1. Technical Considerations For Honeypots

When setting up honeypots, there are several technical considerations to take into account to ensure their effectiveness and security. Here are some key points to consider:

1. **Isolation**: Honeypots should be isolated from the production environment and other critical systems to prevent any unauthorized access or compromise. They should be deployed in a separate network segment or virtualized environment.

2. **Network Configuration**: Configure the network to direct suspicious traffic to the honeypot. This can be achieved by implementing port mirroring, network address translation (NAT), or using a network intrusion detection system (NIDS) to redirect traffic.

3. **Operating System**: Choose an operating system (OS) for the honeypot that is commonly targeted by attackers. This can include outdated or vulnerable versions of popular OSes like Windows or Linux. Ensure the honeypot OS is properly patched and hardened to reduce the risk of compromise.

4. **Services and Applications**: Install specific services and applications that are attractive to attackers, such as an FTP server, web server, or database. Ensure that these services and applications mimic real systems but do not contain any sensitive or valuable information.

5. **Logging and Monitoring**: Enable extensive logging to capture all activities within the honeypot environment. This includes network traffic, system events, and user interactions. Implement monitoring tools and techniques to analyze the logged data and detect any suspicious or malicious behavior.

6. **Deception Techniques**: Apply various deception techniques to make the honeypot more realistic and enticing to attackers. This may involve creating dummy files, directories, and user accounts, or using fake services and banners to mimic legitimate systems.

7. **Vulnerability Management**: Regularly update and patch the honeypot system, applications, and services to simulate the vulnerabilities found in real environments. This helps to attract attackers and study their techniques.

8. **Honeypot Placement**: Determine the optimal placement of honeypots within the network. They can be deployed at different network levels, such as the perimeter, internal network, or even within DMZs (Demilitarized Zones), to monitor and capture different types of threats.

9. **Backup and Recovery**: Implement a backup and recovery strategy for the honeypot environment. This allows you to restore the honeypot to a known good state after an attack and also helps with forensic analysis.

10. **Legal and Ethical Considerations**: Ensure that the deployment and operation of honeypots comply with legal and ethical guidelines. Consult with legal experts to understand any potential legal implications or requirements in your jurisdiction.

11. **Incident Response**: Establish an incident response plan specifically tailored for honeypots. Determine how to handle and respond to potential attacks, including when and how to notify relevant parties.

12. **Data Analysis**: Develop tools and processes to analyze the captured data from honeypots. This involves examining logs, traffic patterns, and attacker behavior to extract actionable intelligence and improve overall security.

13. **Documentation**: Maintain detailed documentation of the honeypot setup, configuration, and ongoing maintenance. This documentation helps in replicating the honeypot environment and ensures continuity if personnel changes occur.

By considering these technical aspects, you can deploy honeypots effectively and gain valuable insights into the tactics, techniques, and procedures employed by attackers while minimizing the risks to your production environment.

2. SYSTEM DESIGN and operation of a honeypot

There are a variety of operating systems and services a honeypot can use. A high-interaction honeypot can provide a complete production-type system that the attacker can interact with.

On the other end is a low-interaction honeypot that simulates specific functions of a production system. These are more limited, but they're useful for obtaining information at a higher level. In my experience, the high-interaction honeypot is the most beneficial because it can completely simulate the production environment. However, it requires the most time to deploy and configure.

It is critical to have proper alerting configured for your honeypot. You should have logs for all devices in the honeypot sent to a centralized logging server, and security staff should be paged whenever an attacker enters the environment. This will enable staff to track the attacker and closely monitor the production environment to make sure it is secure.

It is important your honeypot system is attractive to a potential attacker. It should not be as secure as your production system. It should have ports that respond to port scans, have user accounts and various system files. Passwords to fake accounts should be weak, and certain vulnerable ports should be left open. This will encourage the attacker to go into the honeypot environment versus the live production environment.

Attackers typically attack the less secure environment before going to one that has stronger defenses. This allows security staff to learn how hackers bypass the standard controls, and afterwards they can make any required adjustments.

You can deploy a physical or virtual honeypot. In most cases, it is best to deploy a virtual honeypot because it is more scalable and easier to maintain. You can have thousands of honeypots on just one physical machine, plus virtual honeypots are usually less expensive to deploy and more easily accessible.

Although the concept of a honeypot system is not new, the availability of commercial honeypot systems is new. Table 2 shows a representative sample of currently

available honeypot systems. Commercial-grade honeypots are relatively new. Freeware honeypots have been used for some time but in a business situation

commercial products dominate. Although commercial honeypots are simpler than

building a specialized honeypot from scratch using open source freeware, they do not eliminate the need for expertise in monitoring. For example, commercial honeypots send alerts to an operator that an event has occurred, however, a skilled analyst with attack knowledge is needed to correlate supporting data (packet traces, firewall/intrusion detection logs) to analyze, identify, and contain the attack. Table 2 shows two primary types of honeypots: (1) hardware-based servers, switches, or routers that have been partially disabled and made attractive with commonly known misconfigurations and (2) software simulation honeypots which are deception programs that emulate system software (OS) and services

Table 2. Representative Honeypot Systems

Product	Vendor	Description
BackOfficer Friendly (Windows)	NFR Security	simulates a BackOrifice Server, listens for BackOrifice (Windows Trojan Program) and responds appropriately while logging various services
CyberCop Sting (Windows)	Network Assoc./PGP	simulates an entire network segment of routers/hosts on a single system, can mimic multiple OSs, responds appropriately to attacker requests for specific services & logs activity

