently installed WordPress </wiki/WordPress> version has been around since an older install. | 1 |
| 92 | stelltAuf_ Vermeidungsregel | Although an attacher shouldn't be able to control all $_SERVER variables, none of them should be trusted. | 1 |
| 93 | stelltBereit_ Changelog | So I looked further in this forum and found <http://www.joomla.org/content/view/4335/116/><br>which shows the 1.5.0 Changelog:<br>here are 3 entries that refer to the problem;<br>10-Dec-2007 Laurens Vandeput ** Bug Squash Event: Brussels<br>* SECURITY A5 [HIGH] Critical CSRF allow portal compromise - Administrator components. Thanks to Paul Delbar & Jeroen Loose.<br>09-Dec-2007 Rob Schley ** Bug Squash Event: SF **<br>* SECURITY A5 [HIGH] [#8361] Critical CSRF allow portal compromise. Administrator components<br>09-Dec-2007 Andrew Eddie ** Bug Squash Event from home **<br>* SECURITY A5 [HIGH] [#8361] Critical CSRF allow portal compromise - admin com_users only | 1 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 94 | stelltBereit_ Exploit | exploit: htttp://localhost/wordpress/'wp-admin/ This will cause both $_SERVER['REQUEST_URI'] and $_SERVER['PHP_SELF'] to contain the value: htttp://localhost/wordpress/'wp-admin/ Vulnerable function: line 34, in ./wp-includes/query.php. function is_admin () { global $wp_query; return ($wp_query->is_admin \|\| (strpos($_SERVER['REQUEST_URI'], 'wp-admin/') !== false)); | 9 |
| 95 | stelltBereit_ Patch | § attachment 5487.patch added. | 16 |
| 96 | stelltBereit_ Release | WordPress 2.2.1 is now available <http://wordpress.org/download/> | 4 |
| 97 | stelltBereit_ Schwachstellenveröffentlichung | Hi, checking in the securityfocus, see one bug for joomla, can you see here: http://www.securityfocus.com/archive/1/485676/30/0/threaded | 4 |
| 98 | stelltBereit_ TicketURL | http://joomlacode.org/gf/project/joomla/tracker/?action=TrackerItemEdit&tracker_item_id=8361 will give you the full chronology of the report and conclusion; | 3 |
| 99 | stelltFest_ Missverständnis | Apparently there is some disagreement about this issue | 1 |
| 100 | vermutet_ Motivation | who knows, it would almost be interesting to make up a site that forges a referer thats a wp blog and see if anything can be figured out. | 1 |
| 101 | vermutet_ Problem_ Ursache | The injection is 100% certainly via XMLRPC, because the posts are getting their character set maimed as well on the repost (they are using UTF-8, but I have some blogs set not to use it). | 10 |

| Id | Konzept | Beispielquote | Vorkommen |
|---|---|---|---|
| 102 | vermutet_ Ursache_ andere | An exploit in WordPress is not the only way this sort of thing can happen.<br>For example, if they ever gained read access to your wp-config.php file (and you didn't notice and change the password), then they have your database info and can simply insert the content directly, no need to go through your website at all. | 3 |
| 103 | vermutet_ Ursache_ weitere | I'm actually thinking that there are two different exploits, but I haven't found the other. I think that this is the exploit they used to actually change the content of the post, and I think there is another that they used to set it to "publish".<br>That's why I think there were two separate xmlrpc calls. | 2 |
| 104 | verspricht_ Detailbeschaffung | I'm checking with my shared hosting provider if I can get access to the mysql logs. | 1 |
| 105 | verspricht_ Ratbefolgung | I'm going to do as you advice and upgrade asap. | 1 |
| 106 | verspricht_ Symptombekämpfung | Now I gotta go back and dig them out... | 1 |
| 107 | verspricht_ Überwachung | I will be watching closely for any additional activity. | 3 |
| 108 | weist_ Richtung | So I think all we need is a proper is_admin() check and not any cap checks. | 3 |
| 109 | zitiert_ Diskussionspartner | Replying to thee17 </ticket/5313>:<br>Because the method of exploiting this was posted, this needs fixed and posibly fast. | 6 |
| 110 | zitiert_ Hilfeleistenden | Copying DD32's forum comment here:<br>Can anyone take a read through their webservers access logs and look for anything suspect accessing the admin pages? Also check for other users, and change the admin passwords. It is hard to work out what is happening here without knowing where the problem is coming from. | 2 |

