A=SIT Technology
E-Government Innovationszentrum

30. Oktober 2020 Studie · Blockchain · Web & Cloud · 27 Min. Lesezeit

# Sicherheitsanalyse von IPFS

**Archiviert.** Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

**Stand** 2020-10-29
**Quelle** https://technology2.a-sit.at/veroeffentlichungen/sicherheitsanalyse-von-ipfs/

Im Rahmen dieses Projekts wurde IPFS, ein dezentrales Peer-to-Peer-Hypermedia-System, einer Sicherheitsanalyse unterzogen. Dabei konnten Schwachstellen mit schwerwiegenden Folgen gefunden werden, die dazu ausgenutzt werden können, das gesamte IPFS-Hauptnetzwerk nach Belieben zu segmentieren und zum Erliegen zu bringen. Ein entsprechender Angriff wurde umgesetzt und erfolgreich evaluiert. Besonders kritisch ist in diesem Zusammenhang, dass dafür kaum Kosten oder sonstige Aufwände entstehen, wodurch der Angriff prinzipiell von jedermann ausgeführt werden kann. Protocol Labs, die Firma, die die IPFS-Entwicklung maßgeblich vorantreibt und als Hauptsponsor des an sich quellenoffen entwickelten Systems auftritt, wurde zeitnah informiert, was bereits nach kurzer Zeit zur Umsetzung erster Gegenmaßnahmen geführt hat. Im Sinne eines Responsible-Disclosure-Prozesses wurde der Projektbericht nicht vor Ende Oktober 2020 veröffentlicht und dessen Freigabe mit Protocol Labs koordiniert. Im Zuge dessen hat Protocol Labs einen Post am offiziellen IPFS-Blog zu diesem Thema veröffentlicht. Zusätzlich wurde eine (aktuell in Begutachtung befindliche) wissenschaftliche Publikation zu den Projektergebnissen erarbeitet.

Links

IPFS-Blog

Pre-Print

## 1. Einleitung

Ein erklärtes Ziel von IPFS ist, das World-Wide-Web zu re-dezentralisieren. Im speziellen bedeutet das, dass Inhalte nicht wie aktuell meist von einigen wenigen Großkonzernen wie Amazon, Google, Cloudflare oder Facebook gehostet werden sollen, sondern direkt von Nutzerinnen und Nutzern. Um diesem Ziel näherzukommen, konnte Protocol Labs bereits über 250 Mio. USD an Risikokapital [4] lukrieren. Praktisch wurden in einigen Bereichen bereits Etappenziele am Weg zur Umsetzung dieser Vision erreicht. Beispielsweise verzeichnen auf der IPFS-Technik basierende Portale wie *PeerTube* [5] oder *DTube* [6] Millionen täglich aktive Nutzer. Darüber hinaus erreicht ein von Protocol Labs betriebener HTTP-Gateway

(der es auch ohne eine lokale IPFS-Installation ermöglicht, auf IPFS-Inhalte zuzugreifen) wöchentliche Nutzerzahlen in Millionenhöhe. Um das übergeordnete Ziel eines dezentralen Webs zu erreichen, ist ein robustes, sicheres Peerto-Peer-Netzwerk erforderlich. Im Fall von IPFS wird diese Funktionalität durch ein dezentrales, 1 https://ipfs.io 2 https://github.com/ipfs/go-ipfs 3 https://protocol.ai 4 https://www.coindesk.com/257-million-filecoin-breaks-time-record-ico-funding 5 https://peer.tube 6 https://d.tube strukturiertes Peer-to-Peer-Netzwerks auf *S/Kademlia* -Basis [1] umgesetzt, welches durch das Subprojekt *libp2p* [7] bereitgestellt wird. Wie alle auf Kademlia-basierenden [2] Konzepte stellt auch S/Kademlia eine *Distributed Hash Table* (DHT) bereit, die sowohl für Routing-Anfragen nach anderen Teilnehmerinnen und Teilnehmern, als auch Dateien verwendet wird (s.u.). Im Kern handelt es sich um ein Mapping von logischen Identifikatoren auf IP-Adressen und Ports. Routing ist ein iterativer, kollaborativer Prozess; jede Instanz führt lokale Routingtabellen und beantwortet Anfragen entweder durch Übermittlung der angefragten Daten, oder durch Übermittlung einer Menge an bekannten Netzwerkknoten die dem Ziel (aus der eigenen Sicht) näher sind. In letzterem Fall wird der Prozess so lange wiederholt, bis das Ziel gefunden wurde. Entscheidend ist in diesem Zusammenhang, dass jede Instanz nur eine eingeschränkte Sicht auf das Gesamtsystem hat und auf Basis von lokalem Wissen Anfragen beantwortet. Der gesamte Klasse von strukturierten Peer-to-Peer-Netzen sind zwei miteinander verschränkte Eigenschaften gemein: (1) Alle Anfragen werden im Netzwerk streng deterministisch (und damit vorhersehbar) abgearbeitet, was nur durch (2) strikte Obergrenzen bezüglich der gleichzeitig zu anderen Teilnehmerinnen und Teilnehmern offengehaltenen Verbindungen möglich ist. Um insbesondere keiner Limitierung bezüglich offener Verbindungen zu unterliegen, definiert IPFS zusätzlich zur strukturierten S/Kademlia-Komponente den so genannten *Swarm* . Dabei handelt es sich um eine für sich genommen unbegrenzte Menge an aktiven Verbindungen zu anderen Netzwerkteilnehmerinnen und -teilnehmern. Der Swarm wird im Rahmen von IPFS bevorzugt für Anfragen nach gesuchten Inhalten genutzt: Eine jede Anfrage wird im ersten Schritt mittels Broadcast über alle aktiven Verbindungen an den Swarm gesendet, und erst wenn keiner dieser direkt kontaktierten Teilnehmerinnen und Teilnehmer die gesuchten Inhalte bereitstellen kann, wird auf die S/Kademlia-Komponente zurückgegriffen. Daraus ergeben sich zwei unmittelbare Konsequenzen: Einerseits können Anfragen teilweise in konstanter Zeit ( $O(1)$) beantwortet werden, andererseits ergibt sich dadurch ein gewisses Maß an Resistenz gegen bestimmte Angriffsklassen, die auf die strukturierten Eigenschaften der S/Kademlia-Komponente abzielen.

Das IPFS-Hauptnetzwerk ist ein vollkommen dezentrales, offenes System, d.h. es gibt keine Vertrauensbasis zwischen Teilnehmerinnen und Teilnehmern. Tatsächlich ist das gesamte Netzwerk vollkommen vertrauenslos organisiert und es gibt schlicht kein Vertrauensmodell. Zwar ist der Identifikator eines jeden Netzwerkknotens von einem public/private Key Pair abgeleitet und die gesamte Kommunikation ist Ende-zu-Ende-verschlüsselt, allerdings können beliebige Identifikatoren generiert werden. Da kein Ressource-Testing (Proof-of-Work, o.Ä.) eingesetzt wird, ist dies effizient durchführbar. Dadurch ergibt sich die Möglichkeit, mittels Brute-Force gezielt Identifikatoren zu generieren, sodass diese die Routingtabelle eines zuvor ausgewählten Opfers füllen und alle anderen, ehrlichen Teilnehmerinnen und Teilnehmer aus dieser Routing-Tabelle verdrängen. Dies ist im Fall von IPFS praktisch umsetzbar, kann direkt auf den Swarm ausgeweitet werden und bildet den Kern des im Rahmen dieses Projekts entwickelten Angriffs.