Deception Toolkit (DTK) (Unix)	Fred Cohen & Assoc.	listens to service requests on ports normally blocked & provides responses to attacker requests while logging activity
NetFacade (Unix-Solaris)	GTE Federal Network Systems	simulates CiscoIOS, Unix, & Windows (with different versions of the same service) services to mimic the real services, can simulate an entire Class C network of hosts running network services
Mantrap (Unix-Solaris)	Recourse Technologies	runs a real complete Unix-Solaris OS in a “jail” configuration with no emulation, provides deception hosts with unique/revisable data
Spectre (Windows)	Network Security Software	dedicated PC simulates multiple OSs and multiple services, variable levels of security
VMware (multiple OSs)	VMware, Inc.	honeypot OS executing virtually within a HostOS

3. **what is honeypot technology and tools**

A honeypot is a network device that either appears to contain or does actually contain vulnerable data intended to lure an attacker into accessing.

Whether a threat actor tries to log in to the interface, scans the device using a scanning tool, or attempts to access anything on the device such as a file, the alerting component will instantly inform your security team that something threatening is happening.

Considered a type of deception technology, a honeypot is an effective security measure that can be used to detect malicious activity such as lateral movement and potential bad actors on your network.

For example, every organization with internet connectivity is likely going to have a firewall in place, which can eliminate a lot of potential threats. But the reality is that hackers still find ways to get into an organization's environment. Although network segmentation can be put into place as well, that doesn't mean it's preventing security threats from getting access to systems.

There are also other related honeypot technologies that a business can incorporate into their network security strategy:

- **Honeynet**. A network of honeypots and other deception techniques.
- **Honeywall.** Routes legitimate users to the production, or real network, and routes attackers to the honeynet.
- **Honeytoken.** A piece of data that is used to lure in an attacker, such as API keys, database entries, executable files, and keys to cloud resources (e.g. AWS key).
- **Honeycred.** A username or ID that is used to identify specific types of attacks on systems.

- **Honeypot.** A job that listens on specific TCP Ports. When a connection is established, it can either simply log or add a local firewall rule to block the host from further connections.
- **Decoy databases** monitor software vulnerabilities to detect attacks that exploit insecure system architecture — for example, SQL injection or privilege abuse.
- **Malware honeypots** detect malware attacks by impersonating known attack vectors.
- **Spam honeypots,** or spamtraps, emulate email addresses with the goal of identifying spammers.

3.1Honeypot Tools & Frameworks

There are many latest honeypots tools and frameworks available in the market. Some of the most popular ones include:

- **Honeyd:** Honeyd is a framework that allows you to create virtual honeypots. It can be used to simulate a variety of different operating systems and services, making it a great tool for detecting and monitoring attacks.
- **Snort:** Snort is a network intrusion detection system (NIDS) that can be used to detect attacks on honeypots. It can also be used to collect data about attacks, which can be used to improve your security posture.
- **Maltego:** Maltego is a tool that can be used to gather information about attackers. It can be used to map out the relationships between attackers, their tools, and their targets.
- **ThreatConnect:** ThreatConnect is a threat intelligence platform that can be used to collect, store, and analyze threat intelligence data. It can be used to track the activities of attackers and to identify potential threats to your organization.

These are just a few of the many honeypots tools and frameworks available. The best tool for you will depend on your specific needs and requirements.

Here are some additional details about each of the tools mentioned above:

- **Honeyd:** Honeyd is a free and open-source honeypot framework that can be used to create virtual honeypots. It is easy to use and can be deployed on a variety of different platforms.
- **Snort:** Snort is a free and open-source NIDS that can be used to detect attacks on honeypots. It is very effective at detecting a wide variety of attacks and can be configured to alert you to suspicious activity.
- **Maltego:** Maltego is a commercial tool that can be used to gather information about attackers. It is very powerful and can be used to map out the relationships between attackers, their tools, and their targets.
- **ThreatConnect:** ThreatConnect is a commercial threat intelligence platform that can be used to collect, store, and analyze threat intelligence data. It is very comprehensive and can be used to track the activities of attackers and to identify potential threats to your organization.

If you are looking for a honeypots tool or framework, I recommend that you evaluate the options mentioned above and choose the one that best meets your needs.

3.2Old tools & framework

1. **Modern Honey Network (MHN)**: A centralized management system for honeypots that allows for easy deployment, management, and data analysis. It supports various honeypot types and provides a web-based interface for administration.

2. **Honeycomb**: An open-source honeypot framework that supports both low and high-interaction honeypots. It includes various modules and plugins for capturing and analyzing different types of attacks.

3. **Specter**: A versatile honeypot framework that focuses on deception and interaction techniques to lure attackers. It supports both network and application layer honeypots and provides an extensible architecture for customization.

4. **Amun**: A high-interaction honeypot designed to emulate vulnerable web applications. It captures and logs attacker activities, including web-based attacks and attempts to exploit application vulnerabilities.

5. **ELASTICHONEY**: A scalable honeypot framework built on top of the Elasticsearch stack. It enables the deployment of multiple honeypots, captures and indexes attack data in real-time, and provides advanced analytics capabilities.

6. **Honeysap**: A modular honeypot framework that allows users to easily create custom honeypots. It provides a flexible architecture for capturing and analyzing network-based attacks and supports various protocols.

