ISBN: 9798375252568

4. Hacker-AI - Basic Features and Consequences

Overview

I will discuss Hacker-AI as a tool and part of military strategy, a cyberwar that I will call Cyberwar 2.0. Hacker-AI features are discussed within cyberwar segments (i.e., distinct phases or stages that achieve specific objectives). I assume four main cyberwar phases: preparation, attack, exploitation, and protection of gained position.

The Hacker-AI features will be all about tools or capabilities: Preparation involves planning and organizing resources for cyberwar's subsequent campaign phases. The attack involves tools to achieve specific objectives, such as capturing "ground" or positions within "occupied devices". Exploitation involves continuing operations to take advantage of the gains from the attack phase. Finally, the protection of gained positions provides tools to secure and consolidate the advantages made during the previous campaign phases; it will prepare with tools for potential (late-stage) counterattacks by defenders, new adversaries, or challengers.

Feature-list

I will discuss a list of specially named features. This list provides us with an overview of what could be expected from Hacker-AI. The chosen names with their included features were chosen to resemble the goals that the feature is expected to accomplish.

<u>Preparation - Information Gathering</u>

1. **Tech Library**: Library of hard-/software details, including small product version differences. Repository of everything worth knowing by AI that develops software for every tech platform
2. **Cyber Reconnaissance**: Getting details (type, owner) about as many devices and their software as possible. Also, understanding concrete usage patterns, device locations, or owner's occupation
3. **Tech-Simulator**: "Sandbox" on which hacker solutions for all devices known in the real world are developed quickly, safely, and reliable

Hacker-AI will try to gain access to as much information as possible about technologies, devices, and software details. The Tech-Library also contains past technologies because software revisions contain valuable lessons for hacking security features. The Tech Simulators will utilize these details to create proactive or on-demand solutions the deployed malware uses.

Attack Tools

4. **Cyber Beachhead Planner**: Tools to get software on attacked device; bootstrap its deployment
5. **Rights- or Permission-Elevation**: Hacking methods so that malware can do whatever it wants
6. **Cyber Cradle Builder**: Tools for malware that needs/finds on device hideouts to stay protected.
7. **Cyber Whisperer**: Having reliable backdoors and stealth piggybacking of data exchange; operating backdoors via existing user communication methods

The attack is segmented into 3 main phases: entry (beachhead), expansion (bootstrapping/rights elevation), and position building (hiding-out). Additionally, methods and tools are introduced to facilitate stealth/covert communication within these 3 phases.

Attack Exploitation

8. **Cyber Masterthief**: Tools to specifically steal data like user credentials and crypto-keys
9. **Cyber Freeloader**: Utilizing other device apps, resources, and features (i.e., living off the land)
10. **Cyber Covert/Shadow Recorder**: Surveillanceware that stores from multiple devices pre-processed (aggregated) all relevant intelligence about a user

Hacker-AI will steal what it needs, including crypto keys and user credentials; it can enable malware to use all device features and resources (storage, camera, mic, etc.). Finally, malware can surveil a person and create pre-processed and (almost) immediately usable/operational intelligence.

Fortification/Protection of Position

11. **Cyber Ghost**: Undetectable software that avoids any (suspicious) trace showing its presence
12. **Cyber Devil**: Irremovable malware fights late-comers for the same exclusive position on device
13. **Covert/Private Backdoor Facilitator**: Restricting access to a backdoor to their original owners/ creators using asymmetric encryption

Once malware generated by Hacker-AI gets access to a device, it will try to stay on it permanently as a Cyber Ghost or Devil or via re-entering through an exclusive backdoor that could quickly reestablish full system access for an attacker.

Miscellaneous Misdirection/Decision Layer

14. **Cyber Patsy Designer**: Software creates "evidence" or diversions to have humans stop investigating suspicious problems arising from Hacker-AI or Cyberwar 2.0
15. **Attack Synchronization/Management**: Managing command and control of Hacker-AI in a Cyberwar 2.0 or for Cybercrime 2.0 product scenarios

Fortification and Misdirection/Decision Layer will be discussed in the next chapter. "Hacker-AI Advanced Features and Considerations".

Creating the above list is likely not comprehensive. If we drill down into details, the mentioned tools may have features that could lead to new attack opportunities or "product features" useable by cybercriminals.

Additionally, Hacker-AI operators will need to know how good each feature is compared to comparable versions or solutions. This implies that these operators will use metrics for measuring performance, optimization, or comprehensiveness.

Also, the data exchange architecture between the attack units could be a combination of centralized and decentralized components with a middleware of temporary/redundant hubs or tool/data repositories for supplying tool features for storing or aggregating data before the attacker's central data repository receives them. However, if readers would assume that operating Hacker-AI that wages a Cyberwar 2.0 or is involved in Cybercrime 2.0 would require expensive server resources, that expectation is wrong. Malware could steal all required resources and create a redundant, distributed server that would do all the tasks the operator would need to have done without a single purchase of server resources.

What to Expect

Hacker-AI will likely be a highly automated set/system of features that helps operators to attack any computer system easily and quickly. It does not need to be fully autonomous or even have a mind of its own or goals. The automation and the user interface could give users the impression that Hacker-AI has/shows in its interface to humans some form of "Enthusiasm". But that motivation is likely a method or consequence to remind humans to follow up on tasks toward their goals.