Nachfolgend werden alle zum Verständnis des entwickelten Angriffs erforderlichen Grundlagen umrissen. Anschließend werden die in IPFS entdeckten Schwachstellen skizziert, welche in Abschnitt 4. aufgegriffen werden, um darzulegen, wie diese für konkrete Angriffe bis hin zur vollkommenen Übernahme des

gesamten IPFS-Hauptnetzwerkes ausgenutzt werden können. Abschnitt 5. zieht über die Projektergebnisse Bilanz und gibt einen Ausblick auf mögliche weitere Angriffsmöglichkeiten, bzw. naheliegende weitere Sicherheitsanalysen.

# 2. Peer-to-Peer-Netzwerkgrundlagen

Dieser Abschnitt fasst die im Kontext dieses Projekts wichtigen Grundlagen zu Peer-to-Peer-Netzwerken zusammen. Insbesondere wird ein Überblick über strukturelle Unterschiede in der Organisation von Peer-to-Peer-Netzen gegeben. Darüber hinaus werden für dieses Projekt relevante Angriffskonzepte im Peer-to-Peer-Bereich erläutert.

7 https://libp2p.io

## 2.1. Strukturell unterschiedliche Peer-to-Peer-Netzwerkkonzepte

Im Kontext diese Projekts ist insbesondere die Kategorisierung von Peer-to-Peer-Netzwerken in *strukturiert* und *unstrukturiert* relevant. Generell wird anstatt der IP-Netzwerktopologie eine Abstraktionsschicht zur Netzwerkorganisation herangezogen, die lediglich logische Identifikationen zur Identifikation und Lokalisierung von Netzwerkknoten verwendet, welche transparent auf IP-Adressen abbilden. Die grundlegende Motivation hinter diesem Ansatz ist, die Komplexität der eigentlichen Netzwerktopologie, welche in weiten Teilen die geografische und organisationsinterne Struktur des Internets widerspiegelt, zu verstecken. Dies kann auf zwei Arten erfolgen.

### 2.1.1. Strukturierte Peer-to-Peer-Netze

Strukturierte Peer-to-Peer-Netze stellen üblicherweise lediglich Operationen wie das Lokalisieren von Netzwerkknoten oder Daten (und Pointern zu Daten) zur Verfügung. Dadurch lässt sich effizientes Routing mit definierten Umlaufzeiten umsetzen. Die verteilten Routingprotokolle von strukturierten Peer-to-Peer-Designs wie *Chord* [3], *CAN* [4] und *Kademlia* [2] garantieren, dass nie mehr als $O(\log n)$ viele Hops benötigt werden, um einen Knoten oder Daten zu lokalisieren. Routinganfragen konvergieren immer auf dem kürzest möglichen Weg zum Ziel. Eine Grundvoraussetzung hierfür ist die Definition einer Distanzfunktion zwischen Knoten (und Daten). Um eingehende Anfragen effizient zu beantworten, reicht ein Abgleich gegen alle in der lokalen Routingtabelle gespeicherten Knoten aus, da eine Sortierung auf Basis dieser Distanzfunktion vorliegt. Folglich geht aus der Routingtabelle direkt hervor, welche bekannten Knoten näher am angefragten Ziel sind. Auf Grund der garantierten Konvergenz von Routingpfaden lässt sich darüber hinaus (zumindest konzeptionell) zweifelsfrei feststellen, ob ein gesuchter Knoten aktuell nicht Teil des Netzwerks ist. Eine Eigenschaft dieses durchgängig deterministischen Verhaltens ist jedoch, dass Angreifer gezielt Knoten mit bestimmten Distanzen zu einem Opfer betreiben können, um so sicherzustellen dass diese an zuvor festgelegten Stellen in die Routingtabelle aufgenommen werden. Wie in Abschnitt 2.2.2. beschrieben, kann dies genutzt werden, um Knoten vom Netzwerk abzuschneiden.

### 2.1.2. Unstrukturierte Peer-to-Peer-Netze

Unstrukturierte Peer-to-Peer-Netze besitzen keine vorgegebene Netzwerkstruktur und erlauben im Prinzip beliebige Organisationsformen von Routingtabellen und Anfragebeantwortungen an Hand frei wählbarer Parameter. Folglich werden weder Maximalumlaufzeiten, Effizienz, noch Konvergenz von

(Routing-)Anfragen garantiert. Dadurch gibt es konzeptionell de-facto keine Limitierungen, was die zur Verfügung gestellte Funktionalität betrifft. Somit lassen sich auch erweiterte Funktionen, wie Volltextsuche direkt vom unstrukturierten Peer-to-Peer-Netzwerk bereitstellen. Ein naiver Umsetzungsansatz ist Flooding, wodurch kurze Umlaufzeiten erreicht werden können, jedoch ebenso hohe Netzwerklast entstehen kann. Optimierungen, wie z.B. *Random-Walks* [5], schaffen hier Abhilfe, erhöhen jedoch die Komplexität des verteilten Routingverfahrens. Insbesondere im Kryptowährungskontext sind unstrukturierte Peer-to-Peer-Netze populär, da hier vornehmlich Broadcasting eingesetzt wird, um Transaktionsinformationen schnellstmöglich an alle Teilnehmerinnen und Teilnehmer verbreiten zu können. Durch die unstrukturierte Organisation und die nichtdeterministische Ausbreitung von Routinganfragen im Netzwerk kann es aus Sicht eines Angreifers, bzw. einer Angreiferin schwieriger sein, bestimmte Attacken auf Peer-to-Peer-Netzwerke auch auf Basis von konzeptionell per Definition durchführbaren Angriffsmustern in die Praxis umzusetzen.

## 2.2. Peer-to-peer-spezifische Angriffe und Gegenmaßnahmen

Unabhängig von der Struktur eines Netzwerks stellt sich die Frage nach der hierarchischen Organisation. Peer-to-Peer-Netze können zentral organisiert sein, d.h. eine, bzw. einige wenige Instanzen sind für den Betreib essentiell, oder vollkommen dezentral, wodurch jedoch unter Anderem Zugangskontrollen unter realistischen Bedingungen nicht durchgesetzt werden können. Darüber hinaus gibt es hybride Formen, welche das gesamte Spektrum zwischen beiden Extremen abdecken. Ein prominentes Beispiel für ein zentralisiert organisiertes Peer-to-Peer-Netzwerk ist Tor [8] . Das IPFS-Hauptnetzwerk ist im krassen Gegensatz dazu bewusst vollkommen dezentral und ohne Zugangskontrollen umgesetzt, wodurch ganze Klassen von Angriffen – wenn überhaupt – nur bekämpft, aber kaum verhindert werden. Die Thematik an sich wurde bereits in vorangegangenen A-SIT-Projekten (u.A. [6]) behandelt. Im Kontext dieses Projekts sind mit Hinblick auf die entdeckten Schwachstellen und den umgesetzten Angriff zwei Angriffskonzepte, sowie bekannte Gegenmaßnahmen relevant.