## A.2 Konzept-/Kategorieklassifikation als hierarchische Liste

Die Folgende Liste stellt auf der obersten Ebene (arabische Ziffern) die Hauptkategorien dar. Auf der darunter folgenden Ebene (kleine Buchstaben) sind die zu der Hauptkategorie gehörenden Verhaltenskategorien. Auf der dritten Ebene (römische Ziffern) befinden sich die Verhaltenskonzepte.

1. Informationsbereitstellung
    a) Funktionsweisendarstellung
        i. beschreibt_Patchinhalt
        ii. schildert_Funktionsweise
    b) Herstellerinformationenbereitstellung
        i. fässtZusammen_Releaseinhalt
        ii. gibtAn_Behebungszeitpunkt
        iii. gibtAn_Herstellerbenachrichtigung_Zeitpunkt
        iv. gibtAn_Herstellerreaktion_ausgebliebene
        v. gibtAn_Herstellerreaktion_chronologisch
        vi. kündigtAn_Release
    c) Problemdarstellung
        i. berichtet_Problem
        ii. berichtet_Problem_erneut
        iii. beschreibt_Programmverhalten_ungewöhnliches
    d) Schwachstellendarstellung
        i. begründet_Eingrenzung
        ii. beschreibt_Schwachstelle
        iii. beschreibt_Schwachstellenausnutzung
        iv. gibtAn_Exploitfundort(e)
        v. gibtAn_Schwachstellenausnutzer_mögliche
        vi. gibtAn_Schwachstellenfolge(n)_mögliche
        vii. gibtAn_Schwachstellenfolge(n)_reale
        viii. grenztEin_Schwachstellenvoraussetzung

ix. grenztEin_Schwachstellenvoraussetzung_unbegrenzt
x. schildert_Reproduktion
xi. schildert_Schwachstellenursache
xii. spezifiziert_Version
xiii. spezifiziert_Version_nicht
xiv. stelltBereit_Exploit
xv. stelltBereit_Schwachstellenveröffentlichung

e) Zwischenstatusbereitstellung
   i. gibtAn_Zwischenmeldung

2. Informationsverwertung/-verarbeitung
   a) Informationsbestätigung
      i. bestätigt_Problem
      ii. bestätigt_Problem_Fremdquellen
      iii. bestätigt_Problem_nicht
      iv. bestätigt_Schwachstelle
      v. bestätigt_Schwachstelle_nicht
      vi. erhofftSich_Erkenntnis_zusätzliche
   b) Informationsbewertung
      i. bewertet_Schadensbegrenzung
      ii. bewertet_Schwachstellenmeldung
      iii. bewertet_Schweregrad
      iv. fragtNach_Erfolgsaussicht
   c) Informationskorrektur
      i. berichtigt_Berichterstatter
      ii. berichtigt_Diskussionspartner
      iii. dementiert_Schwachstelle
   d) Review
      i. bewertet_Lösung_mögliche

ii. bewertet_Lösung_positiv
iii. bewertet_Lösung_unvollständig
iv. gibtAn_Verbesserungsvorschlag
v. weist_Richtung

e) Ursachenidentifikation
i. begründet_Ursache
ii. begründet_Ursache_unzulässig
iii. fragtNach_Schwachstellenursache
iv. vermutet_Problem_Ursache
v. vermutet_Ursache_andere
vi. vermutet_Ursache_weitere

f) Verständnisklärung
i. beantwortet_Verständnisfrage
ii. fragtNach_Verständnisklärung