Hacker-AI results are provided in a form so that the attack software (malware) has or will always have the expected features, as stated by human operators. Hacker-AI will potentially have a simplified language to receive requested features quickly, easily, and reliably.

I am not discussing or expecting (full) autonomy within Hacker-AI or that the Hacker-AI is the sole operator of an attack. Instead, I consider the combined results of the above feature list to be Hacker-AI (capabilities).

As with other modern tools, we must expect that this tool is easily adaptable to different attack scenarios and / or applicable to other easily identifiable groups of people (within different nations). Hacker-AI could be used in Cyberwar 2.0, Cybercrime 2.0, and against engineers and manufacturers trying to develop / deploy countermeasures. All information collected by its generated malware is expected to be easily utilizable by different features, e.g., surveillance, industrial espionage, or by criminals who want to seed distrust in a society that is then being turned into a law-and-order state with less individual freedom.

In the next sections, I will explore some of the details behind these features and components. For some, this might be boring; others can't wait to get more details. But I hope that there are some nuggets in for everyone. I don't want some evil government or organization to take this as a blueprint, but cyberdefenders should know what kind of adversary they could face.

Preparation

Every war, including cyberwar, requires a lot of planning, exercise, and simulation to increase the probability of success. Cyberwar 2.0 is a large data operation broken down into steps that can be tested, measured, and simulated in many details. These simulations require many reliable and detailed data collected ahead of the operation. Some of the information is required for getting the malware developed and tested.

Based on the comprehensiveness of the desired tech library, Hacker-AI will learn and then create cyberweapons on demand. These cyberweapons are designed, optimized, and tested by Tech-Simulators based on the requirements for their operation. The tech simulator will show if it could deliver new or adapted cyberweapons at the speed required in a fast-paced Cyberwar 2.0 situation. Operators must be prepared to adapt, which means the used tools must adapt quickly and reliably; otherwise, intimidations made within the cyberwar could fall flat due to a lack of timely follow-up.

Cyber Reconnaissance determines what technologies are around and must be hacked; reconnaissance gives the full cyberwar simulation a realistic picture of what could happen once cyber operations are initiated. Still, old data changes, and new data must be integrated throughout preparation and after hostilities are initiated. An initial comprehensive reconnaissance sweep is followed by continuous sampling or surveillance on much smaller scales.

(1) Tech Library

It is certainly true for human hackers: more information increases the likelihood of hacking a system and decreases the time it can be done. It is assumed that this is valid for automated Hacking AI as well. Hackers gather as many technical details about all technical aspects of a device they attack before narrowing their focus on the most likely vulnerabilities. If starting from scratch, pulling tech details could take days, weeks, or even months until all information required for an attack are together; but time matters and preparation for all types of contingencies will help tremendously. People want access to information quickly, comprehensively, and reliably, which is why societies prepare and maintain libraries.

To maximize Tech Library's value, every published iteration (version) of every software entering the market should be archived or available via link. The same applies to hardware and technical interfaces.

I am not saying that comprehensive libraries with technical information do not (already) exist. I am also not saying that technical information required for hacks cannot be accessed online (comprehensively). Instead, I am just saying that Hacker-AI must have quick access to as much technical info as possible. For state actors, this tech library seems feasible; for regular companies or independent hacker groups, that task is probably a too heavy lift.

Without making assumptions about how and what information is gathered, we could assume that Hacker-AI can access a comprehensive tech library containing information on all digital hardware and software within seconds.

State actors would not be limited by published information. They could use espionage to get proprietary information from manufacturers or even scrape databases or file repositories of experts who collected different types of technical docs or files for whatever reasons. We can assume that the top intelligence services know where to look.

A sequence of software revisions would be used to train Hacker-AI in detecting already found security vulnerabilities. Therefore, the goal is to have different versions of the same software and info on previous security issues as training data.

Also, every hardware revision or iteration could and should be archived with metadata of its potential usages or vulnerabilities. Having the actual hardware in stock would even be better.

Using old compiler versions to generate code that showed similar vulnerabilities could also be used to train the AI. No piece of code (source or binary), tools, libraries, templates, technical information (specifications or design documents), or even checklists should be considered too much or too irrelevant to be archived and then post-processed. Currently, many archives are information dumps. But with digitalization and AI, their owners or operators can expect more treasures to be lifted from libraries; Hacker-AI's Tech library could be a driving force for getting the facilitating tools open-sourced.

Hackers could design the tech library for hackers and Hacker-AI. As a result, all key pieces together will help us to understand technologies more easily. The quality of the tech library is probably measured in how complete information is stored in the repositories and how (fast) the search would help in understanding technical functionality, i.e., how certain technology works, how it was designed, and its capabilities, limitations, and requirements. As someone who did several deep dives into technologies, finding information and organizing it was the most difficult part. Understanding its layers, principles, and possible motivation of optimizations/complexities was not so much.

Although the tech library does not require AI, it could make a significant difference, if used. The tech library is a key component for Hacker-AI; it will enable the development of new attacks on previously ignored systems quickly. The tech library is also essential for developing reliable defenses quickly. Public access could be a net positive as it could unleash the development of security tools more quickly.

The quality of the tech library could be tested and measured in how often it failed to deliver the information required in training or testing of malware generated by Hacker-AI.