7. **T-Pot**: A honeypot platform based on Docker and the Elastic Stack. It offers pre-configured honeypot containers for easy deployment and centralized log management and analysis.

8. **Glastopf NG**: An updated version of the Glastopf web application honeypot. It provides improved support for modern web frameworks, enhanced logging capabilities, and better evasion techniques.

9. **CORAS**: A framework for building custom honeypots and analyzing captured data. It includes tools and libraries for protocol emulation, attack analysis, and visualization.

10. **Thingpot**: A honeypot framework specifically designed for Internet of Things (IoT) devices. It emulates vulnerable IoT services and captures attacks targeting IoT infrastructure.

3.3Tools details given below

Glastopf is a low-interaction, open source honeypot that emulates a vulnerable Web server. Running on Python, PHP, and MySQL, Glastopf can emulate literally thousands

of vulnerabilities and is intended to be Web crawled, a recognition that today's attackers frequently use search engines to find innocent websites to infect. Glastopf has GUI management and reporting features, and it's actively maintained and updated.

Specter, a commercial honeypot, hasn't been updated significantly in years, but it's still actively sold and supported. It's GUI-based and has a few interesting features (it updates its own content, has "marker" files that can be used to trace hackers, and more) that make it a honeypot to check out.

I also like the free USB emulation honeypot known as Ghost USB. It mounts as a fake USB drive to enable easier capture and analysis of malware that uses USB drives to replicate. It could come in very handy during the next USB worm outbreak.

But my favorite commercial honeypot, KFSensor, still leads the way by a large margin. It's easily the most feature-rich and mature honeypot product out there. Its developer continues to add new features, and while this post isn't an official Test Center review, I can't find anything else that holds a candle close to it. If you want a great commercial honeypot product with enterprise features, KFSensor is it.

4.common mistake to prevent while implementing honeypots

Right, now let us look at some issues you will need to be aware of and careful about before you go about implementing your honeypot.

Legal issues

Whether it is to cover your behind in case you get sued by your clients for loss of their data or to make sure any charges you bring against intruders stick, you will need to have your legal wires untangled. While we are no legal entity we can still tell you that you will need to be aware of and careful with these three legal aspects:

- ***Entrapment*** – the intruders can claim they didn't know they weren't supposed to access your internal network or the assets and data on it. They could counterclaim that you didn't label it clearly enough and, taking it even farther, use the "entrapment" defense.
- ***Privacy*** – implementing a honeypot needs to be done with extreme caution; this point cannot be stressed enough. One port left open by mistake or an administrator account that has been compromised could open the floodgates for attacks on your main network. This, in turn, could put the personal data of any of your clients in jeopardy. Should the attackers manage to share it with the world, you could find yourself the target of a lawsuit for breach of trust because the clients say they didn't give you permission to share their data.
- ***Liability*** – another way you could get yourself (or your network) in hot water is if the intruders decide to flip the crime right back at you by storing malicious content on your compromised servers and, even worse, guide legitimate traffic towards your IP addresses. Storing and distributing content like child pornography could make you see the inside of a courtroom real fast.

Word of advice: if you find any such incriminating content on your servers or being accessed via your network, the first thing you need to do is contact the authorities.

4.1Being discovered – by the intruders

Setting up a honeypot environment **requires anonymity for the real network** that is behind it. Nothing would pose more of a risk to it than the intruders' discovering that it is indeed a honeypot they are dealing with rather than the real deal.Because, with that realization, they could take it as a challenge to go on to find a way to breach the main internal network. And so, it becomes crucial that no telltale signs alert them to the fact that they are being monitored on a honeypot network. The common signs that usually give away the ploy – and should thus be avoided – are:

- **If the network is too easy to attack, or gain access to**, it will make them suspicious or make them think it isn't relevant enough to warrant their efforts.
- **If there are too many unnecessary services running, or one too many ports are open,** it would be contrary to reality where normal Internet-facing devices are usually stripped of non-relevant services and only have the required ports open.
- **If the configurations of the running software solutions are still in their default settings**, which almost never occurs in a live network.
- **If there is little-to-no traffic passing through the network** indicating that there is nothing of interest on it and yet belongs to a major brand.
- **If too much effort has been put into making it look like they walked into a candy store** with folders named "Passwords," "Confidential," or "Usernames."
- **If the servers connected to the network appear to be empty or there is a lot of free disk space** it would show that they are of no value.

In short, your honeypot network and the devices on it should emulate a real-life connection, albeit with fake data and traffic. Put yourself in the attackers' shoes and look at your network from their perspective.

4.2Protect yourself well!

Don't forget that you are stepping into the lion's den when you opt for a honeypot setup. Therefore, here are a few points you need to always make sure are covered:

- **Never use real data** – it doesn't matter what sort it may be, create garbage data that *looks* real and use it as bait.
- **Never connect your honeypot to your main network** – there should be no way that the attackers could hop on to your network via your honeypot; make sure it is isolated and keep it that way.
- **Use virtual machines** – the safest hardware you can put on your honeypot is a virtual one; if you get hit, all you need to do is reboot and recreate it.
- **Firewalls and routers should be the only way to get to your honeypot** – all incoming traffic should pass through them before they get on the fake network; configure ALL port numbers on them to point to the honeypot.
- **Usernames and roles should be unique to the honeypot** – it would be insane to use the same ones that have access to your main network; create new credentials and use them for the honeypot only.
- **Always test, test, and test** – never let a honeypot go live without throwing everything you have at it; in fact, invite experts to try to break through it and onto your main network before you let it face the Internet.