3. Einleitung konkreter Maßnahmen

a) konkreten Schutz liefern
i. begründet_Massnahme
ii. beschreibt_Schadensbegrenzung
iii. gibtAn_Workaround
iv. listetAuf_Sicherheitsmaßnahmen
v. schildert_Symptombekämpfung
vi. schildert_Symptombekämpfung_erfolgreich
vii. stelltAuf_Vermeidungsregel
viii. stelltBereit_Patch
ix. stelltBereit_Release

b) Lösungs_Informationsbereitstellung
i. fässtZusammen_Lösung
ii. fragtNach_Lösung_endgültige

iii. gibtAn_Lösung

iv. listetAuf_Lösungsversuche

4. Umgang mit Konflikten

   a) Drängen nach Lösung

      i. drängt_

      ii. drängt_Releaseersteller

   b) Missverständnisbehandlung

      i. klärtAuf_Missverständnis

      ii. stelltFest_Missverständnis

   c) Umgang mit Beleidigungen

      i. belehrt_Diskussionspartner

      ii. entschuldigtSichBei_Hersteller

      iii. lenktEinZum_Diskussionsfrieden

   d) Umgang mit voller Schwachstellenveröffentlichung

      i. begründet_Verhalten

      ii. bewertet_Situation

      iii. offenbart_Schwachstelle(n)

      iv. offenbart_Schwachstellenursachefund_möglichen

5. Unterstützung

   a) Empfehlung

      i. empfiehlt_Update

      ii. empfiehlt_Update_dringlich

      iii. verspricht_Ratbefolgung

      iv. verspricht_Symptombekämpfung

      v. verspricht_Überwachung

   b) Motivation

      i. begründet_Motivation

      ii. bestätigt_Motivation

iii. vermutet_Motivation

c) Unterstützung erbeten

i. begründet_Nachfrage

ii. bittetUm_Hilfeleistung

iii. erhofftSich_Hilfe

iv. fragtNach_Details

v. fragtNach_Details_direkt

vi. fragtNach_Hinweisfundorte_mögliche

d) Unterstützung geben

i. bedanktSichBei_Berichterstatter(n)

ii. bedanktSichBei_Hilfeleistendem/n

iii. bekräftigt_Erfolgslosigkeit_Lösungsfindung

iv. bekräftigt_Erfolgslosigkeit_Ursachensuche

v. bietet_Details_keine

vi. bietet_Details_unmöglich

vii. bietet_Details_weitere

viii. bietet_Hilfe_weitere

ix. lobt_Hilfeleistenden

x. verspricht_Detailbeschaffung

6. Einsatz von Werkzeugen

a) Bugtracker

i. ändert_Ticketbeschreibung

ii. empfiehlt_Vorgehen_weiteres

iii. eröffnet_ticket

iv. eröffnet_ticket_wieder

v. schließt_Ticket

vi. stelltBereit_TicketURL

b) Kommunikationswerkzeug

i. zitiert_Diskussionspartner

ii. zitiert_Hilfeleistenden

c) Versionsverwaltung

i. stelltBereit_Changelog

## A.3 Rollenklassifikation

### A.3.1 Rollenkonzepte

In der folgenden Tabelle sind die verschiedenen Rollenkonzepte aufgeführt und beschrieben. Darüber hinaus ist die Herkunft des Konzepts angegeben (aus dem Text kodiert, aus den Metadaten, als Eigenschaft gefunden, oder von einem Verhaltenskonzept abgleitet).

