Amber: A Zero-Interaction Honeypot and Network Enforcer with Modular Intelligence

Adam Schoeman
Department of Computer Science
Rhodes University
Grahamstown, South Africa
adam@closehelm.com

Abstract—**For the greater part, security controls are based around the principle of Decision through Detection (DtD). The exception to this is a Honeypot, which analyses interactions between a third party and itself, while occupying a piece of unused information space. As honeypots are not located on productive information resources, any interaction with it can be assumed to be non-productive.**

This allows the honeypot to make decisions based simply on the presence of data, rather than on the behaviour of the data. But due to limited resources in human capital, honeypots' uptake in the South African market has been underwhelming.

Amber attempts to change this by offering a zero-interaction security system, which will use the honeypot approach of Decision through Presence (DtP) to generate a blacklist of third parties, which can be passed on to a network enforcer. Empirical testing has been done proving the usefulness of this alternative and low cost approach in defending networks.

The functionality of the system was also extended by installing nodes in different geographical locations, and streaming their detections into the central Amber hive.

Keywords-component; Honeypot; Security Models

I. INTRODUCTION

Antivirus, firewalls, intrusion detection/prevention systems and email filters, all share the trait that they implement their information security enhancements by analyzing the information that they are presented with.

Antivirus software uses signature repositories to compare executed code to known bad snippets of code, labelling them as viruses [1]. Firewalls traditionally define what ports and services are allowed to access a network node and block everything else [2]. Intrusion protection systems (IPS) use the same principal as antivirus, inspecting network traffic and attempts to match known bad sequences to incoming traffic [3]. Email filters make a decision on whether an email is spam based on numerous factors such as destination, source and content [4].

Each one of those basic information security controls has a list of predefined patterns or actions that have been classified as malicious, and when they detect them they take action based on the severity of the identified threat [5].

This process of decision-making can be referred to as Decision through Detection (DtD), because it requires the information security control to base its decision on the behaviour of the information flow that it is presented. If the behaviour matches a schematic that is seen as being malicious, the security control is triggered.

The process of building a behavioural database is intensive and prone to both false positives and negatives, as detailed in Cohen's Models of Practical Defenses Against Computer Viruses [1].

A honeypot, however, is an information security system that has no productive business use, because it occupies otherwise empty information space. It is normally classified as either low or high interactive, referring to the amount of maintenance that the honeypot needs to add value. Therefore any interaction with the honeypot is automatically suspicious, regardless of the behavioural traits of the interaction [6][7]. By not needing to take into consideration the behaviour of the analysed data flows, honeypots use a different decision system based on the presence of information and not the behaviour. Decision through Presence (DtP) is unique to honeypots by virtue of its links to the surrounding information resources.

Network Associate's Cybercop Sting, released 14 July 1998, is the earliest version of a commercial honeypot, originally referred to as a decoy system [8]. It offered consumers a way to audit and monitor intruders before legitimate data and data systems were at risk. Prior to that, Fred Cohen & Associates released a package called The Deception Toolkit. It provided an open source alternative to Cybercop Sting, and enabled the open source community to experiment with the idea of allowing intruders to compromise assets, using the interactions as a way of learning how to defend against them [9].

Cohens's toolkit formed a basis from which the community could actively start researching the field of honeypots as his system was a collection of Perl scripts. One such researcher was Lance Spitzner, founder of the Honeynet Project, and author of Honeypots: Tracking Hackers. Spitzner's book served as a formal foundation for honeypots. It contains real world examples of why honeypots are useful, as well as the

core concepts of honeypots: occupying unused and unproductive Internet space, with the intent of luring and learning about the actions of malicious attackers [6]. In his paper, To Build a Honeypot, Spitzner details the basics of building a server that can be used to record the steps and actions taken by a malicious intruder [10]. The server (or honeypot) would be separated from the valued network so that it could safely sustain damage and report back on how that damage was inflicted.

