

Constructing Attack Scenarios for Attacker Profiling and Identification

Dr. Eric Cole

Abstract

This paper addresses the need for an intrusion detection system that can correlate information from disparate devices in order to create attack scenarios for the purpose of tracking, monitoring, and identifying attackers. It builds upon an existing attack scenario framework to provide an automatic method of attacker profiling. It also demonstrates the method in a real world case study for analysis.

1 Introduction

Over the past few years the number and sophistication of attacks has increased creating the need for more intelligent intrusion detection systems (IDS). Standard IDS no longer provides the capability needed to detect the more advanced category of attacks. There is a demand for intrusion detection that can correlate information to construct attack scenarios for the purpose of tracking, monitoring, and identifying attackers. Only by identifying actual scenarios will the systems be smart enough to be able to track an alert on advanced attacks. This paper provides a method to profile attacks and attackers by constructing attack scenario graphs. This paper uses the correlation framework provided by the paper *Techniques and Tools for Analyzing Intrusion Alerts* by Peng Ning, et. al[1]. The framework provides an excellent foundation to extend into attacker profiling and identification research.

During an attack, the attacker reveals information through their actions and behaviors embedded within packets received across a network that contains information about the attack that can be used as an approach to intrusion detection.[2] This information is used

to create a profile of the attacker, his techniques, means and methods of operation. This profile can be used on a larger, more global scale to identify attackers and predict attacker behavior. This paper focuses on the profile aspect and the corresponding benefits.

Section 1 provides an introduction with explanation of the problem and related solution space. Section 2 provides an overview of the formal framework used as a basis for this research. It presents the details of the framework notation and correlation technique. Section 3 provides an overview of the profile attributes used to create a fingerprint of an attack and attacker. Section 4 presents examples to demonstrate the use of correlation graphs in constructing attack scenarios for attacker profiling and identification. Section 5 details the modification of the formal framework to include attacker profile attribute information. Section 6 includes information on the necessary and recommended architecture for implementing this intrusion detection system. Section 7 presents a famous attack as a case study in the implementation of this intrusion detection method. Lastly, Section 8 provides a conclusion to the paper and future areas of research.

2 Constructing Attack Scenarios

The paper *Techniques and Tools for Analyzing Intrusion Alerts* by Peng Ning, et. al.[1] presents a correlation framework for constructing attack scenarios based on the prerequisites and consequences of attacks. It also presents a collection of analysis utilities that facilitate the navigation of large sets of intrusion alerts. The correlation framework and the analysis utilities have been implemented in a toolkit called Toolkit for Intrusion Alert Analysis (TIAA). [1]

This section provides the details of the formal framework, notation, and correlation technique.

2.1 Formal Framework Highlights

The framework correlates alerts on the basis of prerequisites and consequences of attacks. Prerequisites are the necessary condition for the attack to be successful and consequences are the possible outcomes of an attack. The framework matches the consequences of some prior alerts with the prerequisites of some later ones, thus constructing attack scenarios. Attack scenarios are the combination of steps that attackers use in their attacks. They show the logical connection between otherwise independent IDS alerts. Most attacks are related as different stages of attack sequences, with earlier attacks preparing for later ones. A complete attack is composed of many scenarios and each scenario can contain sub-scenarios. Attack scenarios are represented in a hyperalert correlation graph, which uses nodes to represent alerts and edges to represent the relationships between the alerts.[1]

2.2 Framework Notation

The framework notation represents prerequisites and consequences in terms of predicates. Predicates are basic constructs that represent the prerequisites and possible consequences of an attack. The *prerequisite predicate* represents the condition that must be present for an attack, for example:

$$\text{UDPVulnerableToBOF}(\text{VictimIP}, \text{VictimPort})$$

This prerequisite predicate represents the attackers discovery that a host at VictimIP runs a service at UDP port VictimPort that is vulnerable to a buffer overflow attack. The *consequence predicate* represents the possible results of an attack, for example:

$$\{\text{GainRootAccess}(\text{VictimIP}), \text{rhostsModified}(\text{VictimIP})\}$$

This consequence predicate shows that if the vulnerability is exploited the attacker will have

root access and the rhosts file will be modified. The framework notation uses conjunction (\wedge) and disjunction (\vee) to represent complex attacks where several conditions must be satisfied at the same time for the attack to be successful, for example:

$$\text{UDPVulnerableToBOF}(\text{VictimIP}, \text{VictimPort}) \wedge \text{UDPAccessibleViaFirewall}(\text{VictimIP}, \text{VictimPort})$$