### 2.2.1. Sybil-Attacke

Unter einer Sybil-Attacke versteht man im Peer-to-Peer-Kontext das Auftreten eines einzelnen Akteurs durch nach außen hin viele, scheinbar unabhängige Identitäten. An sich handelt es sich dabei nicht notwendigerweise um bösartiges Verhalten, oder um einen konkreten Angriff. Allerdings kann dieses Vorgehen ausgenutzt werden, um einen verhältnismäßig hohen Einfluss z.B. auf das verteilte Routingprotokoll innerhalb eines Peer-to-Peer-Netzwerks zu erhalten und infolgedessen disruptive Aktionen auszuführen [7].

### 2.2.2. Eclipse-Attacke

Als Eclipse-Attacke wird das Umzingeln einzelner Knoten eines Peer-to-Peer-Netzwerks [8] bezeichnet. Im Kontext strukturierter Peer-to-Peer-Netzwerke ist dies durch gezieltes Erzeugen von Identifikatoren bestimmter Distanzen zu einem zuvor ausgewählten Opfer möglich. Anfälligkeit gegenüber Sybil-Attacke begünstigt Eclipse-Attacken. Schafft es ein Angreifer, bzw. eine Angreiferin Knoten so zu betreiben, dass diese die gesamte Routingtabelle eines Opfers füllen, kann dieses vom restlichen Netzwerk abgeschnitten werden. Dezentrale, offene Peer-to-Peer-Netze sind typsicherweise anfällig für diese Angriffskombination.

### 2.2.3. Gegenmaßnahmen

Auf Grund der typischerweise engen Verschränkung zwischen Sybil- und Eclipse-Attacken kann beidem Einhalt geboten werden, wenn es gelingt, die Anzahl der Identifikatoren, bzw. Netzwerkknoten, die von einer einzelnen Partei betrieben werden können, zu beschränken. Hierfür bietet sich Zentralisierung und in begrenztem Ausmaß situationsabhängig auch Resource-Testing (wie z.B. Proof-of-Work-basierte Identifikatorgenerierung) an [1]. Grundvoraussetzung ist unabhängig von der gewählten Strategie ein Verfahren zur Identifikatorzertifizierung und bzw. Identifikatorverifizierung. D.h. der Besitz eines Identifikators muss (typischerweise mit Hilfe digitaler Signaturverfahren) nachgewiesen werden. Wichtig ist in diesem Zusammenhang, dass jeder Teilnehmer und jede Teilnehmerin im Netzwerk eine derartige Zertifizierung unabhängig prüfen können muss, und für diese Prüfung kein Kontakt zu einer zentralen Instanz notwendig sein darf, da eine solche Umsetzung einerseits dem Peer-to-Peer-Paradigma widerspricht, andererseits schlicht nicht praktikabel ist. Für zentral organisierte Peer-to-Peer-Netze bietet sich beispielsweise eine PKI (in Form zertifikatsbasierter Identifikatoren) an. Im vollständig dezentralen Fall gab es bisher keine zufriedenstellenden Lösungen für dieses Problem. Im Rahmen eines vorangegangenen Projekts [9] wurde ein Lösungsvorschlag auf Basis von Trusted-Computing im Android-Umfeld entwickelt, welcher im Rahmen einer internationalen IT-Sicherheitskonferenz vorgestellt wurde [10].

Eclipse-Attacken können insbesondere im Fall strukturierter Peer-to-Peer-Netze verhältnismäßig einfach ausgeführt werden, sofern Sybil-Attacken effizient durchführbar sind. Grund dafür ist, dass a priori klar ist, wie Identifikatoren für einen erfolgreichen Angriff verteilt sein müssen, um ein Opfer zu umzingeln. Im Kontext des IPFS-Hauptnetzwerks wurde im Rahmen dieses Projekts festgestellt, dass keinerlei Gegenmaßnahmen implementiert sind. Teilweise lässt sich dies darauf zurückführen, dass ein vollständig offenes System ohne Beitrittsbarrieren angestrebt wird. Weitere Details zu IPFS-spezifischen Schwachstellen werden im nachfolgenden Abschnitt beschrieben.

## 3. Identifizierte Schachstellen in IPFS

Ein grundlegendes Problem im Kontext der Abwehr gegen Eclipse-Attacken ist die generische 8 https://www.torproject.org/ Ausrichtung von IPFS, bzw. von libp2p. IPFS kann einerseits als Implementierung eines Gesamtkonzepts zum dezentralen Datenaustausch angesehen werden. Andererseits handelt es sich aus softwarearchitektonischer Sicht um einen relativ losen Zusammenschluss von Komponenten, welche mit libp2p-Modulen interagieren. Durch die Abstraktionsebenen und die anwendungsunabhängige Implementierung einiger libp2p-Komponenten ergibt sich die Situation, dass Verhalten, das im Rahmen der IPFS-Kernfunktionalität stattfindet, von einigen Submodulen als weniger relevant klassifiziert wird, als bösartiges Verhalten. Relevant in diesem Zusammenhang ist primär das Verbindungsmanagement, wofür der *ConnMgr* (Connection Manager) verantwortlich ist. Ziel des Connection Managers ist, immer ein „vernünftiges“ Maß an Verbindungen offen zu halten, um einerseits hohe Konnektivität und Redundanz im Routing sicherzustellen, gleichzeitig jedoch eine Überlastung zu verhindern. Die unteren („lowWater“) und oberen („highWater“) Schranken sind auf 600, bzw. 900 Verbindungen voreingestellt, können jedoch frei konfiguriert werden. Diese Werte lassen sich auch ohne direkten Zugang zu einer Instanz durch entsprechenden Verbindungsauf- und Abbau ermitteln. Wird eine neue Verbindung hergestellt (egal ob eingehend oder ausgehend), wird diese dem Swarm hinzugefügt. Sofern in der Routingtabelle des IPFS-Knotens noch Platz für diese neue Verbindung vorhanden ist, wird der Kommunikationspartner in diese aufgenommen. Alle sechzig Sekunden (dieser Wert ist vorgegeben)

wird die Anzahl der offenen Verbindungen im gesamten Swarm überprüft. Sofern die obere Schranke überschritten wird (was im Regelbetrieb häufig im Ausmaß von 300-500 Verbindungen passiert), startet der folgende Prozess:

1. Ignoriere alle offenen Verbindungen, die in den letzten 20 Sekunden hergestellt wurden. Dieser Wert wird's als *Grace Period* bezeichnet, ist konfigurierbar und kann ohne direkten Zugriff auf eine Instanz durch gezielten Auf- und Abbau von Verbindungen ermittelt werden.
2. Bleiben nach der Subtraktion dieses Werts von der Gesamtmenge aktiver Verbindungen weniger als „lowWater" viele Verbindungen übrig, kommt es zu keiner Trennung und der Prozess wird an dieser Stelle beendet.
3. Andernfalls sortiere alle restlichen offenen Verbindungen nach Punkten. Punkte werden erteilt, wenn z.B. aktiv Daten mit einem Knoten ausgetauscht werden, dieser Daten anbietet, oder die betreffende Verbindung in die Routingtabelle aufgenommen wurde.
4. Trenne (ausgehend vom niedrigsten Punktestand) so lange Verbindungen, bis lediglich „lowWater" viele offene Verbindungen bestehen. Getrennte Verbindungen werden aus der Routingtabelle und dem Swarm entfernt.