To Build a Honeypot led Spitzner and co-author Niels Provos to create a novel and simple way of deploying a honeypot: through Honeyd, a virtual honeypot daemon [7]. Honeyd can simulate a TCP stack as to further fool would be attackers, and can be configured with a number of templates, which mirror certain grades of common production servers.

This paper will attempt to bring the concept of honeypots into a working, security model that is both efficient and useful. The paper will introduce and dissect current security models, and then bridge them into a new model. Certain tests were then used to test the validity of the new security model.

Section II will introduce and give an in depth explanation of the two security models, as well as the two distinct phases within each of the models. Section III explains how the empirical test bed was designed to address the critical success factors, the technical design and implementation and the results of the test suite. After discussing the findings, the possibility of expanding the capabilities of the security control is evaluated and tested. Section IV acts as a summation of this research.

II. Decision throught Presence and Decision through Detection

A. Discovery Phase

The basic principle of this research is to construct a system that is able to combine the most useful pieces of the two security models, namely Decision through Detection (DtD), and Decision through Presence (DtP). While the drawbacks of antivirus and traditional firewall's DtD security model have been mentioned above, their advantages have outweighed their disadvantages, which is testament to their success over the years.

While it does require a great deal of human capital and technical infrastructure to build signatures, pattern files and a detection engine to run them [11], the model has the advantage of scaling perfectly as defensive nodes increase. Because any work or investment that is done on one defensive node can also be applied to another similar defensive node, the total cost decreases as nodes are added (1).

The only unit cost is that associated with the delivery of the signature to the node (1), which has become relatively small with the cost and speed of data transfer over the Internet [12].

$$Cost\ to\ Detect = (RC / n) + (DC * n) \qquad (1)$$

RC: research cost
n: number of nodes
DC: distribution cost

It is this scaling that has facilitated the commoditization of DtD products, and is the reason for their commercial success [13]. Distributing the cost of research amongst all the users of the DtD system also means that each individual node controller receives a certain degree of improvement to its security posture, without having to assign research and development resources to it. In contrast to this, systems that make use of the DtP security model have a cost of zero, because the evaluation is simply that of existence.

B. Action Phase

DtP loses its cost advantage in the second, Action, phase because its Discovery phase is based only on whether something had a presence, and not the nature of that presence. It is unknown at this stage, in a DtP model, if the information that is being seen is benign or malicious. The burden then falls on the Action phase of the security model to determine how to proceed with the data.

In the case of honeypots, the system owner would deploy the honeypot to gather information on possible new attacks, and use it as a research basis to develop a counter-attack or to modify existing defences [14]. The value of this kind of action is that the output is more significant to the organisation housing the honeypot because it is an information stream that is actively targeting their systems. But the cost equation for the DtP security model does create barriers.

As every context requires analysis to add any value, the cost of maintaining a DtP system as a going concern is extremely high, both in time and skilled labour, as shown in (2). Every context (*C*) of incoming information above a certain threshold would need to be researched to extract information and value out of it, and the process needs to be repeated for every node (*n*).

$$Cost\ of\ Action = (C * threshold) * RCperC * n \qquad (2)$$

C: information context
RCperC: research cost per information context
N: number of nodes

DtD, on the other hand, benefits from the higher cost of the earlier Discovery phase, because the signature that was used to detect the context has already decided on its nature, be it benign or malicious. But using centralized research does have an additional, intrinsic cost associated with it. As certain requirements need to be generalized for the signature to be significant to as many nodes as possible, consumers of centralised research pay an additional fee for the lack of specialization. This fee materializes as an increased rate of false positives, and is levied on each deployed node (3).

$$Cost\ of\ Action = n * FPRate \quad (3)$$

n: number of nodes
FPRate: False Positive Rate

C. Security Model Synopsis

The above equations show that both security models have competitive advantages, defined in terms of cost, in different phases. The DtD security model has the upper hand in the Action phase because most of the work has already been done in the earlier Discovery phase, but DtP is superior in the Discovery phase due to its simplistic criteria. To help minimise the cost in both phases, it would then be preferable for a system to utilise a hybrid security model that uses the Discovery phase from DtP and the Action phase from DtD where possible.