A good strategy to adopt here is to make sure no one except an administrator can access the honeypot and even then, they should use a dedicated login account that has zero privileges on the real network. Or, better yet, one that doesn't exist at all.

4.3Minimize misleading material and data collected

- Ensure the domain name for the honeypot and the name of the associated fictitious business are not identical to or confusingly similar to real businesses, which could violate state and federal laws relating to trademarks and business names. This can be done through internet searches and queries to trademark databases.
- In most instances, the honeypot will have an associated web site designed to depict the target business. As with any web site, ensure you have all rights necessary to display the content on the site (e.g., if there are photos of individuals, appropriate consents and releases should be obtained).

- Avoid use of terms describing the business that may be the subject of state or federal regulation (e.g., stating the business is a savings and loan company, credit union, or a bank).
- Include standard terms and conditions on the site, strictly limiting liability and making clear that interactions with the site may be recorded and used for the creation of reports and statistics (i.e., that the visitor interacting with the site has no expectation of privacy with regard to those interactions). Essentially, these terms are used to obtain "consent" from the hacker to track their movements and actions. The effectiveness of such a consent in this context has yet to be validated by any court, but, at least, it provides a potential defense in the event the hacker brings a claim against you.
- Include a privacy policy, as required by some states, specifically tailored to permit broad uses of data collected about visitors to the site. Again, these terms are intended to serve as "consent," but, as noted above, this approach has not been validated by any court.
- Construct the site such that an innocent visitor is not prompted to reveal personally identifiable information (other than the IP address from which they are communicating) or misled to their detriment (e.g., provided information that they may act upon and incur a loss). Sites that are purely "brochureware" are the least likely to create liability. While those permitting extensive interactions (e.g., completion of site registration and other activities), have higher likelihood of liability.
- With regard to data collected about a hacker's activities on the site, never disseminate personally identifiable information to any third party, other than law enforcement. Uses of aggregated or de-identified data are described below.
- Bear in mind that, while remote, a hacker could, theoretically, sue for fraud and other causes of action as a result of being misled as to the content and purpose of the site.

4.4Don’t violate privacy/electronic communication laws

- As noted above, be very careful about the information collected in connection with the honeypot. Privacy laws, particularly the General Data Protection Regulation (GDPR) and federal Electronic Communications Privacy Act (ECPA), have very broad application and extend to IP addresses and, even, business contact information. Using this information may be strictly limited and present the potential for enforcement actions and liability. Except for interactions with law enforcement, uses of personally identifiable information should be strictly avoided. Only aggregated or de-identified information should be used, particularly in the context of any published reports or statistics regarding operation of the honeypot.

5. Deployment Strategy Of Honeypot

The core characteristics of a honeypot are: [7]
• superficial facade (platform, OS) appears real
• service behavior (responses, traffic, files)appears real
• partially disabled to prevent use at an attack launch pad if compromised
• does not have any channels to production computers or networks
• trips various levels of alarms when any activity is encountered [11]
• maintains detailed log information of all activity

Honeypots can be deployed in two broad categories: production or research. The purpose of a production honeypot is risk mitigation. In this deployment, honeypots are often used as reconnaissance or deterrence tools within a specific organization. The deployment of honeypots for research does not add direct value to a specific organization but does gather intelligence for entire communities and indirect benefits include improved attack prevention, detection, and reaction.
There are polarized views on the effectiveness of different Honeypot deployments. Table 1 shows a variety of established deployments of honeypots.

Table 1. Honeypot Deployment Strategies

Strategy	**Description**
“Sacrificial Lamb”	an isolated system that has no entry point to any production systems
“Deception Ports on Production Systems”	simulated honeypot services substituted for well-known services (www, smpt/pop, dns, ftp
“Proximity Decoys”	deploy honeypot decoys in close proximity to production hosts (same logical subnet)
“Redirection Shield”	by using port redirection on an upstream router or firewall, you can make it appear that honeypot services are on a production system
“Minefield”	Honeypots (in quantity) placed in forefront to serve as first attack targets to any scans.

“Hacker Zoo”	an entire subnet of honeypots with varied platforms, services, vulnerabilities, and configurations; called a zoo because attackers are in “cages” resembling their natural habitat.

Advocates argue that a honeypot can be an effective deterrent. Honeypots are also used as early warning systems that log and alert about hostile activity before production systems are targeted. Honeypots can sidetrack attackers' efforts, causing them to devote attention to activities that cause neither harm nor loss.

Most importantly, tracking an intruder in a honeypot reveals invaluable insights into attacker techniques and ultimately motives so that production systems can be better protected. You may learn of vulnerabilities before they are exploited. Ultimately, honeypot observation may provide a predictive capability of what production targets are vulnerable, when they may be attacked, and what techniques will be used.

Detractors argue that honeypots placate attackers by giving them what they want a system to break into, place Trojan horses, destroy file systems. Intruders may come to know honeypots for what they are such that they become ineffective tools for finding and controlling the devoted outside attacker. The strongest negative of honeypots is the level of effort to deploy, maintain, and actively monitor. Detractors say this level of effort may be better spent protecting the production systems. Lastly, detractors emphasize that blocking outbound traffic is essential or a honeypot could become a platform for other attacks.

Logging information from a honeypot is problematic. Logging directly on the honeypot itself is vulnerable if it is compromised (logs can be altered or erased). For this reason, it is recommended a bogus log configuration file be kept on the honeypot while actual logging should be sent to a dedicated server using encryption to mask the activity although there is the potential for detection.