Wenn ein Angreifer, bzw. eine Angreiferin ausreichend viele Instanzen betreiben kann, die sich entsprechend verhalten, kann dieser Prozess genutzt werden, um ein Opfer dazu zu bringen, alle Verbindungen zu anderen Netzwerkknoten zu trennen. Dies kann im Wesentlichen durch das „Erschleichen" von Punkten ohne Gegenleistung erreicht werden. In Folge dessen kann ein Opfer vom restlichen Netzwerk abgeschnitten werden, wenn vom Opfer abgesetzte Anfragen bezüglich anderer Knoten oder Daten, sowie vom restlichen Netzwerk kommende Anfragen an das Opfer blockiert werden. Dies entspricht im Kern einer Eclipse-Attacke.

### 3.1. Konzeptionelle Probleme

Ein Hauptproblem der konzeptionellen Umsetzung von IPFS ist die fehlende Semantik des Punktesystems, auf dessen Basis der ConnMgr entscheidet, welche Verbindungen getrennt werden sollen. Prinzipiell kann jedes Subsystem jeder Verbindung unter einem frei wählbaren Label/Tag Punkte zuweisen. Im Rahmen der zuvor beschriebenen Routine findet jedoch keine Wertung betreffend der Quelle oder Semantik von Punkten statt. Beispielsweise kann eine Verbindung, welche aktiv zum Datenaustausch genutzt wird, jedoch nicht in die Routingtabelle aufgenommen wurde, mehr Punkte erreichen als manche Verbindungen, die als Teil der Routingtabelle für das Auffinden anderer Knoten und Daten genutzt wird. Unter bestimmten Voraussetzungen erscheint dieses Verhalten durchaus sinnvoll, da beispielsweise aktive Downloads oder Uploads nicht durch vom ConnMgr veranlasste Verbindungstrennungen unterbrochen werden sollen. Da jedoch auch das bloße Anbieten von Daten, selbst wenn dies ungefragt erfolgt, im Sinne des aktiven Informations- und Datenaustauschs mit Punkten honoriert wird, lassen sich auf diese Art Punkte erschleichen. Eine weitere Möglichkeit, Punkte zu erhalten, besteht darin, als *Relay* aufzutreten und die eigene Verbindung zum Multiplexen von Verbindungen zu Teilnehmerinnen und Teilnehmern hinter Firewalls oder NATs zur Verfügung zu stellen. Dadurch wird insgesamt erhöhte Konnektivität auch zu auf Grund restriktiver Firewall-Regeln eigentlich nicht von außen erreichbaren Teilnehmerinnen und Teilnehmern ermöglicht. Insgesamt ergibt sich eine schwierige Situation, da Nutzerinnen und Nutzer dazu angehalten werden sollen, Kapazitäten für die Bereitstellung von Daten zur Verfügung zu stellen. Gleichzeitig macht die offene, dezentrale und vertrauenslose Betriebsumgebung ohne feste Identitäten jedoch eine Rückverfolgbarkeit bestimmter Aktionen (egal ob diese positiv oder negativ zu bewerten sind) auf einzelne

Akteure de facto unmöglich und Sybil-Attacken wird Tür und Tor geöffnet. Dies lässt sich auf Grund der strukturierten Organisation der IPFS-Routingtabellen nach dem Kademlia-Schema direkt ausnutzen, da bei entsprechender Ausnutzung des ConnMgr-Verhaltens eine Situation erreicht werden kann, in der im ersten Schritt die gesamte Routingtabelle und in weiterer Folge der gesamte Swarm von einem einzelnen Angreifer, bzw. einer einzelnen Angreiferin besetzt werden kann. Erschwerend kommen eine Reihe Implementierungsfehler hinzu. Auch wenn diese für sich genommen nicht notwendigerweise Angriffe ermöglichen, verringert sich der Aufwand, der betrieben werden muss, um erfolgreiche Angriffe auszuführen, teilweise beträchtlich. Details sind dem nachfolgenden Abschnitt zu entnehmen.

## 3.2. Implementierungsfehler

Im Rahmen dieses Projekts wurden die nachfolgenden Implementierungsfehler identifiziert. Teilweise bieten sich einfache Lösungsmöglichkeiten an. Alle Erkenntnisse wurde zeitnah an Protocol Labs kommuniziert.

### 3.2.1. Bedingungsloses Entfernen von Verbindungen aus der Routintgtabelle:

Auf Grund fehlender Semantik betreffend der vom ConnMgr herangezogenen Punkte im Rahmen der Routine zum Trennen von Verbindungen wird den Verbindungen in der Routingtabelle kein besonderer Stellenwert beigemessen. Da getrennte Verbindungen bis IPFS 0.4.23 unmittelbar aus der Routingtabelle entfernt wurden, können (sofern entsprechende (vor)generierte Identifikatoren vorhanden sind) auf Grund des begrenzten Platzes in der Routingtabelle, alle Routinginformationen betreffend ehrlicher Netzwerkknoten durch Routinginformationen zu Sybil-Knoten ersetzt werden. Da es keine Feedbackschleife zwischen ConnMgr und Routing-Subsystem gibt, wird derartiges Fehlverhalten des ConnMgrs nicht detektiert. Abhilfe würde einerseits eine Sonderbehandlung von Verbindungen, welche Teile der Routingtabelle sind, schaffen. Andererseits besteht die Möglichkeit, Routinginformationen weiter bestehen zu lassen, auch wenn die darunterliegende Verbindung getrennt wird. Letzteres wurde mit IPFS 0.5 umgesetzt und schafft bereits in Teilen Abhilfe.

### 3.2.2. Primitives Konnektivitätsmonitoring

Das Routingsubsystem verbindet sich, für den Fall, dass weniger als vier aktive Verbindungen bestehen, erneut zu vorkonfigurierten Bootstrap-Knoten. Dabei handelt es sich funktional um reguläre Knoten, die jedoch jeder Instanz zusätzlich zu Ihrem Identifikator auch über ihre IP-Adresse bekannt sind, damit auch ohne weitere Routinginformationen eine Verbindung zum Netzwerk hergestellt werden kann. Bootstrap-Knoten dienen somit als Einsprungspunkt ins Netzwerk. Dieser initiale Verbindungsaufbau zu einem Peer-to-Peer-Netzwerk wird als *Bootstrap- Prozess*, *Bootstrap-Prozedur*, oder *Bootstrapping* bezeichnet. Abgesehen von diesem erneuten Verbindungsaufbau im Fall schlechter Konnektivität (<4 aktive Verbindungen), sind keine weiteren Maßnahmen implementiert, um eine Verbindung zum Netzwerk zu garantieren, bzw. zu überwachen. In Folge dessen ist es Angreiferinnen und Angreifern möglich, die Anzahl aktiver Verbindungen zu einem Opfer nach erfolgreicher Eclipse-Attacke auf vier zu reduzieren, um diesen Zustand der Isolation beizubehalten. Da IPFS auch IPv6, sowie das Multiplexen von (prinzipiell beliebig vielen) logischen Verbindungen über bestehende TCP-Verbindungen unterstützt, ließen sich folglich über hunderttausend Netzwerkknoten nach erfolgter Eclipse-Attacke von nur einem einzigen Computer aus dauerhaft vom IPFS-Netzwerk abschneiden. [9]