| Konzeptname | Beschreibung | Herkunft |
|---|---|---|
| Rolle: Beitragender | Diese Rolle trägt ab und zu zur Entwicklung der Open Source Software bei. | Textkodierung |
| Rolle: Dokumentator | Rolle auf Seiten des Herstellers, welche sich vorrangig um die Entwicklung der Dokumentation kümmert. | Metadaten |
| Rolle: Ehemaliger | Akteure dieser Rolle gehörten mal zum Team des Herstellers. | Metadaten |
| Rolle: Externer | Gehört zum Team des Herstellers. | Eigenschaft |
| Rolle: Forumsmoderator | Kümmern sich um die Moderation der Kommunikationswerkzeuge, die von dem Open Source Projekt eingesetzt werden. | Metadaten |
| Rolle: Hilfeleistender | Bietet bei auftretenden Problemen seine Hilfe an. | Verhaltenskonzept („bietet_Hilfe_weitere“, siehe Anhang A.1 ID 40) |
| Rolle: Interner | Gehört nicht zum Team des Herstellers. | Eigenschaft |

| Konzeptname | Beschreibung | Herkunft |
|---|---|---|
| Rolle: Kernteammitglied | Die Entwickler auf Seiten des Herstellers. | Metadaten |
| Rolle: OSS Benutzer | Benutzt die Open Source Software, welche das Projekt anbietet | Textkodierung |
| Rolle: Problem-Berichterstatter | Berichtet von Problemen, welche bei der Benutzung der Open Source Software auftreten, welche das Projekt anbietet. | Verhaltenskonzept („berichtet_Problem“, siehe Anhang A.1 ID 15) |
| Rolle: Problemlöser | Löst die auftretenden Probleme. | Textkodierung |
| Rolle: Releaseankündiger | Kündigt an und beschreibt den Inhalt eines Release. | Verhaltenskonzept („kündigtAn_ Release“, siehe Anhang A.1 ID 77) |
| Rolle: Schwachstellen-Berichterstatter | Berichtet von Schwachstellen, welche bei der Benutzung der Open Source Software auftreten, welche das Projekt anbietet. | Verhaltenskonzept („offenbart_Schwachstelle(n)“, siehe Anhang A.1 ID 82) |
| Rolle: Supporter | Diese Rolle trägt intensiv zur Entwicklung der Open Source Software bei. | Metadaten |

### A.3.2 Rollenkategorien

In der folgenden Liste werden die Rollenkategorien und zugehörige Verhaltenskategorien, ihre Eigenschaft (Externer, Interner) zugeordnete Rollenkonzepte dargestellt.

1. der betroffene/besorgte Nutzer
   - Verhaltenskategorien:
     - Unterstützung erbeten, Problemdarstellung, Drängen nach Lösung
   - Eigenschaften
     - Extern

- Rollenkonzepte
  - Problem-Berichterstatter, OSS Benutzer

2. der externe Helfer
   - Verhaltenskategorien:
     - Unterstützung geben, konkreten Schutz liefern, Umgang mit voller Schwachstellenveröffentlichung, Schwachstellendarstellung, Drängen nach Lösung
   - Eigenschaften
     - Extern
   - Rollenkonzepte
     - Supporter, Ehemaliger, Schwachstellen-Berichterstatter, Hilfeleistender, Problemlöser, Beitragender
3. der Verantwortliche des Herstellers
   - Verhaltenskategorien:
     - Unterstützung geben, konkreten Schutz liefern, Umgang mit voller Schwachstellenveröffentlichung, Herstellerinformationenbereitstellung, Lösungs_ Informationsbereitstellung, Review, Funktionsweisendarstellung
   - Eigenschaften:
     - Intern
   - Rollenkonzepte:
     - Kernteamentwickler, Dokumentatoren, Forumsmoderator, Releaseankündiger, Schwachstellen-Berichterstatter, Hilfeleistender, Problemlöser

## A.4 Beschreibung der Episoden

Im Folgenden ist eine Kurzübersicht über die Episoden der einzelnen Projekte zu finden. Dabei wird kurz der Behebungsverlauf der Episode umrissen, die Einstufung der Schwachstelle und die Schwachstellenart genannt, sowie die URL der zu den Episode gehörenden Daten (inklusive Dokument ID) aufgeführt. Ich habe alle Links zu den Dokumenten am 15.08.2008 überprüft und nicht mehr existierende Quellen entsprechend markiert.