(2) Cyber Reconnaissance

Gathering information about target networks or computer systems from the outside is relatively easily detectable. However, this external method could deliver unreliable information because defenders could spoof attackers with false information. More inconspicuous is to have software on phones that gather information about infrastructure, systems, networks, and people. TikTok came under suspicion that it might spy on users' environments. It would probably be known if TikTok had already done something like that. The problem is trust that it is not doing something sneaky in later updates. Software could have spy features for a limited time before another update removes it. But wait, the update could have included the removal code.

Knowing what devices are out there and where they are located has tremendous value for countries trying to control populations via surveillance. And for hackers/criminals, network and system reconnaissance information could become useful because this information can help attackers plan, simulate, and execute a successful attack in advance. Actually, it is an essential part of cyberattacks to understand target's strengths and weaknesses while identifying vulnerabilities for which exploits are developed before actions are taken. Also, what contingencies must be taken into account?

Cyber Reconnaissance could map networks, identify and catalog all connected devices, systems, and the relationship between these devices. Every wireless device broadcasts information to all devices within a local network. It is only a matter of a few minutes until sent/received messages reveal the device type, operating system versions, and potentially what additional software is installed. With smartphones, users reveal their physical locations and with whom they are regularly in contact. Smartphones could even be used to determine via proximity detection within (non-public) locations who are the close-by coworkers. These data are considered unavoidable footprints; they can be detected and recorded passively. Listening to network traffic could be too much (data), but if done smartly and optimized, attackers will get what they came for without being caught by other systems. Also, if done carefully, active scanning of network ports could help to detect devices with open ports or vulnerabilities that could be exploited for attacks.

Location information, locally stored phone numbers, and personal information could be extracted by many apps, particularly if a business reason is used to explain why certain permissions are required.

However, turning a smartphone app inconspicuously into a local network scanner is more difficult. But this is possible if the local app can elevate its permissions. Hacker-AI can generate features for this reconnaissance tool that can grant permissions covertly after exploiting local vulnerabilities.

Additionally, social networks can be created with data from proximity detection and enhanced by gathering publicly available information about the target, such as social media profiles, press releases, or public documents.

The above example with TikTok shows that Cyber Reconnaissance does not require persistent malware or spyware. Updates of ordinary apps are enough to generate a treasure trove of data that could be reported to regular servers as encrypted piggybacked data. If Hacker-AI is used, app manufacturers are likely not involved (as assumed with TikTok). It might be a comforting thought that businesses must be forced to cooperate, but it is plausible that software manufacturers are unaware.

For defenders, there is probably little that can be done about Cyber Reconnaissance. It is conceivable that national intelligence agencies, even the National Security Agency (NSA), could be blindsided that reconnaissance happens with devices of their or another country’s citizens. However, everyone in security knows that smartphones in security zones are usually strictly prohibited, independently of phone brand or installed OS.

Although cybercriminals could do Cyber Reconnaissance, it is assumed that large-scale cyber operations usually indicate preparation for cyberwar operations. Larger cyber activities carry some risks and challenges to be flagged as national security risks. Additionally, western governments would probably have an eye on the legality of these operations and ensure that their actions are conducted legally and ethically within the bounds of the law. But if done on foreign territories or by foreign governments, legality is less important than being stealthy. Preventing detection avoids alerting targets. Every detection could compromise operation’s success.

Humans would use a single method, but Hacker-AI would likely ignore human's propensity for one problem, one solution. Hacker-AI may use 100s or 1000s of different reconnaissance tools. If caught because of insufficient reconnaissance data and detection prevention capabilities, the blame for a single found-out version could be redirected to others. Also, Hacker-AI could learn from failures when it uses many different approaches.

Cyber Reconnaissance is part of preparing for another (larger) goal. It will give attackers sufficient information for planning without leaving traces of what will happen in the next step. Doing reconnaissance should be discreet and unassuming. If it is detected, then it could be prevented. Hacker-AI could be used to create reconnaissance tools. Being undetectable is its key performance parameter.

(3) Tech Simulator

Configuring systems is often a labor- and time-intensive task. Hackers and IT departments dealing with different configurations have (certainly) created extremely efficient and fast methods to set up any software or hardware environment they need quickly; this is so normal that this should normally not be mentioned. Selecting systems and testing them for vulnerability could already be a simple menu option, done in seconds.

The tech simulator could be described as a "fishball" in which the inner working of the studied software can be examined and monitored effortlessly from the outside. I am not saying that this tech simulator does not exist yet, but if it exists, it could include some features that are not available in regular virtual machines. Applications could be reset, i.e., the entire environment would restart at a previous time (for which a snapshot was made) - all done to save time. Then hackers or Hacker-AI could step through until the software would reveal its secrets at software's lowest level.

This tech simulator is a special virtual machine that is being configured quickly; it does not require AI. But it would likely be done by hackers for hackers and Hacker-AI. So, we can assume that it will contain a lot of automation so that all humans in this hacking process become eventually expendable.

Tools for Starting Hostilities

Preparation extends to the planning of tools for actual hostilities. These tools are about basic capabilities like entering devices (Cyber Beachheads) and establishing a hidden base of operation (Cyber Cradle Builder). These malware features are based on the reliable elevation of malwares' rights and permissions. These features are adapted to concrete attack (battle) situations and requested by the malware. Additionally, malware must communicate without creating suspicion (Cyber Whispering). All the above basic features are flexible and adaptable and respond to changing circumstances. Each activity is designed to optimize in achieving its objectives. Each malware component will need to maintain its operation, i.e., it pulls resources that it requires from the outside. It is doing this to sustain its operations, secrecy of used tools, and maintain its effectiveness.