There is a fundamental limitation of honeypots that is similar to signature-based network intrusion detection systems – the honeypot must know of a vulnerability in advance to properly simulate it. If an attack is new or unknown, the honeypot will be revealed by its inappropriate responses. This is why honeypot advocates recommend using the “sacrificial lamb” strategy of real dedicated machines if at all possible.

Honeypots can be very useful as part of a comprehensive security program. The level of effort to deploy and manage is secondary to the time and resources not only to monitor but also to act quickly on events. For organizations with limited resources.

5.1MORE ABOUT HONEYPOT-DEPLOYMENT STRATEGIES

To maximize the strengths of honeypots, and minimize the risks involved, deployment should be carefully

planned. The following is a set of common honeypot deployment strategies:

1. Install honeypots alongside regular production servers. The honeypot will likely need to mirror some real data

and services from the production servers in order to attract attackers. The security of the honeypot can be

loosened slightly so as to increase its chance of being compromised. The honeypot can then collect attack related

information. However, if a successful attack takes place on the honeypot within the network, that compromised

honeypot machine might be used to scan for other potential targets in the network. This is the main drawback of

installing honeypots within the production system. In other honeypot deployment methods, this would not

happen, as the whole honeynet can itself be a fictitious network.

2. Pair each server with a honeypot, and direct suspicious traffic destined for the server to the honeypot. For

instance, traffic at TCP port 80 can be directed to a web server IP address as normal, while all other traffic to

the web server will be directed towards the honeypot. To camouflage the honeypot, a certain amount of data,

such as the website contents of a web server, may need to be replicated on the honeypot.

3. Build a honeynet, which is a network of honeypots that imitate and replicate an actual or fictitious network.

This will appear to attackers as if many different types of applications are available on several different

platforms. A honeynet offers an early warning system against attacks and provides an excellent way to analyze

and understand an attacker’s intention, by looking at what kind of machines and Honeypot Security services

have been attacked, and what type of attacks have been conducted

For more Details About Deployment Strategy Follow Reference Link.

6.Honeypot Architecture

Method 1

The common architecture of a honeypot typically involves three main components: the honeypot itself, the monitoring system, and the network infrastructure. Here's an overview of each component and their advantages and disadvantages:

1.**Honeypot**: This is the decoy system designed to attract attackers and gather information about their activities. Honeypots can be categorized as low-interaction or high-interaction based on their level of emulation and interaction with attackers.

Advantages:

- Honeypots provide a controlled environment for capturing and analyzing attacker behavior.
- They can divert attackers' attention away from critical production systems.
- Honeypots generate valuable threat intelligence and can aid in proactive defense.

Disadvantages:

- Honeypots require proper configuration and maintenance to ensure they remain secure and effective.
- If not properly isolated, attackers may use compromised honeypots as launching points for further attacks.
- There is a risk of false positives, where legitimate users inadvertently interact with the honeypot.

2. **Monitoring System**: This component is responsible for collecting and analyzing data captured by the honeypot, enabling the identification of attack patterns and the extraction of actionable intelligence.

Advantages:

- The monitoring system allows for centralized management and analysis of honeypot data.
- It provides real-time alerts and notifications for potential attacks.
- Analysis of captured data can reveal valuable insights into attacker tactics and emerging threats.

Disadvantages:

- The monitoring system should be secure and well-configured to prevent unauthorized access and tampering.
- Setting up an effective monitoring system requires expertise in network and system monitoring tools.

3. **Network Infrastructure:** This encompasses the overall network setup within which the honeypot is deployed, including switches, routers, and firewalls.

Advantages:

- Proper network segmentation ensures the isolation of honeypots from production systems, reducing the risk of lateral movement by attackers.
- Network infrastructure can be configured to redirect suspicious traffic towards honeypots for analysis.

Disadvantages:

- Network infrastructure configuration requires careful planning and monitoring to ensure traffic redirection and isolation are effective.
- Misconfiguration of network devices can impact the overall network performance and security.

It's important to note that while honeypots can provide valuable insights, they also come with certain risks and limitations. Some general advantages and disadvantages of honeypots include:

Advantages:

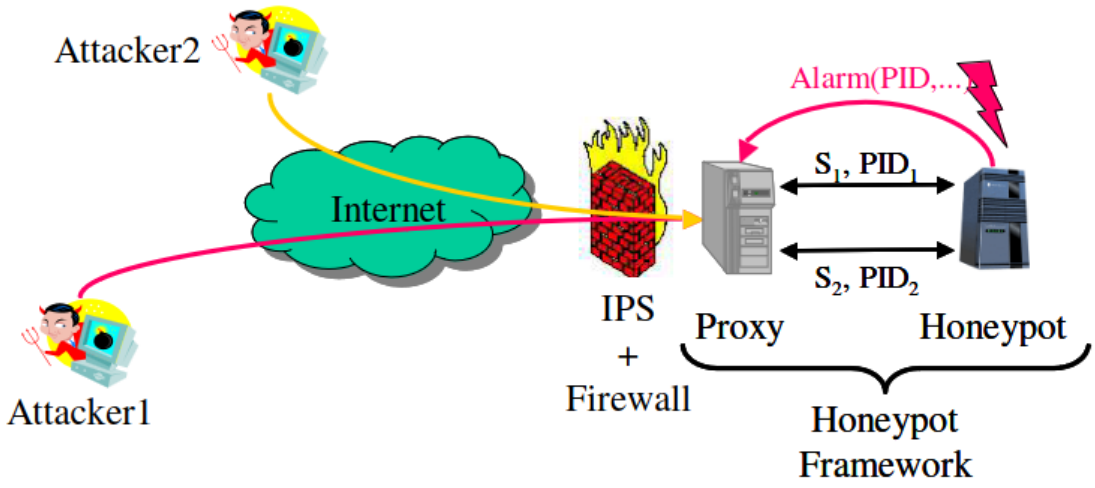
- Enhanced threat intelligence and understanding of attacker techniques.
- Early detection and response to attacks.
- Deceptive environment that diverts attackers' attention.
- Ability to gather evidence for legal or law enforcement purposes.