III. AMBER – TECHNICAL IMPLEMENTATION

A. Technical Design

A technical solution was prototyped that would aim to build a fully functional hybrid system to test how a system would react in a real world environment and with harsh constraints placed on it. The system was named Amber.

Following the security models previously discussed, Amber would need to employ a Discovery phase that is similar to the DtP security model approach (because of the low cost associated with it) but cannot use the DtP Action phase because of the prohibitive costs in that phase. The DtD model is preferable in the Action phase, but because it relies so heavily on its own Discovery phase, it is not possible to implement it on the back of a DtP based Discovery phase.

As superior as the DtD security model's Action phase is, it is also hampered by the cost to resolve false positive incidents. Any system based on this phase needs to aimed to reduce false positives to zero.

Lastly, Amber still needs to improve the overall security posture of the environment that it is connected to. The Critical Success Factors (CSF) for the technical system would be as follows:

- Zero-interactive system (reducing the research cost for both Discovery and Action phase to near zero)
- A false positive rate as close to zero as possible
- Improves security posture of the environment

B. Conceptual Implementation

1) Zero-Interaction

In order to fulfil the zero-interaction CSF, Amber needs to implement the DtP security model's Discovery phase, which requires the system to occupy unproductive Internet space. A suitable piece of Internet space was needed that served productive services to the Internet, but that also had unallocated (and thus unproductive) space.

A /24 IPv4 subnet which housed a group of web servers, each housing multiple websites was identified as the segment to which Amber would play custodian to.

An IP address that was not utilised by the web servers was selected and assigned to Amber as the non-productive Internet space that it would use to collect contexts. A webhosting subnet was chosen due to the low likelihood of any accidental neighbouring traffic, as websites are mostly accessed via their DNS name and not directly through the IP [15].

To maintain the zero-interaction CSF, a suitable Action phase had to be chosen that would lock into the DtP security model's Presence based Discovery phase, but that did not suffer from the high research costs of its corresponding Action phase.

The DtD security model Action plan was also not suitable because it relied too heavily on research done during its own Discovery phase, which is then not available as the DtP simple presence based Discovery phase was utilised.

But it is possible to use the core idea of the DtD security model's Action phase: leveraging off the work done in the previous phase. By identifying any traits during the DtP Discovery phase and determining which could be carried over into the Action phase, it was theorized that the cornerstone of a presence based Discovery phase (namely the idea that nothing should be interacting with the system as it resides on a non-productive Internet space) could also be used in the Action phase.

If an information context was entering the Amber system while it resided on a piece of non-productive Internet space, the context itself could be judged as also being non-productive, and that judgement could be carried over to the Action phase.

This theory allows Amber to use the rapid (and resource cheap) Discovery phase reasoning of the DtP security model in both phases, and links them together with the leveraging theory of the DtD security model. That is to say, it creates a hybrid security model, using the best of the DtP and DtD security models.

2) Zero False Positives

The second CSF is a symptom of generalized classification – a part of centralized research. Systems that use the DtD security model are prone to it because the Detection done during the Discovery phase is based on a set of generalized criteria that try to suit as many nodes as possible. Given that the Action phase depends heavily on the research done during the Discovery phase, actions taken in phase two are as prone to false positives.

As stated earlier, systems that use the DtP model are resilient to false positives because the Discovery phase is tailored to look at contexts that enter zones that have no productive use, and that are specific to the environment that they are attached to. It combats false positives because each context needs to be fully analysed as part of the Action phase, meaning every context requires manual intervention.

Amber loses the confidence built by analysing every context, but because the Action is based on the false positive

resilient Presence based Discovery phase, any actions taken in the Action phase would be based on an information context that had no productive use, making the probability of false positive extremely uncommon. To put it another way: given that all interactions during the DtP Discovery phase are unwarranted, it is unlikely for any action to return as a false positive in the Action phase, i.e. as non-malicious.