None of these initial tools/features are about a specific mission. This way, malware is a platform for capabilities, usable in different versions on different hardware/software platforms. How these capabilities are used depends on the operational goals.

It is certainly possible that the real malware generated by Hacker-AI would ignore this approach and follow a different methodology. It is also possible that the below features are included in one malware app or that it has designated features to code on the targeted system when it requires features. In the following, I am not predicting any tool but discuss only basic capabilities that we can assume from malware that is hacking systems as part of a cyberwar.

(4) Cyber Beachhead

A Cyber Beachhead is the first step of malware on a device it wants to occupy. This terminology uses the association of a beachhead as a foothold established in enemy territory. It starts as a small area that serves as a base of operations from which further advances are done to establish a more permanent presence. In a broader sense, beachheads can also refer to situations in which the presence in new or unfamiliar environments must be formed by attackers that intend to expand or consolidate their position over time.

Establishing this foothold on a target system can be achieved by exploiting a vulnerability, gaining access to a user account, or via the inconspicuous update of software that helps to install the malware. Software that is doing these initial activities is also called dropper.

There are click-free and click-based methods of installing malware. The click-based method requires users to cooperate (unintentionally). Backdoors and other vulnerabilities are often used in click-free methods to infect devices.

Once a beachhead has been established, the hacker can use it as a base of operations to further exploit the target system or network and find a better, safer place to store its software and get started reliably. This also involves installing additional malware, escalating privileges, or extracting sensitive data as part of a mission. A Cyber Beachhead should be assumed to be followed by more sophisticated attacks designed to compromise the target system further.

The idea with a beachhead is that one starts with limited resources because it is difficult to bring in additional resources or support for maintaining or defending the position. Every step must be planned out so that the vulnerable beachhead position is not wasted or limited in its expansion by countermeasures.

Planning Cyber Beachheads and contingencies for the follow-up steps is part of the hacker's preparations. Once the attacker knows details on target's system, he could use Hacker-AI to develop the beachhead technology predictably. Depending on how this beachhead is being created, the attacker may have used a valuable vulnerability to which he does not want to be exposed. Methods and sources must be protected from being revealed, even if it is easy or cheap to find new ones. After arriving at a beachhead, the facilitating tool should be deleted, and the malware should be moved to what is later called a Cyber Cradle.

Creating beachheads is part of Hacker-AI's preparation for different operating systems and applications. Prevention of detection is still a major goal, but because of insufficient Cyber Reconnaissance, the beachhead could be detected. Hacker-AI's malware may not be able to detect immediately that the system generates hidden data traces of its activities or that it has triggered tripwires or entered a honey-pot. Malware would probably not enter a Cyber Beachhead with advanced AI code because the attacker must be concerned that he got into a honey-pot and that his capabilities could fall into enemys' hands.

Hacker-AI would create malware code that assaults the device with info gathered from a distance. The beachhead malware must determine quickly if it was detected and start countermeasures if the situation is known or report the new circumstances. The response time of Hacker-AI for learning, improving, and adapting under conditions when Hacker-AI's malware is about to fail could be a critical parameter to optimize.

(5) Rights / Permissions Elevation

An important aspect of leaving the beachhead and establishing itself in some other location on the device is that malware must gain additional rights and permissions on devices it enters as soon as possible. It could be argued that this step is part of the beachhead planning / tool, which is true, but the same would apply to other follow-up steps we can expect the malware to do.

In general, I expect malware entering a device is trying to gain administrator-level access to a computer or network system. Malware with sys-admin rights is sometimes called a rootkit; it is designed to operate at a low level within the system, making it difficult to detect and remove. Additionally, the sys-admin rights allow attackers to control the system by installing additional malware or performing actions without detection.

About 15 years ago, there was a pretty easy way of creating rootkits using "system hooks" with a generic mechanism that allows the OS to intercept and modify the behavior of events and actions. Preventing system hooks via a low-level solution ("Hooksafe") stopped almost overnight a certain type of rootkits from emerging on devices. But this solution did not eliminate unauthorized system access in general.

In this book, I don't want to call every malware with sys-admin rights a rootkit. One key difference between malware and rootkits is their access/permission level. Malware operates at a higher level, with access to the operating system and other software; rootkits are designed to operate at a lower level, with access to system's hardware and firmware. Therefore, I assume that rootkits have the tools to continue operating even if the OS is reinstalled or the hard drive is being formatted. This level of camouflage and avoidance will be described as a Cyber Devil.

With rights or permissions elevation, users or processes have more privileges to access resources than they normally have. This elevation is often necessary to perform certain tasks for which these rights are required. For example, users with "view" permissions could see file content on a hard drive without changing file access settings. A user with "admin" rights has full control over the filesystem, including granting other users view permission. Basic filesystem operations like opening or reading files cannot have, by default, these admin access permissions. They must be granted by an app or component using the filesystem. An app doesn't have these rights by default; instead, the system creates a layer with exceptions for roles or groups with rights and permissions to bypass restrictions without giving these apps full permission. Setting and using these exceptions requires rights - protecting these methods is a complex business from which vulnerabilities could emerge. It is a complex interplay between layers in the OS. We should not be surprised that developers always don't do it by the books; they get caught within this web of rules, permissions, exceptions, and go for a shortcut in which they don't need to understand all details.