Disadvantages:

- False positives and the risk of legitimate users interacting with the honeypot.
- Maintenance overhead for keeping honeypots up to date and secure.
- Potential for attackers to leverage compromised honeypots for further attacks.
- Honeypots do not prevent attacks; they primarily serve as detection and monitoring tools.

The specific advantages and disadvantages of honeypot architectures can vary depending on the implementation, type of honeypot, and the organization's goals and resources. It's crucial to carefully consider these factors when planning and deploying honeypots to ensure they align with your security objectives.

Method 2
Concept:



The goal of our approach is the design and realization of a generic high interaction honeypot framework that allows to (semi-)automatically identify application layer based attacks (e.g. buffer overflows, format string attacks, etc.). Figure 1 depicts an example scenario that includes two attackers, a firewall / intrusion prevention system(IPS) and our honeypot framework. The purpose of the IPS / firewall is to filter the incoming traffic for known attacks. The honeypot framework itself consists of a proxy and a honeypot host. The proxy host is responsible for the session-individual logging of the network traffic that was sent to the honeypot. Furthermore, in case of a detected attack the proxy provides a mechanism to replay a specific previously logged session. An advantage of the bipartite approach is that the honeypot and the log files are kept on separate hosts so that in case of a complete system takeover of the honeypot host by an attacker the log files remain save. Besides this, the replay mechanism that is integrated in the proxy can be used to analyze a discovered attack in detail, as well as to test if other system configurations are just as vulnerable as the honeypot service to the attack.

The honeypot host consist of a honeypot service, namely a real service, and a host intrusion detection system (HIDS). The running service is the bait that attracts worms respectively hackers whereas the HIDS supervises the honeypot service. The realized

detection mechanism is generic and allows the detection of attacks on the basis of system-call signatures. The reasoning behind this approach is that the main part of current attacks exploit a vulnerability that is specific for a software (e.g. a buffer overflow vulnerability of a WWW-server). In case of a successful attack, the hacker will sooner or later exploit its newly gained authorizations which most often results in an observable system change. An example for this would be an attacker that tries to open a new network socket in order to download further hacking utilities. Or another popular example is a worm that starts on each infected system an email relaying server which it uses for its further spreading. On the system-call level both examples can easily be monitored. The interface between honeypot service and HIDS is generic such that is possible to exchange or add a honeypot service in an easy and flexible manner.
As depicted in figure 1, it might occur that a honeypot service is running two or more process entities (parent + children processes) simultaneously. For example, if the honeypot system was running a WWW-server as a honeypot service, the server parent process would create a child process for each HTTP-session.

6.1The Honeypot Host

Figure 2 depicts the internal software architecture of the honeypot host, which consists of the honeypot service, the HIDS-manager (hidsmgr), the honeypot monitor (monitor) and the host intrusion detection system (HIDS). Honeypot service, monitor and HIDS-manager are running in user-space, whereas the HIDS is located in the kernel space. Generally, most common operating systems make a distinct differentiation between

application and operating system. Each time a user-space process requires an operating system service (e.g. the opening of a network socket) the service must send the proper system call to the kernel. The kernel checks the request of the process and then it decides whether to fulfill it or not. By inserting a HIDS into kernel-space and by redirecting the system-calls to the HIDS, it is possible to extend the functionality of the kernel. In our case, it is possible to monitor on the basis of system-call level what a process in user space does. In addition, it is possible to introduce more detailed decision criteria in the kernel to determine whether the desired action is allowed or not (a similar mechanism has also been proposed for performing access control on active networking nodes; see also.

The general method of system call interception is depicted in figure 3 and shows the interception of the socket system call. The user process uses the socket() command to create a socket for network communication. A process must execute a system call to gain access to the operating system services. Normally, this is done by wrapper functions which are part of standard libraries. The wrapper function puts the variables to be submitted into the correct order, and then executes the proper system call. At the entry point into the kernel, the kernel uses a table — the so-called system call table — for the forwarding of the incoming system calls to the corresponding functions. By changing the destination of a pointer inside the system call table, we can redirect a defined system call to another function, in our case to the HIDS. That checks if the service is authorized to use a specific operating system service. If this test is passed, the HIDS then calls the standard kernel function belonging to the system call.