3) Improves the Security Posture of the Environment

The system is still an information security system, which means that through all its zero-interaction optimizations it still needs to improve the security posture of the environment that it is associated with. Without satisfying this requirement, any system would be useless.

As a standalone unit, Amber is not able to take action against anything that passes through the Action phase, because Amber does not use productive Internet space. As Amber is not attached inline or in front of, and does not share Internet space with, any productive entity, any action that Amber takes would only improve the security posture of the unproductive Internet space.

To overcome this Amber would need to rely on an external enforcer that is situated in front of a productive Internet portion of the segment and that could take action on Amber's behalf. Amber would log the source IP addresses of systems that targeted it and pass those IP's in the form of a threat stream to the enforcer (such as a firewall).

The firewall would then be able to drop packets that originated from the source IP addresses and thereby increase the security posture of the environment that the firewall was protecting.

In a real world situation the inclusion of a firewall would be ideal, but because Amber underwent testing as a proof of concept in a live environment, actively intervening or interfering with the environment would limit the ability to fully test the application.

As a proxy test, an inline system could be emulated by mirroring the uplink port on the switching device, and a packet sniffer could be attached to that port. The packet sniffer would be in a position to analyse all the data on the network, which would allow it to compare the threat IP address stream generated by Amber to all the traffic on the segment.

It would then be possible to see if an IP address attempted to access productive Internet space after it had been marked by Amber as being a threat. It can then be assumed that if an external network enforcer had been in place (such as a firewall) the IP address would have been hindered and the overall security posture of the environment would have been improved.

C. Technical Implementation

Amber was to be built on top of an established operating system to save time and resources, and would utilize that OS's available network stack and common packages wherever possible. Amber was written in a combination of Perl, Python and Bash depending on where the strength and development speed of each stage lay.

1) Discovery Phase: Listener

In order for Amber to adhere to a Discovery phase that is similar to that of a DtP security model, it would need the ability to open an arbitrary amount of TCP/IP ports and accept connections to those ports. The connection's source IP address then needs to be recorded for use in the Action phase.

A network daemon was written in Perl and deployed onto an Ubuntu 12.04 x64 server. It would spawn a set of common TCP/IP ports on the listening interface and record the source IP address of those connections. The follow common Internet services ports were chosen based on the potential attack surface that each represents [16]:

- 445/tcp | (ms-ds)
- 135/tcp | (ms-rpc)
- 3389/tcp | (ms-term-serv)

This introduced a problem into Amber that made it vulnerable to spoofing attacks [17]. If an attack knew the listening IP address of Amber, they could craft a packet with a source IP address that was not their own and send it to Amber. Amber would respond by classifying the attacker-chosen source IP address as a threat and streaming it to the network enforcers. This would allow an attacker to selectively deny a victim services that were protected by an Amber threat stream enabled network enforcer.

To combat this the Linux kernel firewall was used to detect any incoming transactions that successfully established a TCP/IP three-way handshake through the SYN, SYN-ACK, ACK process [18]. If an incoming connection managed to establish a three-way handshake then it would indicate that the source IP address provided is the actual IP address from where the connection originated.

While checking for the three-way TCP/IP handshake is a valid way to mark a connection as being established and not spoofed, an attacker would still be able to fool the method by guessing the correct SYN-ACK sequence number, as proposed by Bernstein in his paper on SYN Cookies [19].

Since then, Linux has by default enabled the proposed remedy for this, which are SYN cookies. But, in a paper by Dan Kaminsky, the author revisited the solution and noted that they are susceptible to a brute force attack due to the increase in possible Internet throughput [20].

It would take approximately eight million packets to successfully brute force a SYN cookie-protected system, which could take less than fifteen minutes on non-specialised hardware (assuming a packet rate of 100,000 packets per second).

