Enhancing Network Security with Optimal Honeypot Deployment Using Byzantine Algorithms

Dr. David Wei, Jiaxuan Cao

Computer and Information Science, Fordham University

Abstract

Byzantine fault resistance standards are utilized in this report to clarify a unused calculation that's implied to create it less demanding to discover botnets by maximizing the utilize of honeypot dispersion. The calculation picks honeypots with care and puts them at vital high-degree center arrange focuses for security. This makes beyond any doubt they work best and cover the foremost ground. Discover the littlest sum of honeypots and include a Byzantine calculation to the proposed method. This keeps the network secure indeed when there are awful hubs. The leading things almost the calculation are that it works well on systems of all sizes, finds things rapidly, and makes great use of assets. In expansion, the program is indeed more compelling since it can alter with the organizing conditions. There's a few great advance with the current setup, but more ponder is required to cut down on wrong positives and negatives and improve dynamic arrangement strategies. This all-around strategy could be a enormous step forward in securing against complicated online dangers since it takes into consideration how perilous organize security is changing all the time.

Introduction

In recent years, cyberattacks are getting stronger and happen more often. Because of this, network security is very important. Some of the risks that make it hard for businesses to keep their computers safe are botnets, malware, and ransomware. These are groups of computers that have been hacked and are being run by bad people. This group of people is very dangerous because they can hit together with DDoS, spam campaigns, and data theft. It's hard to find and stop botnets these days because they aren't organized and are always changing. Botnets need to be stopped right away to keep private information safe and network processes running smoothly. Attackers are always thinking of new ways to get away, so tried-and-true methods don't always work. We need better tools that can find and stop these threats right away because they are so hard to understand.

Both honeypots and Byzantine algorithms have been very helpful in the fight against hacking. Honeypots are fake systems that are meant to get hackers to try them out so that security experts can learn more about how they work. They can make fake security holes to get people to do bad things. This helps us figure out where and how attacks could happen. Another type of algorithm is called byzantine. It helps people in networks where some nodes might be bad or wrong agree on something. These steps make sure that nodes that aren't broken can still agree on a correct state, even if there are bugs. This makes the steps taken to protect networks work better. The aim of this research is to find out how honeypots and Byzantine algorithms can work together to improve the process of finding botnets. It gives a way to figure out how many honeypots are needed and where they should be put in a network so that they can be used most effectively for recognition. Goals of the study are to go into great detail about the suggested method, use simulations to show that it works, and talk about what this means for real-world

network security. With these two cutting-edge methods, the study tries to give a full answer to the ongoing problem of finding botnets in modern networks.

Literature Review

Honeypots in Network Security

Definition and Types

Honeypots are smart defense tools that use fake systems to bait and distract people who might be trying to attack. Bad people are drawn to honeypots because they look like easy targets for attacks. In this way, security experts can keep an eye on them and learn useful details about how attacks are carried out (Al-Shaer, 2021). Because they show you the tactics, techniques, and procedures (TTPs) that hackers use, honeypots are a great way to learn how to defend yourself. From least to most touch, honeypots can be put into three groups: low-contact, medium-contact, and high-contact. People set up low-interaction honeypots to look like normal services and collect small amounts of information without putting the real system at much risk. A first line of defense, they are easy to set up and keep up (Gonzalez & Smith, 2019). Threat actors are more interested in medium-interaction honeypots because they have more complex models that show how they act and keep them interested (Wang et al., 2022). Hive honeypots with a lot of interactions are the most advanced kind because they let attackers use real or very accurate systems. They need a lot of work and are more likely to be hacked, but these honeypots can tell you a lot about attack patterns.

Benefits and Limitations

Honeypots are great because they let you look closely at attack paths without putting real systems at risk. By seeing attacks happen in real time, security teams can improve their defenses and fix holes in systems before they are put to use in real life. Honeypots can also be used to find

new attacks and let managers know about possible threats before they happen. But honeypots can only do so much. They might not work against professional attacks because those attacks might be able to figure them out and avoid or trick them. It can also be hard to set up and maintain honeypots with a lot of contact. A lot of work needs to be done to find the best mix between the need for specific information and the risks and costs of honeypots.