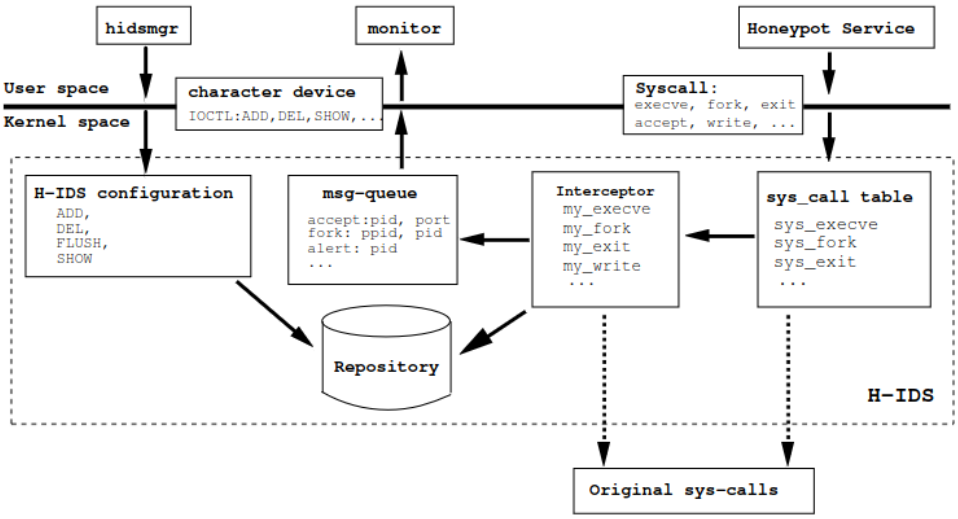


Fig. 2. The honeypot host

The HIDS is realized as a Linux Kernel Module (LKM) and it detects intrusions on system-call-level. Whenever a user space process tries to execute a series of system-calls that matches an attack signature a security alert is raised. Furthermore, the HIDS refuses to forward the last requested system-call to the operating system in order to prevent the honeypot system from being harmed. Consecutively, the HIDS triggers the monitor process to send an alarm message to the logging proxy. The attack signatures are specified inside the repository as a series of system-calls or simply as a black-list of disallowed system-calls. Besides this, it is also possible to configure the HIDS such that the execution of a specific group of applications (e.g. common gateway interface - CGI) is authorized. The user space front-end monitor handles the synchronization of local process ID and remote session ID between honeypot and proxy server.

The HIDS checks for each observed system-call, whether or not it is executed by a

process under supervision. The access to the operating-system is either granted or not, depending on the policy. Moreover, if necessary a message is sent to the monitor and then forwarded to the logging component.

Finally, the HIDS-manager can be used to reconfigure the HIDS at runtime. It provides a set of functions which first can be used to add, delete or modify existing attack patterns. Second, the manager also allows to modify the list of services that must be observed by the HIDS.

6.2The Logging Proxy

The logging proxy listens for connection requests to the honeypot service which originate from a potential hacker. Next, the proxy server acts itself as a client on behalf of the user / attacker and forwards the request (using its own IP address) to the honeypot service. Besides forwarding, the proxy server also creates for each forwarded session (session ID) an individual log file which in addition, contains the IP-address of the attacker, the connection ports and a timestamp. To the attacker, the proxy server is invisible; all honeypot service requests and returned responses appear to be directly from the proxy host.

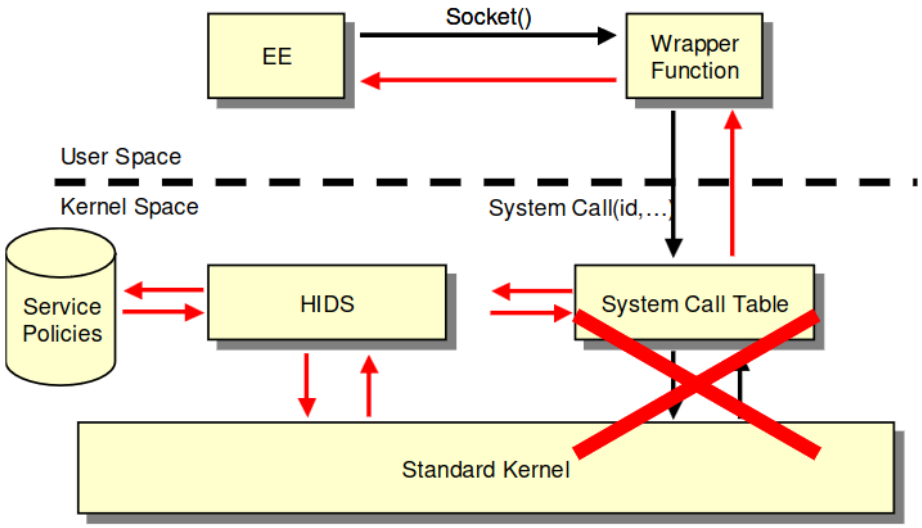


Fig. 3. Interception of a system call

The proxy logs the connection data of the honeypot service. A difficulty thereby is to match attack session and PID of the corresponding process on the honeypot host, as one host keeps the logs while the other one detects the attacks. If a new client connection is initiated by an attacker, the proxy sends the new session ID via the control channel to the monitor on the honeypot monitor. This one acknowledges the successful connection of the proxy to the honeypot service by sending a message to the proxy server, which contains the PID of the corresponding child process and the ports of the incoming connection. The ports are used to track which connection and session ID belong together.

Furthermore, the proxy server maintains a list of currently established connections to the honeypot service. On a fork of the honeypot service a new PID message is automatically sent by the honeypot service monitor to the proxy server. In case that a process tries to execute a series of an unauthorized system-calls (attack signature), the honeypot monitor triggers an alert and sends a corresponding message to the proxy server. The alert-message contains the PID of the honeypot service process that violated the security policy and the kind of violation. By the means of the alert message the proxy server tags the corresponding session and adds the alert information to the

proper log file. In addition, the proxy server is able to stop the ongoing attack session — if specified so in its local security policy.