Under normal circumstances, a few key features (sometimes only a single one) are required to grant rights or permissions elevation. The process or person must have the authority to grant an elevation; this is typically a process with administrator rights. There should be a clear justification, like the need to perform a specific task or access a specific resource. Additionally, appropriate security control methods must be in place to prevent unauthorized elevation; these methods also encompass methods to monitor any potential abuse of elevated privileges. Once elevation happens, it is important to be documented to track and audit who has elevated privileges and potentially why. As shown, the rights or permissions elevation process is complex and potentially risky, but it is required - it is bedrock OS technology. Unfortunately, we should not wonder why we often hear about (severe) security problems.

Hacking rights or permissions elevation can be done in the protected environment of sandboxes, i.e., virtual machines and the above tech simulator. Human hackers would do that potentially on live/production systems if they don't know them in detail, but doing so is risky. Repeated testing or probing of system's security is a relatively easily detectable anomaly that AI within defenders' safety tools should detect.

For elevating rights / permissions, hackers have several approaches to exploit vulnerabilities in gaining unauthorized access or escalating privileges. Hackers could even try to steal reused / default or weak passwords that they could guess, or they use social engineering to phish or trick users into divulging their login credentials. Another method has users install trojans (malware) in which he grants sys-admin privileges because the installer asks for these permissions. Granting these rights is a mistake. But the bigger mistake is to have users involved in these low-level security issues.

Unfortunately, protecting protection methods make rights and permissions management a large component within the OS. But then: is the OS also protecting the protecting protection methods? Code around protection could contain many vulnerabilities that have been overseen, and every new update could create new ones. Different OS, code, contexts through user settings, and unconcerned implementation of features in additional software packages will contribute to a large pool of possible vulnerabilities to get rights or permissions of malware elevated without authority or authorization. Humans have not found more because finding them is labor- and time-intensive. Some criminals or intelligence services are paying for these 0-Day vulnerabilities high 6-figure dollar amounts.

Due to its inherent complexity, OS and devices always have (inadvertently) vulnerabilities. It is a reasonable hypothesis that having vulnerabilities is an emerging property of complexity; if that hypothesis can be proven or disproven is another matter. As long as security methods commingle with regular / user code within complex systems, we should not expect vulnerabilities to remain undetected and unused.

If AI is used defensively to find vulnerabilities, then they must also fix all found problems in the deployed software - on all systems, i.e., release new revisions of the OS / app, before these fixes are effective. Hacker-AI could check out the new version for new vulnerabilities at that time.

A performance criterion for Hacker-AI could be time, i.e., how fast a Hacker-AI version or iteration could find new vulnerabilities. A secondary criterion could be how many compared to other Hacker-AI instances it could detect.

(6) Cyber Cradle Builder

After entering a device via the Cyber Beachhead, the initially executed code has confirmed that it is not trapped in a honey-pot or a virtual machine from which it cannot access the system resources on device’s hardware without going through a non-circumventable software. By elevating its rights and permissions, the beachhead code would find and move to a hideout called the Cyber Cradle. I have chosen the word cradle to use the association with a device or structure designed to support or hold something in a safe or stable position, such as a phone, a telescope, or a machine.

The Cyber Cradle holds software applications in a secure location and makes them available, i.e., to be started when necessary or triggered via hidden criteria.

Hiding malware can be done in many different ways. Malware can be disguised as a legitimate program (like a trojan) installed on a computer through normal means. Or malware could hide in legitimate software, even within a software update, or in benign files via macros stored within user documents. Busy readers should jump to the end of the next paragraph because I will continue with more examples.

Existing macros can be modified without changing their behavior; users would not detect changes from what they expected. These modifications could be done later and hidden. In general, malware can also be injected into legitimate programs or processes without altering their behavior. E.g., data structures or programming or scripting language modifications could be used. Software modifications do not need to contain the entire malware code. They could initiate malware so that it becomes active or reactivated after self-termination - these injections could also be done on code stored within the memory/RAM. Other methods are rootkits, i.e., software that has modified the OS or system components, or bootkits, i.e., software that infects the boot sector of a computer's hard drive allowing software to be loaded before the OS or security software has started; this software is also called a bootloader. It is also possible to have malware running in a virtual machine using software from the main OS but hiding its activities within RAM, i.e., as operations within another app. Another way is to hide in custom encryption. Unique or proprietary encryption is used to protect malware from detection and from being analyzed. Variations of this scheme are custom communication protocols, custom file formats, and even custom command and control servers to receive code, commands, or transmit data. Other methods are based on a custom application programming interface (API) and other programs or systems used to start hidden features of the malware. Malware could also include itself in the firmware of hardware components or within computing units used/provided by the GPU (i.e., graphic card). There is an almost unlimited pool of opportunities how malware could hide.

Additional measures to hide software modification are based on removing suspicious traces. If OS updates files, it changes the access date and file size; these data reveal that a file was modified. However, this information could easily be changed within the corresponding filesystem data structures with the appropriate permissions/rights. Imagine, malware would leave the access time and file size the same as before; there is no hint that the file was modified. Also, as soon as a trigger warns malware that a deep analysis has started, malware could go into a special hiding mode. At the same time, storage hideouts are reset to their original stages and reestablished after the scanning software is finished. Alternatively, Hacker-AI could temporarily move its software into a beta or gamma site, i.e., in prepared places where the scanning or cleaning already happened.

Different inconspicuous locations within the file system could contain marks pointing to dead drops, i.e., places to store data secretly. Other malware within the system can use these dead drops so that they don’t know each other and their locations, but they can still exchange data safely.