### 3.2.3. Unverifiziertes Bootstrapping

Anknüpfend an das zuvor beschriebene Problem findet sich eine weitere Schwachstelle im Bootstrap-Prozess. Beim Hochfahren eines IPFS-Knotens ist der Swarm bereits vor dem Routing-Subsystem verfügbar und akzeptiert eingehende Verbindungen. Auf technischer Ebene ist die Konnektivitätsmanagementroutine lediglich als periodischer Aufruf des Bootstrap-Prozesses umgesetzt. Folglich kann Bootstrapping gänzlich unterbunden werden, wenn es gelingt, vier Verbindungen zu einem Opfer aufzubauen, bevor das Routing-Subsystem bereit ist. Die unmittelbare Konsequenz daraus ist, dass nicht sichergestellt werden kann, ob jemals eine Verbindung zu den vorkonfigurierten Bootstrap-Knoten (und somit tatsächlich zu dem Netzwerk, zu dem eine Verbindung gewollt ist) hergestellt wird. Im Rahmen dieses Projekts wurden keine umfassenden Analysen diesbezüglich ausgeführt, jedoch lassen stichprobenartig durchgeführte Experimente darauf schließen, dass die Zeitspanne zwischen Bereitschaft des Swarms und dem Ausführen der Bootstrap-Prozedur ausreichend lang ist, um diese Schwachstelle in der Praxis, unter Berücksichtigung realer Netzwerklatenzen, ausnutzen zu können. Abhilfe kann leicht geschaffen werden, indem der initiale Aufruf des Bootstrap-Prozesses sicherstellt, dass tatsächlich eine Verbindung zu den konfigurierten Bootstrap-Knoten hergestellt wird.

### 3.2.4. Bedingungsloses Vertrauen auf bereitgestellte Informationen

Ein Problem zieht sich wie ein roter Faden durch viele Bereiche der gesamten IPFS-Codebasis. Typischerweise wird Informationen, selbst wenn sie von einem Knoten stammen, mit dem zuvor nie kommuniziert wurde, blind vertraut. Konkret schlägt sich dies beispielsweise im Routing-Subsystem nieder, das dafür verantwortlich ist, logische Identifikatoren auf IP-Adressen abzubilden: Kündigt ein Knoten über eine eingehende Verbindung an, unter welcher IP-Adresse und welchem Port er seinerseits eingehende Verbindungen akzeptiert, wird diese Information ungeprüft übernommen und im Netzwerk weiterverbreitet. In Fällen wie diesen würde ein einfacher Verbindungstest genügen, um zumindest die Verbreitung offensichtlicher Falschinformationen zu unterbinden. In anderen Bereichen ist eine derartige Verifizierung nicht immer möglich, was auf die vertrauenslose Betriebsumgebung und dezentrale Organisation von IPFS zurückzuführen ist.

# 4. Angriffe auf IPFS bis hin zur Netzwerkübernahme

Ausgehend von den zuvor beschriebenen Schwachstellen wurde ein umfassender Angriff auf die Integrität des gesamten IPFS-Hauptnetzwerks konzeptioniert, implementiert und, soweit dies ohne negative Folgen auf den IPFS-Produktivbetrieb möglich war, evaluiert. Im nachfolgenden Abschnitt wird der Angriffsaufbau skizziert. Anschließend wird in Abschnitt 4.2. beschrieben, wie das gesamte IPFS-Hauptnetzwerk durch Angriffe auf Bootstrap-Knoten in dem Sinne übernommen werden kann, als dass ein Angreifer alle im Netzwerk ausgetauschten Informationen und alle Abläufe im verteilten Routingprotokoll nach Belieben beeinflussen kann. Abschließend wird in Abschnitt 4.3. eine Kostenabschätzung für großflächige Angriffe bis hin zur kompletten Netzwerkübernahme dargelegt.

## 4.1. Angriffsaufbau

Generell erlaubt der entwickelte Angriff das Isolieren von IPFS-Knoten. Grundlage dafür bildet das effiziente Generieren von Identifikatoren für eine Sybil- und anschließende Eclipse-Attacke. Nach Auswahl eines Opfers ist es möglich, dessen Routingtabelle binnen Minuten vollständig mit Sybil-Knoten zu füllen. Eine

vollständige Eclipse-Attacke, welche auch den gesamten Swam umfasst, ist für Instanzen, welche die vorkonfigurierten ConnMgr-Parameter nutzen nach 50 Minuten zu über 75% erfolgreich (siehe Abbildung 1).

9 Tatsächlich wird die Konnektivitätsprüfung lediglich einmal pro Minute durchgeführt. Durch geschicktes Timing könnten daher wesentlich mehr Knoten dauerhaft isoliert werden, als es die Limitierung der Anzahl von TCP-Port vermuten ließe.