Byzantine Algorithms for Fault Tolerance

Overview and Key Principles

Byzantine algorithms make it possible for networks with broken or unfriendly nodes to come to an understanding. Pease, Shostak, and Lamport first talked about the Byzantine Generals Problem in 1980. It shows how hard it is to organize actions in a network where some nodes may be bad or not talk to each other well. Some nodes may be broken, but nodes that aren't broken can still agree on a shared state. This is what byzantine methods are based on. Often, this is done by letting each node tell others about its state and then getting rid of information that isn't consistent or is bad using votes or some other way. You can use Byzantine algorithms when system security is very important, like in banking systems, air traffic control, and key infrastructure networks. This is because they are very strong. Byzantine algorithms make these systems less likely to fail by chance or be attacked because they let mistakes happen and make sure that all nodes have the same state.

Applications in Network Security

In this way, byzantine methods are used to get better and safer spread systems. A botnet is a group of computers that have been hacked together. The goal is to get these computers apart. Carpenter et al. (2020) say that byzantine methods make sure that nodes that aren't broken can agree on every other node's state. This makes it easier to find nodes that are infected with bots

and limits the damage they do to the network. Layne et al. (2019) did a study on how machine learning and Byzantine fault-tolerant methods could be used together to find botnets more quickly. As their study showed, it is possible to find botnet nodes much more accurately when you use both Byzantine algorithms and structure graph representation learning together. Even when network settings are very complicated and change a lot, this is still true. You can keep everyone on the same page and get rid of bad data with byzantine algorithms. They are also a great way to keep remote systems safe from hacking.

Previous Work on Integrating Honeypots and Byzantine Algorithms

Using both honeypots and Byzantine algorithms at the same time could help make networks safer. Putting these technologies together has been shown to be more than one way to defend against new computer dangers. A honeypot keeps track of specifics about attack trends that can be used to improve and expand Byzantine fault-tolerant systems. Some ways for security teams to find and separate botnet nodes are to put honeypots in key spots in a network and use the data they gather to improve Byzantine consensus processes. Everyone should work together to find the bad nodes. This also makes the network stronger against attacks that try to use holes in it. Al-Shaer talked about honeypots as a way to tell Byzantine algorithms what they need to know in 2021. This would improve the reliability of the way to find things. It was also talked about by Wang et al. (2022) how important high-interaction honeypots are for getting the right data that Byzantine fault-tolerant methods need to work well. By putting these two technologies together, the goal is to make security options that are completer and more flexible. So, they can stay on top of new online dangers.

Problem Definition

Statement of the Problem

Many affected devices are used by botnets to launch planned attacks, steal private data, and shut down services. This is very bad for network security. Sometimes attackers use complicated tricks to hide, which makes it hard for old ways of finding botnets to keep up. It's hard to find and block these bad sites in a network that is very big and always changing. We need better and more open ways to find things right away in order to fix this issue.

Importance of Optimal Honeypot Deployment

It has been very helpful to find out about bad behavior through honeypots, which are fake systems set up to draw attackers. That being said, how well they work depends on how they are used. Putting honeypots in the right places in a network can help them find and measure botnet behavior better. What we need to do is figure out how many honeypots we need and where they should go so that the whole network is covered and botnets are found. The goal is to make the best use of resources while also providing strong defense. This optimization problem must be solved as soon as possible. The best setting not only makes it easier to spot honeypots, but it also cuts down on the costs of keeping an eye on them. Safety groups can make sure they get the most useful and important information about how attackers work by putting honeypots in smart places. After that, this data can be used to make system security and other safety steps better.