This notation means that the prerequisite is that the host is vulnerable to the UDP buffer overflow and that UDP is allowed through the firewall for that victim and that port. Next, prerequisites and consequences are combined and represented as hyperalerts. Therefore, hyperalerts represent attack scenarios. A *Hyperalert Type T* represents the prerequisite and consequence of each type of alert. It contains the knowledge about a type of attack via three parameters (fact, prerequisite, consequence) described as follows:

- Fact – A set of attribute names, each with an associated domain of values, representing the information that is reported along with the alert.
- Prerequisite – A logical combination of predicates whose free variables are all in fact, representing what must be true in order for the attack to be successful.
- Consequence – A set of predicates such that all the free variables in consequence are in fact, representing what could be true if the attack succeeds.[1]

The following is an example of a hyperalert type T:

$$\text{SadmindBufferOverflow} = (\{\text{VictimIP}, \text{VictimPort}\}, \text{ExistHost}(\text{VictimIP}) \wedge \text{VulnerableSadmin}(\text{VictimIP}), \{\text{GainRootAccess}(\text{VictimIP})\})$$

A hyperalert instance h is generated if the attack is detected and reported by the IDS, for example:

$$\text{hSadmindBOF} = \{(\text{VictimIP} = 152.1.19.5,$$

VictimPort = 1235), (VictimIP = 152.1.19.7,
VictimPort = 1235)}

ExistHost (152.1.19.5) \wedge VulnerableSadmin
(152.1.19.5)

ExistHost (152.1.19.7) \wedge VulnerableSadmin
(152.1.19.7)

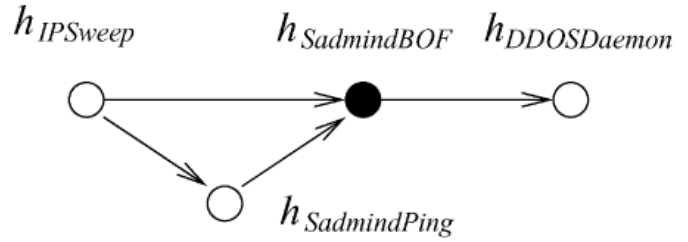
GainRootAccess (152.1.19.5), GainRootAccess
(152.1.19.7)

Notice that each of the three parts represents the three parameters of the hyperalert type populated with the necessary variable information. In this example two hosts were attacked via a sadmin exploit.[1]

2.3 Hyperalert Correlation

In order to build full attack scenarios, hyperalerts are then correlated to determine if any previous hyperalerts prepare for any later ones. In a sequence S of hyperalerts, a hyperalert h is a correlated hyperalert if there exists another hyperalert h' in S such that either h prepares for h' or h' prepares for h . Otherwise, it is an isolated hyperalert if no such h' exists. Stated another way, h_1 prepares for h_2 if some attacks represented by h_1 make the attacks represented by h_2 easier to succeed. The goal of correlation is to discover all pairs of hyperalerts h_1 and h_2 in S such that h_1 prepares for h_2 . Sometimes there may not be attacks that prepare for others. This is an isolated hyperalert, for example the instance of a single packet attack.[1]

Hyperalerts are correlated with each other and represented in a hyperalert correlation graph, which uses nodes to represent alerts and edges to represent the relationships between the alerts. Figure 1 is taken from the paper as an example:



(a) A hyper-alert correlation graph HG

Figure 1 - Hyperalert Correlation Graph

In this example $h_{IP\text{Sweep}}$ prepares for $h_{S\text{adminPing}}$ and $h_{S\text{adminBOF}}$ respectively, $h_{S\text{adminPing}}$ prepares for $h_{S\text{adminBOF}}$, and $h_{S\text{adminBOF}}$ prepares for $h_{DDOS\text{Daemon}}$. [1]

Although the concept of pre and post conditions exists in several other papers (referred to here as prerequisites and consequences) the formal notation of the hyperalert and its corresponding correlation graph are unique to this framework. In the hyperalert correlation graph the nodes are alerts and edges are relationships between alerts. They represent attack scenarios on the basis of the prepare-for relation and reflect the high level strategies of logical steps behind the sequence of attacks. The hyperalert correlation graph can potentially reveal the intrusion strategies behind the attacks and lead to a better understanding of the attackers intention. Second it can be used to profile previous attacks and thus identify ongoing attacks by matching to the profiles. This can be used for attacker profiling, identification, and forensics. In this paper hyperalert correlation graphs are used to create an attack fingerprint of the attackers techniques and methods of operation.