To address this, Amber incorporates the netfilter packet limit module, which limits the amount of packets entering the system. By instituting a limit of 800 ACK packets per second an attacker would have to expend 2.2 hours of brute forcing to successfully guess a SYN Cookie. To further defend against brute forcing SYN Cookies, Amber's connection timeout was

set to 600 seconds. An attacker could effectively launch a Denial of Service attack against Amber because of the low packet per second limit, but because Amber is not attached to any productive Internet space, a DoS attack directly against it would not harm the segment.

2) Action phase: Streamer

Once the Listen has validated a source IP address, the Discovery phase ends and the Action phase starts. In order to implement the findings of the Discovery phase the IP address needs to be recorded and passed on. As Amber is unable to take action against IP addresses due to the restrictions placed on the environment in which it functions, the IP addresses are stored locally on Amber and compared to a window of segment traffic via the attached packet sniffer.

The packet sniffer uses a separate Ubuntu 12.04 x64 as its base operating system, and uses the tcpdump binary to capture all incoming traffic to the segment via a mirrored uplink port. A truncated version of the packet is captured because an analysis of the complete packet contents is not needed.

Amber would normally be active for 24 hours, during which time the packet sniffer would record all the traffic entering the segment. At the end of the 24 hour period, Amber's daemon would have logged all the validated source IP addresses locally, and this list would be exported to the packet sniffer where it would search through its packet captures over the same period.

For each IP address logged by Amber, the packet sniffer would check if the same IP address attempted to target another host on the segment, an action that would not have been possible if Amber's threat IP address stream had been connected to an upstream network enforcer. Therefore, if the packet sniffer finds a source IP address accessing another system on the segment after it has already come in contact with Amber, it is reasonable to say that Amber can possibly improve the security posture of the segment, if used in combination with an upstream network enforcer.

3) Results

The 24 hour long test was run a total of ten times over a six-month period from August 2012 to January 2013. Each source IP address that Amber validated and passed onto the packet sniffer was identified, on average, as accessing the rest of the segment 6.89 times.

Therefore, Amber has the ability to improve the segment security posture, as measured by the method described above.

Of the ports that the Listener opened the most frequently targeted was 3389/tcp, or Microsoft Terminal Services.

- 3389/tcp – 343 instances
- 4455/tcp – 199 instances
- 135/tcp – 50 instances

D. Adding modular intelligence

After proving that Amber had the ability to improve the security of a segment, the question was asked if the gains in security posture would scale proportionately to an increase in Amber nodes that were able to generate a validated source IP address threat stream. Unfortunately, due to limited IP address space on a segment and Amber's DtP-based Discovery phase needing Internet space that had no other productive use, increasing the amount of Amber nodes on a segment reduces the amount of usable Internet space for that segment.

A test was done where a second Amber node was added to the same segment, but the increased security posture (as measured by any additional validated source IP addresses that were seen by the second node, and which attempted to target another system on the segment) was less than 1%.

Based on this minor increase in security posture by adding a second Amber node, and each node requiring the conversion of productive Internet space to unproductive space, it was decided that the practice of additional Amber nodes on the same segment was impractical. However, if the trade off of productive to unproductive Internet space could be offset, then any increase in the security posture could be relatively significant. It was also theorized that sampling Internet space that was geographically removed from the first node, would yield a validated source IP address threat stream that was significantly different for the first node's stream.

Two additional Amber nodes were set up on geographical areas that were unrelated to the first. The original Amber node resided in South Africa (named za-amber), while node 2 and 3 were placed in the United States (named us6) and Germany (named de3) respectively.

Early on it became apparent that there might be a limited amount of correlation between threats recorded in these different locations. Reviewing the threat streams generated by us6 and de3 showed that none of their validated source IP addresses were appearing on za-amber's packet sniffer. A new, precursor hypothesis was devised: is there any correlation between attacks seen between the US, DE and ZA geographical regions? This would act as a sanity check to the hypothesis of using distributed nodes to increase the security posture of a single segment. This updated hypothesis would also be trivial to test, and if it were false, then the distributed node hypothesis would automatically be false, due to its reliance on a correlation between regions to exist.