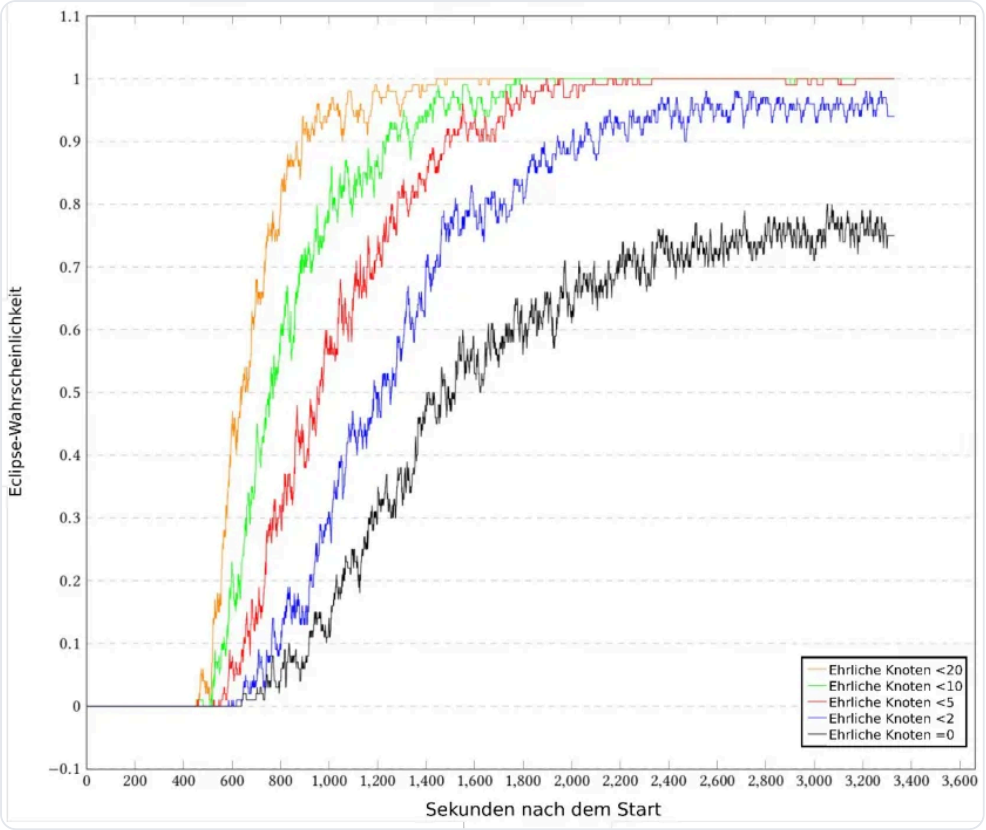



Abbildung 1: Erfolgswahrscheinlichkeit einen IPFS-Knoten vollständig zu isolieren (d.h. den Swarm vollständig mit Sybil-Knoten zu füllen, sodass kein Kontakt zum restlichen Netzwerk mehr besteht)

### 4.1.1. ID-Generierung

Um eine nachhaltige Isolation von IPFS-Knoten zu gewährleisten, ist es notwendig, die Routingtabelle eines Opfers vollständig füllen zu können. Um dies zu ermöglichen wurden 29TB an Identifikatoren vorgeneriert. Diese Menge deckt gleichverteilt einen ausreichend großen Teilbereich des gesamten Identifikatorraums ab, um auf alle realistischen Netzwerkgrößen (bis zu über 4 Mrd Netzwerkknoten) vorbereitet zu sein. Auf Basis dieser Menge kann für beliebig wählbare Knoten jene Untermenge an Identifikatoren bereitgestellt werden, die benötigt wird, um Sybil-Knoten in wohldefinierten (logischen) Distanzen zu einem Opfer so betreiben zu können, dass der gesamte verfügbare Platz in der Routingtabelle des Opfers vereinnahmt wird. Eine Vorgenerierung ist notwendig, da ein SHA-256-Hash in die Distanzberechnung einfließt, wodurch ohne Brute-Force-Verfahren keine Identifikatorgenerierung für bestimmte Distanzen möglich ist. Im Rahmen dieses Projekts hat sich jedoch bereits zu Beginn herausgestellt, dass dies für alle denkbaren Netzwerkgrößen unter realistischen Bedingungen effizient möglich ist. Es wurde ein dateibasiertes Speicherformat für diese Identifikatormenge gewählt: Jeder generierte Identifikator wird an Hand der ersten 14 Bit in eine Datei mit dazu korrespondierendem Dateinamen gespeichert, wodurch sich insgesamt

214=16384 Dateien ergeben. Sollen zu einem Opfer passende Identifikatoren ausgelesen werden, wird die Distanzfunktion im ersten Schritt auf diese ersten 14 Bit angewendet, um die passende Datei zu erhalten. Diese wird anschließend durchsucht, um all jene Identifikatoren zu finden, welche alle notwendigen Distanzen zum Opfer abdecken. Die fehlenden Identifikatoren, um die ersten 14 Bit abzudecken, wenden on-the-fly generiert, da dies (je nach Rechenleistung) innerhalb von einigen Sekunden bis wenigen Minuten möglich ist.

Der Hauptgrund für die effiziente Umsetzbarkeit dieses Verfahrens ist, dass IPFS gänzlich auf Resource-Testing (wie z.B. Proof-of-Work) zur Identifikatorgenerierung verzichtet. Allerdings hat sich die populär Meinung, dass Resource-Testing das gezielte Generieren von Identifikatoren abwenden kann, nicht bestätigt – würde das Generieren von Identifikatoren länger dauern, würde die Vorgenerierung schlicht mehr Zeit in Anspruch nehmen. Einmal generiert, kann die Identifikatormenge jedoch beliebig verwendet werden, wodurch sich der Angriffsaufwand abgesehen von steigenden Stromkosten auf Grund der längeren Dauer der Generierung insgesamt nicht nennenswert erhöht.

Die in diesem Abschnitt beschriebene Funktionalität wurde abschließend in ein Webservice integriert, welches die Suche über die Dateien mit vorgenerierten Identifikatoren umsetzt. Sowohl Identifikatorgenerierung als auch Suche wurde auf Basis des frei verfügbaren IPFS-Quellcodes umgesetzt. Durch die Umsetzung als Webservice lassen sich Angriffe ortsungebunden umsetzen. Im Prinzip könnte auch IPFS selbst, bzw. libp2p verwendet werden, um globale Verfügbarkeit dieses Webservice zu ermöglichen. Um Missbrauch zu verhindern, wurde jedoch ein nicht öffentlich erreichbares Deployment im lokalen Intranet gewählt.

### 4.1.2. Sybil-Attacke

Ausgehend von der Möglichkeit, alle für eine Sybil- und Eclipse-Attacke benötigten Identifikatoren auf Abruf zu haben, wurde ein funktionsreduzierter libp2p-basierter Netzwerkknoten implementiert, welcher sich zu einem frei wählbaren Opfer verbindet. Dabei tritt dieser Knoten im Sinne einer Sybil-Attacke nach außen hin als hunderte unabhängige Knoten auf. Durch die verhältnismäßig geringe Größe des IPFS-Netzwerks, ist es ohne zusätzlichen Aufwand durch den bloßen Verbindungsaufbau möglich, auf Basis nahezu aller vorgenerierten Identifikatoren bereits einen Großteil der Routingtabelle eines Opfers zu okkupieren. Um auch die schon von anderen (ehrlichen) Netzwerkknoten besetzten Routingtabelleneinträge besetzen zu können, müssen diese zuvor verdrängt werden. Die Vorgehensweise dafür wird nachfolgend beschrieben.

### 4.1.3. ConnMgr-Angriff

Im Kern wird ein Opfer dazu gebracht, sich selbst vom ehrlichen IPFS-Netzwerk zu trennen, indem das in Abschnitt 3. beschriebene Verhalten des ConnMgr ausgenutzt wird. A priori ergibt sich für diejenigen von vornherein in die Routingtabelle des Opfers aufgenommenen Verbindungen auf Basis vorgenerierter Identifikatoren bereits ein Punktevorteil gegenüber durchschnittlichen ehrlichen Verbindungen. Dadurch sinkt die Wahrscheinlichkeit, dass diese getrennt werden von vornherein zu Ungunsten ehrlicher Verbindungen. Da es sich dabei jedoch um weniger als 600 Verbindungen handelt, bleiben ausgehend von der Standardkonfiguration der oberen (900) und unteren (600) Schranken bezüglich des Verbindungsmanagements einige ehrliche Verbindungen nach Überschreiten der oberen Schranke bestehen. Um diese Lücke zur unteren Schranke zu schmälern, werden periodisch Daten angeboten, um zusätzlich Punkte zu erhalten. Im letzten Schritt werden weitere Sybil-Knoten mit zufällig generierten