3 Profile Attributes

Hyperalert correlation graphs can be used for attacker profiling and identification based on attacker tools and sequences. An attacker may have a custom script to launch a series of attacks that he repeatedly uses or an attacker may have certain favorite attacks or sequences that he always uses. Either of these scenarios could be

used to fingerprint the attacker via his methods of operation. This type of profiling can also be extended to possibly predict attacks in progress, however that is beyond the scope of this paper. This profiling technique is easily incorporated into forensic analysis. Since the framework is focused on what has happened or is happening according to the alerts report by an IDS this information should be leveraged to track down and prosecute the attacker and to link the attacker with other incidents on a global scale.

The information that an attacker reveals about himself during the attack is used to build a profile of the attacker. The profile consists of attributes such as skill level, exploits used, files downloaded from the Internet, covering tracks, etc. These profile attributes are known as the attacker's methods of operation (MO). The MO is the summary of the habits, techniques, and peculiarities of the attacker that is used to identify the attacker and predict his behavior. The scope of behavior can be categorized as tactical, which is the behavior during the attack, or strategic, which is his overall goals and objectives (i.e. motives).[2] The method in this paper attempts to address both categories. The following lists some possible profile attributes for methods of operation and some examples of each:

- Exploits used (i.e. a favorite buffer overflow or set of exploits)
- Sequence of exploits attempted (i.e. could this be a script)
- Tools used (i.e. nmap, password crackers, rootkits, etc.)
- Techniques to cover tracks (i.e. erasing log file entries)
- Evasion techniques (i.e. did the attacker care about avoiding detection)
- Attack technique (i.e. port-scan followed by exploit or blind exploit. This is what is represented in the attack scenario correlation graph)

- Time considerations (i.e. duration of time the attacker spent on the device, network, attack, etc. Also includes time of day. This can also be used to predict the rate at which he works and when to expect him to be present.)
- External site communication (i.e. does the attacker access specific external sites to download tools and exploits. Does the attacker use certain proxies)
- Files changed (i.e. does the attacker always change certain files, for example, to create a back door)
- Files created (i.e. does an attacker always create certain files, for example, a directory in root named "...")
- Commands used (i.e. does the attacker use certain commands, and certain parameters for commands, or a favorite shell)
- Calling card (i.e. does the attacker leave a calling card such as a file on the system, website defacement, etc.)
- Attackers known accomplices (i.e. groups the attacker may be part of)
- Characteristics of the target (i.e. does the attacker always attack or favor a certain operating system)
- Misc. (i.e. any other peculiar habits)

The aggregate of the attacker's profile attributes can be used to identify him for the purpose of profiling, identification, and investigation.

4 Example Attack Scenarios for Profiling and Identification

The following examples demonstrate the use of hyperalert correlation graphs in constructing attack scenarios for attacker profiling and identification. Figure 2 represents a typical attack. The attacker performs an IP sweep and portscan to find a target. Then he attempts two different exploits. Exploit B worked and the attacker changes a file on the system to create backdoor access. The attacker contacts a site on

the Internet and downloads some tools, which allow him to install a rootkit. Lastly, he removes

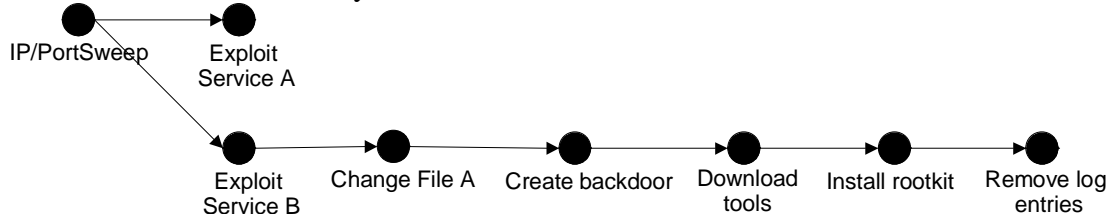


Figure 2 - Example Attack Scenario: Compromise

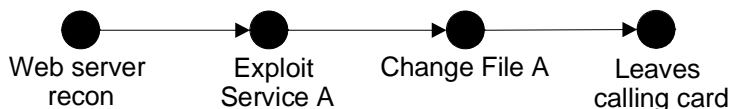


Figure 3 - Example Attack Scenario: Hactivism

The following profile attributes of this attack scenario are used to build the attacker profile:

- Tool used for the IP/port sweep
- Operating system of the target
- Exploits used
- File changed on the system
- Type of backdoor access
- Internet site used for tool download
- Tools downloaded
- Type of rootkit used
- Commands typed on the system
- Log entries that were removed
- Other miscellaneous profile attributes

The next example in Figure 3 is a hactivist who defaces a website. The hactivist already knows his target so he doesn't do an IP sweep. He does however perform some reconnaissance on the web server to investigate vulnerabilities. Next he uses exploit A and changes a file on the system, which is the index.html. He leaves his calling card as the hactivist message on the website. The following profile attributes of this attack scenario are used to build the attacker profile:

- Type of reconnaissance performed
- Type of exploit

the log entries to cover his tracks.

- Operating system of the target
- File changed on the system
- Message in calling card

5 Framework Notation with Profile Attributes

Due to the flexible and open notation of the hyperalert framework presented by [1], adding the profile attribute information into it is simple and straightforward. The profile attributes fit nicely into the *fact* parameter of a hyperalert type T. The fact parameter is a set of attribute names, each with an associated domain of values, representing the information that is reported along with the alert.[1] The following example hyperalert instances refer back to Figure 3 and include the additional profile attributes in bold:

```

RECON_WebServer = ({VictimIP = 192.169.10.10, VictimPort = 80, OS = WindowsNT, Type = BannerGrab, SourceIP = 10.10.1.200}, ExistHost (192.168.10.10) ^ HostIsWebServer (192.168.10.10), {InformationLeak (192.168.10.10)})
  
```

```

EXPLOIT_IISBoF = ({VictimIP = 192.169.10.10, VictimPort = 80, OS = WindowsNT, Type = IIS_DLL_BoF, SourceIP = 10.10.1.200}, ExistHost (192.168.10.10) ^ VulnerableIIS_DLL_BoF (192.168.10.10),
  
```

{ExecuteArbitraryCode (192.168.10.10),

GainRootAccess (192.168.10.10),

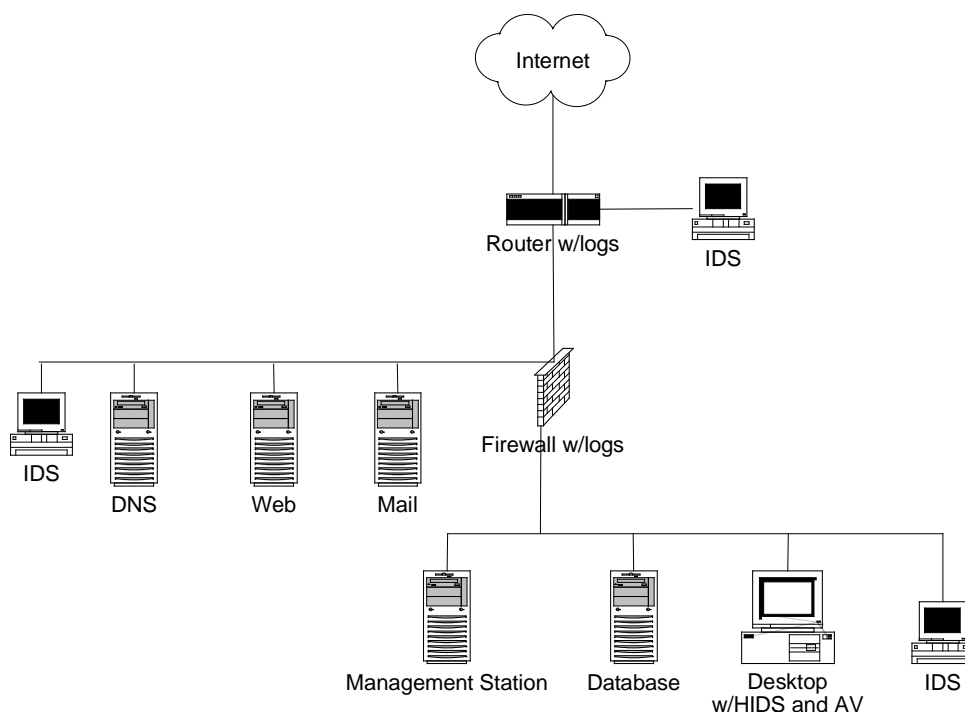


Figure 4 - Example Security Architecture for Alert Correlation

WebsiteDefacement (192.168.10.10))