A performance criterion for Hacker-AI could be time, i.e., how fast a Hacker-AI version or iteration could find new cradle locations. A secondary criterion could be how many compared to other Hacker-AI instances it could provide.

(7) Cyber Whispering

A Firewall is security software or hardware separating a local computer or a network of local computer systems or devices from the Internet while forming a more protected internal network, the intranet. Unfortunately, systems within the intranet are not protected, and firewalls are useless against malware; they can get in touch with outside systems. The best that could be said about firewalls is that they try to be security conscience routers.

Malware inside the intranet can act as spyware, sending encrypted data to an external server; a firewall is likely to do nothing. Additionally, malware could likely receive messages from an outside source as well. Firewalls should block or restrict access to potentially malicious traffic, but adversaries have learned to bypass firewalls - via piggybacking. Malware is using browser traffic to camouflage its data exchange.

Firewalls are designed to monitor and control incoming and outgoing network traffic based on predetermined security rules. They are configured to allow or block specific types of traffic from known malicious sources or traffic that uses certain ports or protocols; firewalls are known systems for which adversaries are trained and prepared. For advanced attackers, existing network protection is insufficient even if firewalls are seen as part of a multi-layered approach to security. In short: firewalls cannot prevent malware from entering networks or devices or reduce the risk of infections. If firewall manufacturers promise protection and security, then that is currently a marketing claim, and the best case: you may get an apology after the next failure. Firewalls are not proactive security tools.

Defenders must be prepared that attackers are piggybacking data in existing connections or channels, using existing connections to transmit encrypted maliciously and unauthorized without being detected.

Piggybacking data is likely done by malware and spyware operating covertly. Diving a little deeper: Piggybacking occurs when attackers exploit legitimate connections, such as HTTP or HTTPS, to transmit data maliciously. HTTP/(S) are Internet communication protocols that transfer data between websites/ servers and user devices; the S is for “secure”, indicating encryption to make it harder for attackers to intercept and read data. Even spyware could use HTTPS to transmit data through a firewall via legitimate channels allowed by firewall's security rules.

Firewalls are computer systems with a complex architecture - although they seem simple. They have vulnerabilities, and spyware could exploit them and bypass their security rules. Firewalls are not decrypting passed-through content; therefore, spyware could evade detection and avoid being blocked by a firewall using encryption and multiple channels or messages to known servers that are covertly already under assailant's control. Finally, it is often overlooked that firewalls are normal computer systems that need updates. If administrators can do updates, attackers and their malware can also.

Applying piggybacking data on existing communication methods will overwhelm network security. But this is nothing in comparison to what Cyber Whispering could do. Normally, whispering refers to soft-voiced, discreet, or confidential communication, conveying a sense of secrecy, which creates exclusivity and privacy with another peer without being overheard by others. Whisperers are not drawing attention or raising their low profile; they are heard by their audience but not by others. Unexpected message timing from applications or encrypted communication channels raises suspicion, which can be detected as an anomaly. Cyber Whispering is Hacker-AI adapting to users' normal communication tools / preferences while stealthily using them.

In Cyber Whispering, spyware / malware impersonates a user and sends private communication to some other party, and removes all local traces that this exchange ever happened. Text messengers use local databases to store data from data exchange or communication events. Malware will know when and how to use a system and its regular feature without detection; It knows how to remove traces of its actions; it will also know how to use these tools "headless", i.e., without visual user interfaces to get messages sent or received. Cyber Whispering could happen simultaneously when the user uses a hacked device or communication app.

The advantage of impersonating humans is that it can use the security and confidentiality of conversation tools (encryption) against unauthorized eavesdropping from other humans; it would not draw attention to its conversation as long as it would not leave data traces. It would use additional encryption to reveal nothing, even if the call / message is intercepted. I am not aware that malware uses communication tools designed for humans on this level yet. But I wouldn't be surprised.

Don't get caught seems to be the only performance criterion for Cyber Whispering. Attributing communication to malware will be difficult; attackers could have protocols for dealing with different detection situations and making it look like a software bug.

Exploitation

Once malware is deployed on devices, malware can be extended by additional features for stealing valuable data like user (access) credentials, crypto-keys, or other data appreciated by the attacker. Having user credentials available makes it much easier for attackers to impersonate users and gain resource access from servers for which user credentials are required. Attackers could hack servers directly, which would increase the probability of being detected.

User devices provide many useful capabilities for attackers. Attackers can do whatever users can with their devices (Cyber Freeloaders / Shoplifters). Attackers do not need to come with an army of drones; they could coerce people to give them access to their drones, 3D printers, or even home camera used as hidden CCTV. Additionally, like Pegasus spyware, smartphones (as our constant companions) could be used as covert audio / video surveillance recorders; the data could be pre-processed (removing irrelevant noise) and aggregated to operational intelligence before being uploaded to the surveillance system.

(8) Cyber Masterthief and Stolen Data

Once malware with elevated rights is on a device, there are no secrets that could be hidden from that software. User data, identified as useful, is copied and sent to the outside with no additional data traces that this has even happened.

The most important data for attackers are encryption keys. However, for an attacker being in the background, he does not need crypto keys; malware could analyze the content received or sent on the device or crypto hardware directly and report the results to the outside. So, if manufacturers use crypto-card hardware to protect their transmitted content, how do they detect that this hardware is not being misused in the background? Answer: they can't.