### A.4.1 Wordpress

#### Episode 1

#### Verlauf

- Hersteller am 22.03.07 benachrichtigt
- Behebung am 02.04.07 veröffentlicht

**Einstufung** Hoch

**Schwachstellenart** SQL Injection

#### Dokumente

**10:** http://www.notsosecure.com/folder2/2007/04/03/wordpress-212-xmlrpc-security-issues/

**11:** http://wordpress.org/development/2007/04/wordpress-213-and-2010/

**12:** http://trac.wordpress.org/ticket/4091

#### Episode 2

#### Verlauf

- Problemmeldung 10 oder 11.2007 im Forum
- Presseveröffentlichung 02.02.2008, Ab hier beginnt die Schwachstellenveröffentlichung
- vollständig behoben 04.02.2008
- Releasveröffentlichung 05.02.2008

**Einstufung** Hoch

**Schwachstellenart** Umgehen der Authentifizierung (Schnittstelle)

**Dokumente**

**01:** http://trac.wordpress.org/ticket/5313

**02:** http://wordpress.org/support/topic/134928

**03:** http://wordpress.org/support/topic/134928/page/2

**04:** http://wordpress.org/support/topic/134928/page/3

**05:** http://www.theseekerblog.com/?p=284 [Seite existiert nicht mehr, Daten nur noch in HU]

**06:** http://www.village-idiot.org/archives/2008/02/02/wordpress-232-exploit-confirmed/

**07:** http://www.theseekerblog.com/?p=285 [Seite existiert nicht mehr, Daten nur noch in HU]

**08:** http://www.bontb.com/2008/03/wp-content1-trojan-virus-for-wordpress-bloggers/

**09:** http://wordpress.org/development/2008/02/wordpress-233/

**25:** http://www.village-idiot.org/archives/2008/02/02/wordpress-232-xmlrpc-vulnerability-fix/

**Episode 3**

**Verlauf**

- Bug behoben am 29.05.2007
- (Sicherheits-)Problemmeldung durch den Bug in einem Blobeintrag am 07.06.2007
- Veröffentlichung der Behebung 21.06.2007

**Einstufung** Hoch

**Schwachstellenart** SQL Injection

**Dokumente**

**13:** http://trac.wordpress.org/ticket/4357

**14:** http://wordpress.org/support/topic/120857

**15:** http://kev.coolcavemen.com/2007/06/wordpress-22-security-hole-identity-theft/

**16:** http://www.milw0rm.com/exploits/4039

**26:** http://wordpress.org/development/2007/06/wordpress-221/

### Episode 4

#### Verlauf

- Schwachstellenveröffentlichung am 31.07.07 im Bugtracker
- Patch von externem Helfer 31.07.2007
- Geschlossen und vollständig behoben 01.08.07
- Releasveröffentlichung am 05.08.2007

**Einstufung** Hoch

**Schwachstellenart** XSS

#### Dokumente

**17:** http://trac.wordpress.org/ticket/4689

**18:** http://mybeni.rootzilla.de/mybeNi/2007/wordpress_zeroday_vulnerability_roundhouse_kick_and_why_i_nearly_wrote_the_first_blog_worm/ [Seite existiert nicht mehr, Daten nur noch in HU]

**19:** http://mybeni.rootzilla.de/mybeNi/2007/this_is_the_first_weblog_xss_worm/ [Seite existiert nicht mehr, Daten nur noch in HU]

**28:** http://wordpress.org/development/2007/08/wordpress-222-and-2011/

### Episode 5

#### Verlauf

- Schwachstellenveröffentlichung in der Bugtraq Mailingliste am 15.12.2007
- Im Bugtracker am 19.12.2007
- Releasveröffentlichung am 29.12.2007

**Einstufung** Hoch

**Schwachstellenart** Umgehen der Authentifizierung

**Dokumente**

**20:** http://www.securityfocus.com/archive/1/485160

**21:** http://trac.wordpress.org/ticket/5487

**22:** http://www.blackhatdomainer.com/how-to-know-today-what-shoemoney-is-going-to-post-tomorrow/

**27:** http://wordpress.org/development/2007/12/wordpress-232/

### A.4.2 Joomla

**Episode 1**

**Verlauf**

- Schwachstellenveröffentlichung 27.12.2007 im Forum
- Patch von externem Helfer 02.01.2008
- Fix im SVN 04.01.2008
- offizieller Patch in Form eines RCs am 14.01.2008
- Releasveröffentlichung 11.02.2008