It is impractical to monitor all the segments that the distributed Amber nodes live on, but a similar test for correlation could be achieved by comparing the source IP address streams generated by each node. Each node is built robustly and is able to function without intervention, reverting to a known trust state if a part of it fails, giving a high rate of listening coverage.

A central command and controller server was built that would be able to log into each node and pull down the source IP address threat streams, archive it locally, and compare each IP address to the full history of IP addresses across all managed nodes.

1) Findings

The correlation test was run over a period of three months, with each of the nodes operating continuously, minus some minor downtime due to hosting infrastructure. In addition to the original three TCP/IP ports being spawned by the listener, TCP/IP port 80 and 443 were used to gather data. These two ports were included to increase the potential connection surface. Since the objective was to measure correlation between the different geographic areas, incorporating these high traffic Internet ports [21] aids the objective without sacrificing any of the test's integrity.

While the use of caching servers have the potential to corrupt the tests that measure Amber's ability to improve the security posture of a segment (caching servers hide the source addresses for numerous Internet users), it does not interfere with this hypothesis.

The three nodes measured the following successful connections:

- ZA-Amber IP: 1132 IP addresses
- US6 IP: 3356 IP addresses
- DE3 IP: 557 IP addresses

Each source IP addresses stream, from each of the nodes was compared to the other two node's IP address streams. A match would indicate that a particular IP address was discovered by both nodes, and showed that there is a correlation between the nodes.

TABLE I. CROSS CONNECTIONS

Nodes	ZA-Amber	US6	DE3
ZA-Amber		0	3
US6	0		11
DE3	3	11	

The data in Table I shows that there was a very small set of IP addresses that were logged by two or more nodes. The German and United States based nodes logged eleven common source IP addresses, while the South African node only logged three IP addresses that also attempted a connection with the German node.

The US and South African nodes didn't share a single source IP address. Based on this investigation it would seem that there is no value derived from distributing Amber nodes in geographically different locations.

While these results are on a minor scale, it does invite the question as to whether or not there is any value in applying threat countermeasures in one region based on intelligence collected in a different region. There are numerous commercial products that offer this service [22] and while it was not possible to test this at the same scale as the large security vendors, it would seem that the ability of a small threat stream to be useful outside of its region is extremely limited.

IV. CONCLUSION

The two security models (Decision through Detection and Decision through Presence) both aim to improve the security posture of the assets that they are tasked to protect, but each manages to fulfil that custodianship with a different methodology and different costs. Despite their differences, each of the security models can be factorized into two distinct phases: the Discovery phase and the Action phase. Each model provides a distinct cost advantage in one of the two phases, thus bestowing a differing competitive advantage to both.

In the traditional implementations, DtP is able to assess a threat in the Discovery phase at a relatively low cost, but falls short in the Action phase due to the amount of intensive research that is required. DtD on the other hand requires a large amount of research in the Discovery phase. And while the research can be centralized and distributed to an infinite number of nodes, it does also increase the likelihood of false positives being detected as threats. Despite DtD's expensive Discovery phase, it has a superior Action phase as it leverages off the research completed in its Discovery phase.

Amber attempts to create a hybrid system that merges the strengths of both security models, while still improving the overall security posture of the assets under its custodianship.

Using the DtP security model's Discovery phase and the DtD security model's Action phase methodologies, Amber was able to improve the security posture of the environment that it was connected to, as measured by its ability to detect a source IP that would try to connect to multiple other hosts on the segment.

It was not practical to increase the amount of nodes on a single segment due to each node needing a piece of unused Internet space, but it was theorized that multiple nodes could be interconnected through a command and control server, if they were located in dissimilar geographically regions.

A preliminary correlation test was devised that tested whether there was any link between source IP addresses logged between the multiple nodes, before a test was done to see if the security posture could be increased by increasing the nodes.

The research yielded mixed results, showing that there was almost no link between source IP addresses logged by the different nodes. This means that there was no advantage to extending Amber's source IP address stream to include that of multiple nodes across different geographical regions. This also questions the usefulness of initiatives that attempt to apply threat data from one region onto other regions.