Process ID Tracking Generally, processes can be identified by their unique process identification number, the PID. As already mentioned, many networking services — that can be used as the honeypot service — create a new process environment for each connection that they accept. Accordingly, a mechanism is required to keep track of the processes that must be observed by the HIDS. Figure 4 depicts our principle of PID tracking. Initially, we open a shell which we subsequently add — with the help of the HIDS manager — to the list of processes that are monitored by the HIDS. Next, the chosen honeypot service is started from inside the shell, which automatically assigns it to the group of services that must be observed by the HIDS.

Normally, a new process is created by a networking service through the execution of the fork() system-call (other possibility clone()). In this case the operating system creates a new process environment and assigns a new process ID to it. Now, in case that an attacker connects to the honeypot service, then the operating system creates the new process by executing the fork() command and returns two values. One value, which is the PID of the newly created process, is returned to the parent process, whereas the newly created child process receives a zero as return value. Whenever a process that is member of the list of services that must be observed by the HIDS creates a child process, then the newly created child process is automatically assigned to be a member of the list.

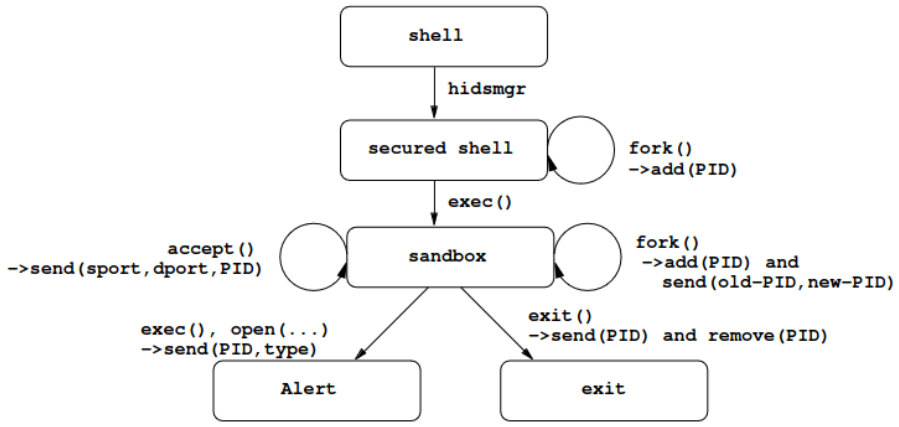


Fig. 4. Process ID (PID) tracking

6.3Replay Mechanism

The session-specific log-files that are created by the proxy server can be used by the replay tool, that is part of the proxy server, to repeat a selected (attack) session. In case that the honeypot service is stateless and deterministic, the attacker can be replaced by the replay tool and each suspicious log-file can be replayed in sequence. This allows to analyze attacks in detail as well as to test if other systems are equally vulnerable to an attack.

Resourced Used

Honeypot Design
https://www.networkworld.com/article/3234692/increase-your-network-security-deploy-a-honeypot.html
(https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1011&context=msia_etds),
https://homes.cs.washington.edu/~arvind/papers/hshoneypots.pdf
honeypot technology and tools

https://www.blumira.com/video/what-is-a-honeypot/

https://www.linkedin.com/pulse/honeypots-types-technologies-detection-techniques-tools-ahmed-eissa/

https://www.csoonline.com/article/2614083/security-no-honeypot-don-t-bother-calling-yourself-a-security-pro.html

https://www.csoonline.com/search?q=%22common%20mistake%20to%20prevent%20while%20implementing%20honeypots%22#gsc.tab=0&gsc.q=%22%20honeypots%22&gsc.sort= (all about honeypot)

Common mistake resources

https://www.comparitech.com/net-admin/how-to-establish-a-honeypot-on-your-network/ ,

Deployment Strategy

https://www.researchgate.net/figure/Honeypot-Deployment-Strategies_tbl1_39551 68 ,

http://www.ijcns.com/pdf/ijpcsc2.pdf ,

http://www.ijates.com/images/short_pdf/1408183790_P210-216.pdf

Honeypot Architecture

https://www.researchgate.net/publication/221500107_A_Honeypot_Architecture_for_Detecting_and_Analyzing_Unknown_Network_Attacks

A Practical Guide to Honeypots. (n.d.). Retrieved from cse.wustl.edu: https://www.cse.wustl.edu/~jain/cse571-09/ftp/honey/#sec3.3

Boulaiche, A. (2008). *HONEYD DETECTION VIA ABNORMAL BEHAVIORS.* Retrieved from HONEYD DETECTION VIA ABNORMAL BEHAVIORS: https://www.scitepress.org/Papers/2008/19272/19272.pdf

Checkpot_Github. (n.d.). Retrieved from https://github.com/honeynet/checkpot

Cowerie_wikipedia. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Cowrie_(honeypot)

Cowrie_Github. (n.d.). Retrieved from Github: https://github.com/cowrie/cowrie

EC-Council. (2020). Certified Ethical Hacker (CEH) version 11 - Evasion techniques. In EC-Council, *Certified Ethical Hacker (CEH) version 11* (pp. 1000-1003).

KFSensor. (n.d.). Retrieved from KFSensor: http://www.keyfocus.net/kfsensor/

Wikipedia. (n.d.). *Kippo_Wikipedia.* Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Kippo#cite_note-githubcommit-3