**Einstufung** Hoch

**Schwachstellenart** CSRF

**Dokumente**

**01:** http://forum.joomla.org/viewtopic.php?f=267&t=248109

**02:** http://www.securityfocus.com/archive/1/485676/30/0/threaded

**03:** http://www.joomla.org/content/view/4446/1/

**05:** http://joomlacode.org/gf/project/joomla/tracker/?action=TrackerItemEdit&tracker_item_id=8361

**13:** http://www.joomla.org/content/view/4563/1/ [Seite existiert nicht mehr, Daten nur noch in HU]

### Episode 2

#### Verlauf

- Schwachstellenveröffentlichung 12.10.2007 im Forum
- Patch von externem Helfer 03.01.2008
- Fix im SVN 04.01.2008
- Releasveröffentlichung 11.02.2008

**Einstufung** Niedrig

**Schwachstellenart** XSS

#### Dokumente

**06:** http://forum.joomla.org/viewtopic.php?f=267&t=222837

**07:** http://nvd.nist.gov/nvd.cfm?cvename=CVE-2007-5427

**13:** http://www.joomla.org/content/view/4563/1/ [Seite existiert nicht mehr, Daten nur noch in HU]

**25:** http://www.securityfocus.com/bid/26031

### Episode 3

#### Verlauf

- Schwachstellenveröffentlichung am 17.06.06
- Im Forum am 18.06.06
- Offiziell bestätigt am 19.06.06
- Workaround gepostet von Extern am 19.06.06
- Release am 26.06.06

**Einstufung** Hoch

**Schwachstellenart** SQL Injection

**Dokumente**

**14:** http://forum.joomla.org/viewtopic.php?f=267&t=70117

**15:** http://secunia.com/advisories/20746/

**16:** http://www.joomla.org/content/view/1510/74/1/2/ [Seite existiert nicht mehr, Daten nur noch in HU]

**Episode 4**

**Verlauf**

- Schwachstellenveröffentlichung am 20.09.2007
- Im Bugtracker am 20.09.2007
- Fix im SVN am 03.10.2007

**Einstufung** Hoch

**Schwachstellenart** Malicious File Execution

**Dokumente**

**17:** http://joomlacode.org/gf/project/joomla/tracker/?action=TrackerItemEdit&tracker_id=32&tracker_item_id=7227

**Episode 5**

**Verlauf**

- Schwachstellenveröffentlichung am 28.09.2007
- Im Bugtracker am 28.09.2007
- Fix im SVN am 03.12.2007

**Einstufung** Moderat

**Schwachstellenart** XSS

**Dokumente**

**18:** http://joomlacode.org/gf/project/joomla/tracker/?action=TrackerItemEdit&tracker_id=32&tracker_item_id=7358

**Episode 6**

**Verlauf**

- Schwachstellenveröffentlichung am 13.10.2007
- Im Bugtracker am 13.10.2007
- Fix im SVN am 07.02.2008
- Releasveröffentlichung 11.02.2008

**Einstufung** Niedrig

**Schwachstellenart** XSS

**Dokumente**

**19:** http://joomlacode.org/gf/project/joomla/tracker/?action=TrackerItemEdit&tracker_id=32&tracker_item_id=7608

**Episode 7**

**Verlauf**

- Schwachstellenveröffentlichung am 12.06.2007
- Im Bugtracker am 12.06.2007
- Externe Lösung am 29.06.2007
- Fix im SVN am 02.07.2007
- Release am 21.07.2007