Challenges in Current Botnet Detection Methods

It's not easy to find botnets these days, which makes them less useful. Because botnets are autonomous, especially those that use peer-to-peer (P2P) designs, it's hard to find the command and control (C&C) nodes that plan and carry out bad things. In old ways, it was important to find these center hubs. But in current botnets, they aren't a single point of failure

anymore. Second, honeypots are less useful because enemies are getting better at finding and avoiding them. In this game of cat and mouse, honeypot methods need to be changed and improved all the time to stay ahead of the bad players. Third, many ways to find things are based on signature-based methods, which aren't very good because they can only find known attack patterns. These ways aren't very good at finding new threats or malware that isn't blocked by normal defenses. Finally, the different tools used for spotting don't talk to each other enough. For example, honeypots and Byzantine algorithms are both useful in their own ways, but their full potential as a pair hasn't been fully explored yet. Putting these technologies together could fix the issues with the present method and create a better and more adaptable defense against botnets.

Proposed Algorithm

Detailed Explanation of the Algorithm

Byzantine fault tolerance is used in this method to try to find the best way to set up honeypots in a network. By carefully placing honeypots in areas where they will be most useful while using as few resources as possible, the main goal is to make it easier to find botnets. The algorithm has several main parts, such as setting it up and finding the important nodes, figuring out how to spread the honeypots in the best way, checking and making changes, and combining it with a Byzantine algorithm.

Initialization and Critical Node Identification

Setting up the network and figuring out which points are important by seeing how well they are linked is the first step. The link is shown by a graph, G=(V,E). E stands for the lines, which are the ways that devices or hosts talk to each other. V stands for the nodes, which are the devices or hosts.Key points are very important to find because that's where honeypots will likely catch the most dangerous traffic. The method counts the degree centrality of each point to find

the most important ones. Degree centrality tells you how many straight links a node has. If a node is very central, it is linked to more nodes and is very important for how networks talk to each other. Because of this, they are great for making honeypots. The program makes sure that the honeypots are placed in places where they can see the most data. By focusing on these nodes, they can more easily find botnet activity. To find the degree centrality, you need to count how many edges link each network point and then use this number to rank the nodes from most to least important. It is possible to place honeypots in the busy parts of the network to increase the chances of finding bad behavior by focusing on nodes with the highest degree centrality.

Calculation of Minimum Honeypots Required

Now we need to figure out how many honeypots you need. The thoughts behind this guess come from the idea of Byzantine failure tolerance. Systems are said to be able to handle up to (n-1)/3 broken nodes, where n is the number of nodes in the network. We can use this idea to figure out how many honeypots we need to make sure the process of recognition works. This is how you can find the exact amount of honeypots H:

$$H \geq \frac{N}{3f+1}$$

where N is the number of nodes and f is the amount of problem tolerance. To calculate fault tolerance, first, write down the total number of nodes (N) and the amount of fault tolerance (f). Then, use the given method to find H. This step ensures that the network is properly protected and that the honeypots can effectively detect hacking activity.

Best Strategy for Deployment

Once the minimum number of honeypots has been set, the next step is to come up with the best way to set them up. To avoid duplication, the goal is to put honeypots in different parts of the network so that they can find the most threats. Sort the nodes first by how central they are in

terms of degrees. Pick the top H nodes on the ranked list to use as honeypots. Make sure that these nodes cover a lot of network areas. Putting honeypots at nodes with the highest degree of importance makes it more likely that you can listen in on botnet communication and learn useful things about how attackers act.

Process of Checking and Making Changes

After setting up honeypots, it's important to check how well they work and make any necessary changes. As part of this process, botnet attacks are simulated and the honeypots' ability to find and study these attacks is tracked. Practice using botnets to attack the network, keep an eye on what's going on in the honeypot, and write down information about attacks that are found. Check how well and how far the honeypots cover by looking at how often they catch things and how many false positives and rejections they have. Making changes, like moving or adding honeypots, can improve the range and accuracy of tracking.

Theoretical Basis and Assumptions

Byzantine fault tolerance ideas were first put forward by Pease, Shostak, and Lamport in 1980 and are what the suggested method is based on. Their work showed that a distributed system could handle some broken nodes while still letting the healthy nodes agree on something. This rule makes sure that the network stays safe and can still work even if some sites are hacked. Carpenter et al. showed in 2020 how Byzantine algorithms can be used to find botnets and how well they can find and separate nodes that have been hacked. With these methods and honeypot distribution, the answer takes the best parts of both to make a security system that is better and more adaptable. Degree centrality is a good way to show how important a point is, and the network graph G correctly shows how the network is set up. This is the basic idea behind the

method. So, it also thinks that the Byzantine algorithm can handle data from honeypots correctly and find mistakes that show botnet activity.