Identifikatoren gestartet, deren Verbindungen zum Opfer hin über bestehende Verbindungen gemultiplext. Bei den bestehenden Verbindungen handelt es sich um genau diejenigen, die nicht von vornherein vom Opfer in dessen Routingtabelle aufgenommen wurden. Hierfür wird die in Abschnitt 3.1. erwähnte Relay-Funktionalität ausgenutzt. In diesem Kontext besteht die Hauptproblematik darin, dass Verbindungen, welche viele zusätzliche Verbindungen multiplexen (aus naheliegenden Gründen im Sinne der Konnektivität zwischen Netzwerkknoten) eine hohe Anzahl an Punkten erreichen können. Tatsächlich steigt diese linear (im Verhältnis 1:1) mit der Anzahl gemultiplexter Verbindungen. Da für den Angriff die Funktionalität einzelner, nach außen hin voneinander unabhängigen Sybil-Instanzen stark reduziert wurde, können bereits auf durchschnittlich leistungsfähiger Hardware ausreichend viele Verbindungen hergestellt (und gemultiplext) werden, um auch unter realistischen Bedingungen 600 Verbindungen mit mehr Punkten als der am höchsten bewertete ehrliche Knoten zu erreichen. In Folge dessen trennt der ConnMgr des Opfers alle ehrlichen Verbindungen und das Opfer isoliert sich selbst.

Zusätzlich zu diesem Angriff zur vollständigen Isolation beliebiger IPFS-Knoten kann dieses Schema auch genutzt werden, um den ConnMgr der Bootstrap-Knoten so auszunutzen, dass der Bootstrap-Prozess immer dazu führt, dass jede sich zum IPFS-Netzwerk verbindende Teilnehmerin, bzw. jeder sich verbindende Teilnehmer ausschließlich Routinginformationen zu von einem Angreifer, bzw. einer Angreiferin betrieben Knoten erhält. Details hierzu sind dem nachfolgenden Abschnitt zu entnehmen.

## 4.2. Netzwerkübernahme über Bootstrap-Knoten

Beim initialen Verbindungsaufbau zum IPFS-Hauptnetzwerk im Zuge des Bootstrapping-Verfahrens spielt der Swarm keine Rolle. Technisch handelt es sich beim Bootstrapping um eine Routing-Anfrage an zumindest einen (vor)konfigurierten Bootstrap-Knoten, dessen IP-Adresse bekannt ist, Ziel der Routing-Anfrage ist im Bootstrap-Fall der Identifikator des eigenen Knotens. Auf Seiten eines kontaktieren Bootstrap-Knotens wird die lokale Routingtabelle nach dem (zu diesem Zeitpunkt nicht vorhandenen) Identifikator durchsucht und als Antwort eine Menge der dem gesuchten Ziel nächstliegenden Identifikatoren (und IP-Adressen) retourniert. Folglich genügt es aus Sicht eines Angreifers, bzw. einer Angreiferin, die Routingtabellen aller im IPFS-Hauptnetzwerk verwendeten Bootstrap-Knoten zu füllen, um so sicherzustellen, dass neu verbindende Knoten keine Informationen über das tatsächliche IPFS-Netzwerk erhalten können. Da IPFS Stand Mai 2020 eine kleine (<10) Anzahl statischer Bootstrap-Knoten verwendet, liegt dies auf Basis der vorgenerierten Identifikatormenge jedenfalls im Bereich des machbaren. Auf Nachfrage wurde von Protocol Labs die Konfiguration der Bootstrap-Knoten übermittelt. Dabei handelt es sich um reguläre libp2p-Instanzen, welche eine Grace Period von einer Minute verwenden, sowie eine untere Schranke von 1000 und eine obere Schranke von 2000 offenen Verbindungen. Auf Grund dieser eingestellten Schranken muss eine hohe Anzahl gemultiplexter Verbindungen offengehalten werden, um auch tatsächlich die gesamte Routingtabelle mit Sybil-Knoten füllen zu können. Grund dafür ist, dass zwar nicht mehr Punkte pro Sybil-Knoten benötigt werden, sondern mehr Sybil Knoten insgesamt durch Verbindungsmultiplexen einen höheren Punktestand erreichen müssen. Dabei handelt es sich jedoch nach wie vor um realistisch umsetzbare Ziele. Im Speziellen kann ein Bootstrap-Knoten nach dem anderen isoliert werden, da

- einmal in die Routingtabelle aufgenommen – Verbindungen insgesamt eine geringe Wahrscheinlichkeit haben, getrennt zu werden, auch wenn dies nicht auf alle Bereiche der Routingtabelle zutrifft. Ist einmal ein Zustand erreicht, in dem alle Bootstrap-Knoten isoliert sind, ist es lediglich eine Frage der Zeit, bis das gesamte Netzwerk ausschließlich zu von einem Angreifer,

bzw. einer Angreiferin betriebenen Instanzen Verbindungen aufbaut. Wie lange es dauert, bis dieser Fall eintritt, ist stark davon abhängig, wie häufig IPFS-Instanzen neu gestartet werden. Ungeachtet dessen ist die Barriere für diesen Angriff als verhältnismäßig niedrig einzustufen, da ein IPFS-Knoten Stand Mai 2020 keine Informationen über bestehende Verbindungen persistiert und somit nach einem Neustart lediglich Kontakt zu Bootstrap-Knoten herstellt.

## 4.3. Angriffskosten

Zusätzlich zu technischen Aspekten des Angriffs spielen in der Praxis insbesondere wirtschaftliche Gesichtspunkte eine Rolle, wenn es darum geht, abzuschätzen wie relevant dieser Angriff für das im Einsatz befindliche IPFS-Hauptnetzwerk ist. Ausgehend von Nutzerzahlen und der bisher generierten Wertschöpfung des Projekts, sowie des lukrierten Risikokapitals, kann IPFS auch für verhältnismäßig mächtige Angreifer attraktiv sein. Der im Rahmen dieses Projekts entwickelte Angriff verursacht jedoch kaum Kosten und ist für jedermann einfach nachvollziehbar und durchführbar. Insbesondere, wenn die für den Angriff benötigte Vorbereitungszeit nur eine untergeordnete Rolle spielt, und auf bereits vorhandene Hardware zurückgegriffen werden kann, entsteht (abgesehen von Stromkosten) lediglich für die Anschaffung des benötigten Speicherplatzes zur Speicherung von 29TB an Identifikatoren finanzieller Aufwand. Stand Mai 2020 beläuft sich dieser laut des Preisvergleichsportals *Geizhals.at* auf weniger als 600€ [10] ausgehend von einem Terabyte-Preis von weniger als 20€. Soll dedizierte Hardware für die Identifikatorgenerierung beschafft werden, bietet sich auf Grund des Preis-Leistungsverhältnisses ein System auf Basis einer *AMD Ryzen 3700X* Achtkern-CPU an, was sich Stand Mai 2020 mit rund 900€ [11] zu Buche schlägt. Insgesamt ergibt sich auf Basis dieser Modellrechnung ein Rahmenbudget von 1500€ (exklusive laufender Kosten) für die Generierung von Identifikatoren.