Data to be stolen from computers depends solely on the motivations and goals of the attackers. The data could be personal, financial, confidential, or from governments. Personal information could include names, addresses, phone numbers, social security numbers, and bio-identification data like fingerprints, voice, or facial data. Stolen financial information like credit card numbers, their utilization / limits, bank account numbers with transaction amounts, and other financial data could be used for fraud and to gauge the vulnerability toward bribing or blackmailing.

Companies have confidential business information like trade secrets, intellectual property, financial records, or other types of sensitive or secret data from which often the destiny of a business depends. Governments have classified documents, military secrets, and other types of sensitive data related to their national security. Their secret data are often related to sources or methods and should never be released.

Some company or governmental data are millions or even worth billions. Adversaries are investing large amounts in getting them. Intellectual property is not about the secrecy of publicly available information like patents, trademarks, and copyrighted material but information on how they could be turned into economic gains. Secret know-how is often protected with NDAs (non-disclosure agreements) or secret licensing agreements that include written documents or electronic files, allowing licensees to utilize secret technology effortlessly.

The credentials mentioned above are login information used to access systems or services; hackers and cybercriminals are targeting them to impersonate users. Normally, credentials include a combination of username, password, security tokens, or authentication data, often provided via a secondary device. This two (2)-factor or multi-factor authentication (2FA/MFA) assumes that an attacker doesn't know how this additional data record (usually a number or a string) is delivered. 2FA/MFA assumes that the second device is not compromised or is used covertly within an attack.

Once attack software has certain privileges on user devices, malware can use additional Hacker-AI-generated instructions to extract certain data from installed software. The most generic method of attacking data or features within existing software is to modify software with reverse code engineering. So Hacker-AI would have created instructions to modify the installed software by injecting additional binary code with the software. The binary-modified software would work as if nothing has happened, but the additional instructions would extract the intended data predictably and covertly. There is currently nothing a user or defender could do to prevent or even detect that this happened.

The malware installed on the compromised devices doesn't have to be smart. It follows instructions to modify the binary code by merging and injecting a few 100's or 1,000's byte values at the designated spots within the hacked software; then, the malware waits until the hack delivers the result. This type of code insertion is generic, i.e., independent of device type or OS. This attack can be applied on any file stored in the filesystem, or it could be applied when the file is being loaded/read or in the memory/RAM waiting to be executed on the CPU. This code injection is very simple; it can be applied without any trace. Can this injection be done for every binary/compiled software? (Guess what.)

With few instructions, the local malware is turned into a cyber thief, stealing whatever was defined by the (external) attack management. The hack requires prior access to the binary app and its OS context. It's conceivable that developing these extract instructions could be done within seconds after the local malware uploaded previously unknown software (version) to a server where it was being put in the tech simulator with the OS and other relevant software as reported by the malware.

Hacker-AI does not need to use phishing tactics, i.e., sending fake emails, creating fake websites, tricking people into divulging their login credentials or other sensitive information, or physically stealing devices like laptops or mobile phones to access stored data. But it is good to point to these methods and blame users indirectly so that the real malware method remains undetected and unchallenged.

We should assume that Hacker-AI could turn its malware into a Cyber Masterthief whose level of expertise in stealing digital data or assets doesn't fail in any task. At the same time, it evades detection by removing data traces. This described feature is named Cyber Masterthief because the target range of data and digital assets it could steal is wide, and the applicability on devices, OS, and apps is almost universal.

Not getting caught and always delivering results are the key parameters for the Hacker-AI generating malware or instructions designed to steal targeted data with certainty. If the generic attack software installed on the attacked device remains undetected and the data received and extracted within the attack remains undetected, digital theft of any information will likely remain undetected.

(9) Cyber Freeloader / Shoplifter

This feature is called Cyber Freeloader or Shoplifter because attacking software takes advantage of the generosity or hospitality of devices and OS for utilizing any device feature. Cyber Freeloaders rely on resources, i.e., contributions like computing time, memory, or network usage of others, without offering any compensation in return. One may say freeloaders take advantage of others intentionally but without intending any harm - however, that is likely not the case with malware in general.

The behavior of malware is probably better described as shoplifting; it is using someone's device resources without permission and regard for consequences. Even without direct harm, using resources without permission is an unacceptable offense and violation. It should be prevented by the OS or security software; however, preventing this is impossible under the current software architecture. In Cyber Shoplifting, attackers steal / use services without permission and utilize resources or may coerce people to provide access to, e.g., a 3D printer, drone, or video camera. Lack of permission and coercion in accessing resources are both methods that are unethical but technically easy to accomplish.

In a Cyberwar 2.0 scenario, I assume that the attacker avoids destruction but uses IT devices to intimidate and exploit weaknesses. Attackers use resources that are being made available to them by people via coercion or without permission (avoiding evidence that the attacker was involved).

The main difference to the Cyber Masterthief scenario from the previous section is that Cyber Freeloaders / Shoplifters are not primarily about stealing data but actively utilizing devices and software features for assailant's agenda. The underlying technology to facilitate this feature is the same.

Concrete scenarios of what attackers may have in mind are related to the use of cameras, microphones, 3D printers, or regular printers and drones by the assailant. The attacker knows about these resources and their availability. Drones could carry additional tools like small containers / reservoirs with or for liquids, wireless cameras, or laser pointers. Adapters for these tools could be 3D printed under secrecy, coerced by intimidation. Some people may even be bribed to collaborate. Accepting bribes could show within prior, less critical tasks that these people can be recruited for other tasks.