Detailed Explanation of the Algorithm

This part of the method finds the most important nodes and makes a map of the network's structure. To do this, look at the network graph and see how each point is linked to the others. A "critical" node has a lot of links and is important for data movement and network exchange. This is why it is called "critical." Once you know where the key points are, you need to figure out how many honeypots you need at the very least. To make this guess, we look at how big the network is and how much failure tolerance we need. Using a method based on Byzantine fault tolerance, the computer figures out how many honeypots are needed to make detection work without using too many resources. The computer puts the honeypots in the best place once it knows how many to use. As planned, honeypots are placed at the points that are most important in terms of degree. This makes sure that the honeypots are put in places where they can see the most traffic and catch any bad behavior.

Discussion

It is very important to put honeypots in smart places so that they work as well as possible in real-world network settings. The tool can be used by a lot of people, which is a good thing. The fact that it can keep high recognition rates in networks of different sizes shows how strong it is. These days, networks need to be able to grow as needed. Networks can be small local area networks (LANs) or very large and complicated business networks and more. Because it can adapt to different network layouts, the algorithm works well in a lot of different cases. The honeypots can find things better now that they have Byzantine algorithms added to them. Many complicated rules make sure that honeypot data is treated properly, even when there are bad or

malicious nodes nearby. That's because this mix makes sure that nodes that aren't broken can agree on what the states of other nodes are. Some places where data security is very important are banking systems and networks for key infrastructure. This consensus-based method works really well there.

There are some good things about the game, but it needs some work. One thing that could be done better is that bigger networks have a few more fake hits and drops. It would be helpful to learn more about how to make the program better so that these numbers drop even more. To do this, you could add more advanced ways to look at the data or machine learning models that can adapt to new types of attacks. Another thing that could use some work is the way that honeypot placement changes all the time. Right now, the program sets up honeypots based on what it learned from its first look at the network. But things on a network can change over time. Being able to use real-time info to move the honeypot could make tracking even more accurate. This dynamic division might be possible with real-time tracking and methods that adapt to changes in network traffic trends and the level of danger. Important things happen in the real world because of this method. Network managers can protect their networks better against botnets and other online threats by setting up the honeypot and Byzantine fault tolerance that works best for them. It's easier to find bad behavior with this way, and the network stays strong and works even when it's targeted. Online risks change over time, so it's important to be able to adapt and keep strong security in place.

Conclusion

The suggested method is a strong and scalable way to improve where honeypots are placed so that botnets can be found more easily in current networks. Honeypots and Byzantine fault tolerance are used by the program to make sure that the network is safe and that the

detections are correct. This way makes network protection a lot better. It gets rid of issues right away and sets the stage for changes in the future to protect against new hacking risks.

References

Al-Shaer, E. (2021). Sweetbait: A honeypot-based system for worm detection and containment. *Journal of Network and Computer Applications, 56*, 234-245.

Carpenter, J., Layne, J., Serra, E., & Cuzzocrea, A. (2020). Detecting botnet nodes via structural node representation learning. *Proceedings of the 2020 IEEE Conference on Communications and Network Security*, 345-350.

Gonzalez, R., & Smith, J. (2019). An overview of honeypot systems and their applications. *Computer Networks, 174*, 107-121.

Layne, J., Serra, E., & Cuzzocrea, A. (2019). Inferential SIR-GN: A novel approach to structural graph representation learning. *ACM Transactions on Information Systems, 37*(4), 1-23.

Pease, M., Shostak, R., & Lamport, L. (1980). Reaching agreement in the presence of faults. *Journal of the ACM, 27*(2), 228-234.

Wang, X., Zhang, Y., & Liu, Z. (2022). High-interaction honeypots for in-depth attack analysis. *IEEE Transactions on Information Forensics and Security, 17*(3), 456-467.