**Einstufung** Niedrig

**Schwachstellenart** XSS

**Dokumente**

**20:** http://joomlacode.org/gf/project/joomla/tracker/?action=TrackerItemEdit&tracker_id=32&tracker_item_id=5654

**26:** http://www.joomla.org/content/view/3670/78/ [Seite existiert nicht mehr, Daten nur noch in HU]

**Episode 8**

**Verlauf**

- Im Bugtracker am 11.06.2007
- Fix im SVN am 02.07.2007
- Release am 21.07.2007

**Einstufung** Niedrig

**Schwachstellenart** HTTP Request Smuggling (HRS)

**Dokumente**

**21:** http://joomlacode.org/gf/project/joomla/tracker/?action=TrackerItemEdit&tracker_id=32&tracker_item_id=5630

**26:** http://www.joomla.org/content/view/3670/78/ [Seite existiert nicht mehr, Daten nur noch in HU]

**Episode 9**

**Verlauf**

- Externe Schwachstellenveröffentlichung am 14.01.08
- Schwachstellenveröffentlichung im Bugtracker am 15.02.2008
- Fix im SVN am 21.02.2007
- Release am 22.02.2008

**Einstufung** Hoch

**Schwachstellenart** Malicious File Execution

**Dokumente**

**22:** http://www.certa.ssi.gouv.fr/site/CERTA-2008-ALE-002/

**23:** http://www.joomla.org/content/view/4609/1/

**24:** http://joomlacode.org/gf/project/joomla/tracker/?action=TrackerItemEdit&tracker_id=32&tracker_item_id=9752

### A.4.3 TWiki

**Episode 1**

**Verlauf**

- Schwachstellenveröffentlichung im Bugtracker am 21.03.2006
- Veröffentlichung eines Hotfix am 25.03.2006
- Release am 31.03.2006

**Einstufung** Moderat

**Schwachstellenart** Umgehen der Authentifizierung

**Dokumente**

**03:** http://develop.twiki.org/~twiki4/cgi-bin/view/Bugs/Item1925

**04:** http://twiki.org/cgi-bin/view/Codev/SecurityAlertTWiki4RdiffPreviewAccess

**Episode 2**

**Verlauf**

- Schwachstellenmeldung an Sicherheitsteam am 28.01.2008
- Veröffentlichung eines Hotfix am 31.01.2008
- Release am 05.02.2008

**Einstufung** Moderat

**Schwachstellenart** Falsche Konfiguration

**Dokumente**

**05:** http://twiki.org/cgi-bin/view/Codev/SecurityAlert-CVE-2007-0669

**Episode 3**

**Verlauf**

- Schwachstellenmeldung an Sicherheitsteam am 29.06.2006
- Veröffentlichung eines Hotfix am 04.07.2006
- Release am 04.07.2006

**Einstufung** Hoch

**Schwachstellenart** Malicious File Execution

**Dokumente**

**06:** http://twiki.org/cgi-bin/view/Codev/SecurityAlertSecureFileUploads

**Episode 4**

**Verlauf**

- Schwachstellenmeldung an Sicherheitsteam am 20.08.2006
- Fix im SVN am 22.08.2006
- Veröffentlichung eines Hotfix am 06.09.2006

**Einstufung** Hoch

**Schwachstellenart** Umgehen der Authentifizierung

**Dokumente**

**08:** http://twiki.org/cgi-bin/view/Codev/SecurityAlert-CVE-2006-4294

**Episode 5**

**Verlauf**

- Schwachstellenmeldung an Sicherheitsteam am 17.11.2006
- Veröffentlichung eines Hotfix am 21.11.2006

**Einstufung** Niedrig

**Schwachstellenart** Umgehen der Authentifizierung

**Dokumente**

**09:** http://twiki.org/cgi-bin/view/Codev/SecurityAlert-CVE-2006-6071

## A.5 Beschreibung des CD Inhalts

Im Hauptordner der CD liegt eine Kopie dieser Arbeit. Darüber hinaus gibt es die beiden Unterordner „Atlas.ti“ und „Axiales Kodieren“. Im ersten Ordner befinden sich:

- meine Hermeneutic Units (Beschreibung im nächsten Abschnitt),
- das Handbuch zu Atlas.ti (in Englisch),
- eine readme.rtf (in Englisch), die Anweisungen zur Installation, Angaben zu den technischen Voraussetzungen und eine Kurzbeschreibung wichtiger neuer Features enthält,
- und eine Demoversion von Atlas.ti (alternativ kann diese unter http://www.atlasti.com/de/demo.html heruntergeladen werden).