10 https://geizhals.eu/?cat=hdx&xf=5588_HDD 11 https://geizhals.eu/?cat=sysdiv&xf=10863_8%7E6764_AMD%7E6770_Ryzen+3000 Abgesehen von den beschriebenen Vorabkosten, entstehen im Rahmen des Angriffs keine weiteren Fixkosten. Die Durchführung des Angriffs verursacht jedoch laufende Kosten, da eine breitbandige, stabile Internetverbindung, möglichst ohne NAT oder Firewall benötigt wird. Folglich bieten sich Cloud-Dienste zur Durchführung an. Für die Im Rahmen dieses Projekts durchgeführten Evaluierungen wurde in diesem Zusammenhang auf eine VPS-Instanz von *Hetzner* zurückgegriffen, welche mit 0.031€ pro Stunde [12] (inklusive 20% MwSt.) in Rechnung gestellt wird. Auf Basis einer Erhebung vom Frühjahr 2020 [11] kann die Gesamtanzahl von gleichzeitig aktiven IPFS-Knoten, die eingehende Verbindungen akzeptieren, auf ca. 3000 abgeschätzt werden. Soll der in diesem Projekt entwickelte Angriff global auf alle diese Knoten ausgeführt werden, ergeben sich laufende Kosten von weniger als 100€ pro Stunde. Ausgehend von den durchgeführten Evaluierungen ergibt sich eine Erfolgsquote von über 75% nach einer Stunde, wobei diese Wahrscheinlichkeit bereits nach ca. 40 Minuten nicht mehr nennenswert ansteigt (siehe Abbildung 1). Betrachtet man die in Abschnitt 3.2.2. beschriebene primitive Umsetzung des Konnektivitätsmonitorings, genügt es nach erfolgreicher Isolation, vier aktive Verbindungen zu einem isolierten Knoten aufrecht zu halten, was die Kosten auf lange Sicht drastisch senkt.

Bedenkt man jedoch, dass die Umsetzung des Bootstrap-Prozesses auf einer kleinen Menge statischer Bootstrap-Knoten basiert (siehe Abschnitt 4.2.), ist es wirtschaftlicher, deren Routingtabellen zu besetzen und abzuwarten, bis sich alle im Netzwerk befindlichen Knoten neu zum Netzwerk verbinden, da die hierfür entstehenden laufende Kosten nahezu vernachlässigbar sind. Darüber hinaus ist es (im Gegensatz zum Swarm) möglich, den aktuellen Stand der Routingtabelle eines jeden einzelnen IPFS-Knotens abzufragen,

wodurch Unsicherheiten bezüglich des Angriffserfolgs ausgeräumt werden können. Genauere Informationen bezüglich der durchgeführten Evaluierungen und der daraus abgeleiteten Angriffskosten sind der im Rahmen dieses Projekts entstandenen Publikation zu entnehmen.

## 5. Fazit

Im Rahmen dieses Projekts wurde die IPFS-Referenzimplementierung in der Version 0.4.23 einer Sicherheitsanalyse unterzogen, welche eine Reihe von Schwachstellen aufgedeckt hat. Im Zuge dessen wurde auf Basis dieser Entdeckungen ein Angriff entwickelt, der es ermöglicht, das gesamte IPFS-Hauptnetzwerk, welches von Millionen täglich genutzt wird, zu nach Belieben zu segmentieren und zum Erliegen zu bringen, ohne dass dies hohe Kosten verursacht. Teilweise liegen die Schwachstellen in der konzeptionellen Ausrichtung des Projekts und dessen Idealen von freiem Informationszugang für jedermann (ohne zentrale Kontrollinstanz) begründet, andere Schwachstellen ergeben sich schlicht aus Implementierungsfehlern. Alle Erkenntnisse wurden an Protocol Labs kommuniziert, was bereits im April 2020 mit IPFS 0.5 zum Ausrollen eines signifikanten Sicherheitsupdates geführt hat. Darüber hinaus wurde der entwickelte Angriff im Sinne der Qualitätssicherung (u.A. in Form von Regressionstests) in die Continuous-Integration-Pipeline des IPFS-Entwicklungsprozesses aufgenommen, um die Anfälligkeit zukünftiger Versionen bereits während deren Entwicklung abschätzen zu können. Das übergeordnete Ziel ist in diesem Fall eine koordinierte Entwicklung umfassender Gegenmaßnahmen, um IPFS sowohl konzeptionell, als auch betreffend der Referenzimplementierung nachhaltig sicherer zu machen. Nähere Details zu den bereits ausgerollten Gegenmaßnahmen können der Publikation, welche im Rahmen dieses Projekts entstanden ist, sowie dem offiziellen IPFS-Blog [13] entnommen werden.

## Referenzen


[1] I. Baumgart und S. Mies, „S/Kademlia: A practicable approach towards secure key-based routing," in *2007 International Conference on Parallel and Distributed Systems*, 2007.

[2] P. Maymounkov und D. Mazières, „Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Peer-to-Peer Systems*, Springer, 2002, pp. 53-65.

[3] Ion Stoica et al., „Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and* 12 https://www.hetzner.com/cloud#pricing 13 https://blog.ipfs.io/2020-10-30-dht-hardening/ *Protocols for Computer Communications*, San Diego, ACM, 2001, pp. 149-160.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp und S. Shenker, „A Scalable Content Addressable Network," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, New York, Ny, USA, ACM, 2001, pp. 161-172.

[5] K. Pearson, „The Problem of the Random Walk," *Nature,* Bd. 72, Nr. 1865, 1905.

[6] B. Prünster, „Sichere Peer-to-Peer-Netze auf Basis von Selbstzertifizierung," 07 05 2018.

[Online]. Available: https://technology.a-sit.at/veroeffentlichungen/sichere-peer-to-peer-netze-auf-basis-von-selbstzertifizierung/. [Zugriff am 11 07 2018].

[7] J. R. Douceur, „The Sybil Attack," in *Peer-to-Peer Systems*, Berlin, Springer, 2002, pp. 251 260.

[8] Atul Singh et al., „Defending Against Eclipse Attacks on Overlay Networks," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, Leuven, ACM, 2004.

[9] B. Prünster, „Sicherheitsanalyse aktueller Peer-to-Peer-Netze," A-SIT, 19 05 2019. [Online].

Available: https://technology.a-sit.at/veroeffentlichungen/sicherheitsanalyse-aktueller-peer-to-peer-netze/. [Zugriff am 28 05 2020].

[10] B. Prünster, E. Fasllija und D. Mocher, „Master of Puppets: Trusting Silicon in the Fight for Practical Security in Fully Decentralised Peer-to-Peer Networks," in *Proceedings of the 16th International Security and Cryptography (SECRYPT)*, SciTePress - Science and Technology Publications, 2019.

[11] S. Henningsen, M. Florian, S. Rust und B. Scheuermann, „Mapping the Interplanetary Filesystem," 18 02 2020. [Online]. Available: https://arxiv.org/abs/2002.07747. [Zugriff am 28 05 2020].

[12] J. Benet, D. Dalrymple und N. Greco, „Proof of Replication," 27 07 2017. [Online]. Available:

https://filecoin.io/proof-of-replication.pdf. [Zugriff am 31 10 2018].