SYSTEM_FileModify = ({VictimIP = 192.169.10.10, **OS = WindowsNT, File = index.html**}, ExistHost (192.168.10.10) ^ AdminAccess (192.168.10.10), {ViolateDataIntegrity (192.168.10.10), ViolateDataConfidentiality (192.168.10.10), WebsiteDefacement (192.168.10.10)})

SYSTEM_WebsiteDefacement = ({VictimIP = 192.169.10.10, **Message = DC_HaX0rs_Rule**}, ExistHost (192.168.10.10) ^ AdminAccess (192.168.10.10), {ViolatePublicIntegrity (192.168.10.10), ViolateDataAvailability (192.168.10.10)})

This hyperalert instance is a basic example of integrating profile attributes into a hyperalert. The parameters in bold are the profile attributes used to fingerprint and attacker. This flexibility allows one or many profile attributes to be included in a hyperalert.

6 Architecture

Security architecture is a critical aspect for correlating information in order to create accurate and detailed attack scenarios. The security architecture includes sensor placement and correlation of events from multiple sources. It involves correlating alerts from various sources, such as network and host based IDS. The framework should collect events outside the firewall, on the internal network, on the DMZ network, from router and firewall logs, from desktop HIDS and AV logs, from the management station, etc. Figure 4 shows an example of a security architecture that includes various devices that should be sending events for correlation. The method presented in this paper correlates events from both system and network devices. The success of the attacker profiling IDS relies on the number and location of sources of information and the completeness of the data.

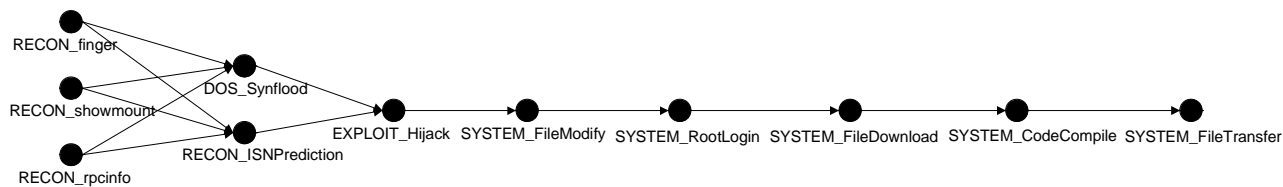


Figure 5 - Mitnick Attack Scenario Graph

7 Case Study: The Mitnick Attack

The case study used to illustrate the method presented in this paper is the famous Kevin Mitnick attack on Tsutomu Shimomura's network. The attack took place on Christmas day 1994 when Mitnick broke into Shimomura's computer and stole cellular phone software. Shimomura was a research scientist at the University of California at San Diego. The attack exploited the trust relationship between two computers on Shimomura's network. Mitnick faked the identity of the trusted host that he discovered during the reconnaissance phase. The real trusted host was silenced with a SYN flood denial of service (DoS) attack. He assumed the source IP address of the trusted host and hijacked the trust relationship allowing him to inject his own commands to create a backdoor, thus giving him full access to the compromised host.

Figure 5 shows an example hyperalert correlation graph of this attack scenario. The attack consists of four clear stages: reconnaissance, denial of service, exploit, and system. The first stage of the attack involved various finger lookups, showmount, and rpcinfo commands to the remote network. These probes came from the source toad.com. This reconnaissance information allowed Mitnick to discover the trust relationship between two systems. The next stage of the attack involves sending a SYN flood denial of service attack to the trusted host of the trust relationship. This silences the host so that it cannot respond when Mitnick assumes its IP address. The SYN flood originated from the non-existent IP address 130.92.6.97. While the SYN flood was in