Im Ordner „Axiales Kodieren“ befindet sich die Tabellenkalkulationsdatei, mit welche ich mir axiale Kodieren erleichtert habe als Open-Office Calc und als Microsoft Excel Datei. Diese Datei werde ich im Abschnitt A.5.2 beschreiben.

### A.5.1 Meine Atlas.ti Hermeneutic Units

Meine Hermeneutic Units (HU), so wird die Datenstruktur für ein Projekt in Atlas.ti bezeichnet, sind auf die folgenden Ordnerstruktur aufgeteilt:

- 00_Arbeit

- Joomla
- Twiki_+_S9y
- Wordpress

Im ersten Ordner „00_Arbeit “ befinden sich die in dieser Arbeit vorgestellten Konzepte und Kategorien der Verhaltensweisen und Rollen. Hier sind auch die Beziehungen zwischen Kategorien und Konzepten zu finden. Außerdem habe ich in dieser HU Beziehungsnetzwerke (Network Views) zu den Beziehungen zwischen Kategorien und Konzepten erstellt. Diese HU umfasst auch Kategorien, welche ich im Laufe der Arbeit verworfen habe.

Die drei restlichen Ordner „Joomla“, „Twiki_+_S9y“, „Wordpress“ beinhalten die Daten meiner Projekte. Direkt in diesen Projektordnern ist die jeweilige HU zu finden und die Dokumente, welche die Metadaten beinhalten. Die Unterordner der Projektordner beginnen alle mit der Bezeichnung „Episode_“ und beinhalten lokale Kopien der Daten im Rich-Text-Format (rtf) oder in Form einer Textdatei. Die Wordpress HU ist vollständig kodiert, während die Joomla HU nur rudimentär kodiert wurde. Diese und TWiki HU dienen dazu die Daten zu den Projekten zu archivieren.

### A.5.2 Tabellen und Diagramme des Axialen Kodierens

Die Tabellenkalkulationsdatei beinhaltet Diagramme, Matrizen und Listen, mit denen ich während des axialen Kodierens gearbeitet habe. In der Tabelle „Übersicht“ sind die einzelnen Eigenschaften, der in Beziehung gesetzten Kategorien, wie in Abschnitt 3.3.3 vorgestellt, und deren Eigenschaften zeilenweise in der ersten Spalte angeordnet. In den Spalten habe ich für jede Episode die Zelle zu den zutreffenden Eigenschaften bzw. dimensionalen Ausprägungen der jeweiligen Kategorie ausgefüllt. Den Eigenschaften folgen Einträge zu Ablauf der Schwachstellenbehebung, Links zu den Daten, Behebungsdauer in Tagen und Monaten und Anmerkungen zu jeder Episode.

In der Tabelle „Ursächlichen Bedingung“ ist die Beziehung von Dauer und optionalen Eigenschaften der ursächlichen Bedingung auf die einzelnen Episoden abgebildet, welche als Vorlage für Tabelle auf Seite 58 diente.

In der Tabelle „Kontext“ sind die Eigenschaften des Phänomens und deren dimensionalen Ausprägung auf die einzelnen Rollenkategorien abgebildet, wobei die Episoden in die entsprechende Abbildung eingetragen wird.

In der Tabelle „Konkreter Schutz“ sind die vom externen Helfer und Hersteller eingesetzten Strategien der Schwachstellenbehebung bzw. Minimierung potentieller Schwachstellenfolgen auf die Episoden abgebildet.

In der Tabelle „Diagramme“ habe ich das Diagramm aus Abbildung 3.3 auf Seite 56 erstellt.