REFERENCES

[1] F. Cohen, "Models of practical defenses against computer viruses," Computers & Security, vol. 8, no. 2, pp. 149 – 160, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167404889900709

[2] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith, "Implementing a dis- tributed firewall," in Proceedings of the 7th ACM conference on Computer and communications security. ACM, 2000, pp. 190–199.

[3] R. Heady, G. Luger, A. Maccabe, and M. Servilla, The architecture of a network-level intrusion detection system. Department of Computer Science, College of Engineering, University of New Mexico, 1990.

[4] E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos, "Filtron: A learning-based anti-spam filter," in proceedings of the 1st conference on email and anti-spam, 2004.

[5] J. Oberheide, E. Cooke, and F. Jahanian, "Cloudav: N-version antivirus in the network cloud," in Proceedings of the 17th conference on Security symposium. USENIX Association, 2008, pp. 91–106.

[6] L. Spitzner, Honeypots: Tracking Hackers. Boston, United States: Addison Wesley, September 2002, ISBN: 0-321-10895-7.

[7] N. Provos, "Honeyd-a virtual honeypot daemon," in 10th DFN-CERT Workshop, February 2003.

[8] P. Newswire. (2012, October) Network associates ships cybercop sting - industry's first 'decoy' server silently traces and tracks hacker activity. Electronic. RP Newswire. New York, United States. [Online]. Available: http://goo.gl/jbvo7

[9] F. Cohen. (2012, 11) A note on the role of deception in information protection. Electronically. Fred Cohen & Associates. Pennsylvania, United States. [Online]. Available: http://all.net/journal/deception/deception.html

[10] L. Spitzner, "To build a honeypot," Whitepaper, 1999. [Online]. Available: http://www.spitzner.net/honeypot.html

[11] J. Dickinson, "The new anti-virus formula," 2005

[12] Y. Ma, Q. Yang, Y. Tang, S. Chen, and W. Shieh, "1-tb/s per channel coherent optical ofdm transmission with subwavelength bandwidth access," in National Fiber Optic Engineers Conference. Optical Society of America, 2009.

[13] D. Gallagher. (2013, 05) Mcafee earnings climb in second quarter. Online. MarketWatch. [Online]. Available: http://goo.gl/nv9fT

[14] Bernardo Machado David, João Paulo C. L. da Costa, "Blind automatic malicious activity detection in honeypot data," International Conference on Forensic Computer Science, vol. 6, pp. 142–152, 2011.

[15] L. Cherkasova, "Flex: Load balancing and management strategy for scalable web hosting service," in Computers and Communications, 2000. Proceedings. ISCC 2000. Fifth IEEE Symposium on. IEEE, 2000, pp. 8–13.

[16] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier, "Shield: Vulnerability-driven network filters for preventing known vulnerability exploits," in ACM SIGCOMM Computer Communication Review, vol. 34, no. 4. ACM, 2004, pp. 193–204.

[17] L. T. Heberlein and M. Bishop, "Attack class: Address spoofing," in Proceedings of the 19th National Information Systems Security Conference, 1996, pp. 371–377.

[18] J. Postel, "RFC 793: Transmission control protocol, September 1981," Status: Standard, 2003.

[19] D. J. Bernstein, "Syn cookies", http://cr.yp.to/syncookies.html, 1996.

[20] D. Kaminsky, "Black ops of TCP/IP 2011," in Black Hat USA 2011. Black Hat USA 2011, 2011, p. 44. [Online]. Available: http://www.slideshare.net/dakami/black-ops-of-tcpip-2011-black-hat-usa-2011

[21] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho, "Seven years and one day: Sketching the evolution of internet traffic," in Infocom 2009, IEEE, 2009, pp. 711–719.

[22] C. Alme and D. Eardly, "Mcafee anti-malware engines: Values and technologies", http://www.mcafee.com/us/resources/reports/rp-anti-malware-engines.pdf ,2010