progress Mitnick completed his reconnaissance by sending packets to the trusting host from apollo.it.luc.edu, a host that he likely compromised. The goal of the packets was to determine the predictability and pattern of the initial sequence numbers (ISN) used by the trusting host. Once the ISN was determined Mitnick moved on to the next stage of the attack, which was exploiting the trust. In this stage Mitnick created a blind connection to the trusting host, pretending to be the trusted host, which has been silenced. He was able to predict the proper sequence number so that even though he couldn't see actual replies he could still respond to them. The fact that it was Christmas day and there was little to no traffic on this network made this exploit successful in a short amount of time. During this connection Mitnick modified the `/.rhost` file by adding the line `"echo + +"`, which created a backdoor to allow him to connect again from anywhere. The last stage of the attack takes place on the system. Mitnick now has full control over the trusting host and logs in, downloads tools, compiles them, then transfers the code he was seeking.[4]

The following profile attributes of the Mitnick attack scenario are used to build the attacker profile:

Stage 1: Reconnaissance

- Commands used – finger, showmount, rpcinfo
- Command line options – finger -l, showmount -e, rpcinfo -p
- Source address – toad.com and apollo.it.luc.edu

Stage 2: DoS

- DoS exploit used –synflood port 513
- Source port range – 600+
- Source address – 130.92.6.97

Stage 3: Exploit

- Type of exploit – trust exploit
- Commands injected – “echo + + >>/.rhosts”

Stage 4: System

- Source address - unknown
- Commands used – finger, rsh, uname, file, head, whereis, gunzip, gzip, zcat, strings, egrep, env, diff, ccom, mkdir, tar, make, pwd, modload, modstat,
- Command line options – finger -l
- Directories accessed - /dev, /dev/tap, /ul/tsutomu
- Files changed - /.rhosts
- Tools downloaded - tap-2.01
- Download site used - unknown
- Files transferred - unknown

These are just some example profile attributes taken from online sources. Not all data was available, thus the “unknown” attributes. With further in-depth analysis more could be derived.[4,5]

8 Conclusion and Future Work

This paper presents a method to profile and identify attackers by analyzing attack scenarios and profile attributes. It demonstrates that an attacker reveals information about himself and the attack that can be used for intrusion detection by creating a profile of the attacker, his techniques, and methods of operation. A sample profile attribute list was presented as well as a case study that utilizes these attributes. Attacker profiles can be used on a larger, more global scale to identify attackers and predict attacker behavior. Incorporating this method with a global entity such as the Internet Storm Center,

www.isc.org, could provide valuable information for monitoring and tracking attackers.

Attacks are not always the result of a single attacker. Attackers often work in small, loosely-formed groups. When multiple attackers cooperate, a profile of the group can be used for intrusion detection. An area of future research that we intend to explore is profiling and identifying attacker groups by association analysis. Groups can be identified by individual attacker profiles and by an attackers known accomplices.

Another area of future research is to extend the method for predictive analysis. This includes predicting attacks in progress and predicting attacker behavior. There are several factors that assist with predictive analysis such as exploits used by the attacker are likely to be used again, the attacker may favor particular operating systems and vulnerabilities, and attackers tend to exhibit the same patterns. These attributes can be used to predict the outcome of an attack in progress and predict what an attacker may do next.

An area of future testing involves implementing this method as part of a penetration testing team efforts. This would provide a good environment to gather data and enhance the framework. Using the method during penetration testing will produce real world results to use for enhancements. The penetration testing team would also be able to answer questions and clarify the techniques they used.

Future work will include the exploration of each of these research areas as well as implementation, testing, and enhancing the framework.

9 References

[1] Ning, P., Cui, Y., Reeves, D., Xu, D. *Techniques and Tools for Analyzing Intrusion Alerts*. ACM Transactions on Information and System Security, Vol. 7, No. 2. May 2004. Pages 274-318.

[2] Yuill, J., Wu, S.F., Gong, F., Huang, M. *Intrusion Detection for an on-Going Attack*. RAID 1999 Proceedings. 1999.

[3] Allen, J., Christie, A., Fithen, W., McHugh, J., Pickel, J., Stoner, E. *State of the Practice of Intrusion Detection Technologies*. Networked Systems Survivability Program. January 2000.

[4]<http://www.nwo.net/security/texts/shimomur.txt>

[5]<http://www.takedown.com/evidence/state-analysis.html>