3D printing a large set of auxiliary drone tools could be considered an important step in preparation for Cyberwar 2.0, particularly with a recruited drone fleet that can threaten physical damage or violence against targeted people or locations. This deployment creates fewer data traces than drones or parts shipped to the targeted locations.

Drones are used for covert intimidation at scale. A call on someone's smartphone in the evening, while a drone lurks outside someone's window, carrying a laser pointer and marking a spot visibly but only seen by the called person. The targeted persons probably understand the seriousness of these threats immediately, while others in the target's surroundings might remain oblivious to these events. With many drones in private hands, targeted intimidation could happen concurrently to many people in an automatically planned-out, optimized manner.

However, drones can be used by attackers to harm people physically. Drones' rotors can be used to attack people by cutting their throats or carotid artery. Additionally, drones could carry a spray bottle with paint, flammable liquids, improvised explosives, or other tools that could damage someone's property, health, or life. Controlling these attacks can be shifted between local IT devices, allowing longer flight distances. It does not require a human operator to monitor these operations in detail; a simple command could suffice to get it done (autonomously). It wouldn't matter what additional resources the drone requires; full attack instructions would be executed systematically to get the operation done automatically.

The performance of this feature is less about not being caught because autonomous drones will show their presence to the directly intimidated person. It is about avoiding physical / digital evidence or witnesses that this event has happened.

(10) Cyber Covert / Shadow Recorder - Surveillance Management

Smartphones have microphones and multiple cameras. They are also used as information-platform on which many users manage large parts of their daily digital life. Smartphones contain phone numbers, eMail addresses, social media contacts, and chat histories stored within local databases.

Giving an attacker control over a smartphone could allow him to put someone under total surveillance. This happened already with Pegasus, spyware developed by the NSO Group, used as early as 2016, probably until now. Pegasus was not used for mass surveillance because of its commercial business model.

The problem with smartphones is that they are much more than simply mobile phones. With the control of smartphones, people can be monitored, i.e., observed in their activities via audio and video; their movements tracked, and their relationship with other people detected. When smartphones appear to be switched off, there is no way that someone without advanced equipment can determine if the phone is really off, i.e., not operating. However, someone may still listen in via a hot mic or camera. Edward Snowden warned us about this as early as 2013. So we have been warned, but nothing has been done.

Many offices and security zones prohibit smartphones from being taken in. Many security experts know about it but decided that ignoring this threat is appropriate. They were hoping that governments would use these capabilities only very targeted. Another hope is governments won't use it for mass surveillance because it's too labor-intensive. With AI, this last argument is probably outdated.

Peoples living and working environments reveal strengths, weaknesses, fears, and concerns. If smartphones take pictures of apartments or someone's bookshelf or coffee table with their magazines or newspapers, additional image processing software could create a floorplan of apartments, including vulnerabilities like thin window glasses or inflammable furniture or curtains. Image analysis software could extract names from books or magazines to determine the level of education or propensity for certain ideas. Also, knowing with whom and where a person gets in touch and if this is regular or sporadic can help to map out organizational hierarchies and social networks beyond social media.

The main problem with total surveillance is the huge amount of data generated. Its volume makes it almost entirely useless. But computational resources in smartphones and local computers (e.g., a local storage/ server hub within user's vicinity) allow post-processing of these data. The audio does not need to be recorded. Instead, audio could be transcribed (i.e., turned into text); these audio-to-text features are already available on most smartphones. Software on a hub could consolidate data via audio-, image-, and video-to-text. Automated tagging and labeling could decrease its size further and increase the operational value of these data significantly. The surveillance software could follow a checklist of information it wants to get with higher priority and repetition; unnecessarily wasteful storage of irrelevant information could be avoided. In this setting, surveillance can be done at scale, i.e., on millions of people, for almost zero operational costs.

Without data reduction, extensive surveillanceware is impractical - the necessary server capacity is not there. It can be used on a few people of significant value only. Human operatives must listen to audio and video to extract essentials. This huge effort is good news for countries with the rule of law because establishing total surveillance should not be cheap or easy. If surveillance is done (with court orders), then only in rare exceptions when severe crimes are about to be committed or investigated.

Unfortunately, commercial products are making the post-processing of surveillance data easier. Predicting progress on the ease or quality of future image or video data extraction is hard. Improving feature performances on image/video tagging is a cutting-edge research topic. It is conceivable that nation-states have early access to the best algorithms.

Could this feature already be part of Hacker-AI? Sometimes, it is even better to have people speculate on what reliable or accurate information a shadowy surveillance system could have about them.

Normally, surveillance is done covertly without the knowledge of the surveilled person. But sometimes, e.g., when the person should be intimidated, he should know that he is under surveillance and that steps to disable or worsen the quality of the surveillance are detected. Evading total surveillance would be suspicious or could have immediate consequences.

A key parameter for the Hacker-AI is how inconspicuous it post-process the collected information and how good the extracted data are. That this is doable seems to me out of the question. When total surveillance could be made fully automated seems only a matter of time. Some governments will find ways to justify it with public safety, but still, it is wrong, even if applied to a few criminals. As a teaser for later chapters, smartphones could and should be made unusable for surveillanceware. If this solution can be applied to the current smartphone/hardware generation is difficult to tell.