Proposal for Hacking game play in Star Citizen

Jornada Del Muerto

Objectives

- Outline game play ideas for the pending hacking mechanics
- Devise a system that is accessible to non-technical players as well as experienced
- Doesn't require solving meaningless puzzles that have little to do with the supposed task at hand, keeping players engaged
- Rewards skilled players while still providing a challenge to both skilled and unskilled
- Expand the level of player agency and add dynamic game play
- Fits into existing and proposed game play elements

Outline

Overview

Cracking into real systems (commonly called hacking) is both difficult and tedious. As a game play mechanic, it would be very easy for the resultant game play to be either very boring and difficult, or incredibly unrealistic while stretching the player's suspension of disbelief to the very limits. In this proposal, it is suggested that the hacking game play primarily be a form of data and info running, with some limited interactive elements to keep the player engaged.

There would be two types of attacks that can occur: physical access, and remote access. The physical attacks are those that would take place with the player physically on or near the target, or victim, ship/component. Physical attacks would take place be becoming physically connected to a ship's SAN bus (SC ship's equivalent of CAN bus) from either inside the ship, or by cutting through the outer hull to access one of several access points to the SAN bus that can be tapped into.

The remote access would be achieved through vulnerabilities in any radio communication. As in real air/marine navigation, there are some signals that are necessarily broadcast in the clear, such as

navigational signals, tower/traffic control communications, and collision avoidance systems such as ADS-B or AIS. However, other radio transmissions would reasonably be encrypted, such as player-group/party chat sessions, drone command & control, scanning signals, and the like. Both might be exploitable, although the systems connected to the later would make them a much more enticing target.

One could not reasonably expect manufacturers to leave their systems free of basic encryption, and despite what is commonly shown in the movies, simply 'breaking' the encryption is simply not logical. However, expecting current security trends to continue, it's not unlikely that manufacturers would fail to implement best practices. Just like cell-phone and other device bootloaders and ROMs are routinely cracked today, providing access to the manufacturer's encryption keys, each manufacturer would use a single encryption key for each of their ships that would be discoverable with access to that ship's firmware. These keys, baked into the ship's firmware at build time, are specific to a single model of a ship. In order to differentiate between signals originating from two nearby ships of the same model, each individual ship makes use of frequency-hopping. Where a transmission sent in the clear would be sent on a single radio frequency, a frequency-hopped signal would rapidly switch between multiple frequencies. As long as both the sender and receiver know the pattern to be used, they can stay in sync and communicate. An eavesdropper trying to listen in would be unable to decipher the messages. However, with the right tools (for which I propose the name: **EtherShark**) and a little cleverness, it is possible to untangle this pattern to achieve signal log on the target transmission. The schack program demonstrates this principle. Once locked onto the signal, and for as long as the target is within signal range of the attacker, exploits can be launched, data intercepted, or the signal jammed which would prevent the target from using anything on that channel.

Whether attached remotely or physically, exploits would be then need to be launched against specific systems on the ship. For remote attacks, this would likely start out by attacking the radio itself in order to gain access to the other systems it would be attached to. These radios might be the voice-coms, radio navigation, ASTC (Air/Space traffic control) or an individual player's MobiGlass. If the radio is attached to the ship SAN bus, exploits can be chained to remotely attack other ship systems. Physical access would still need to exploit the target device as one would hope that 900 years from now, there would at least be user/device authentication for connected systems. Data traversing the SAN bus would likely not be encrypted.

When an exploit is used, it must be used in conjunction with a particular payload. The payload is what does the actual work of the attack, whether that is stealing data, listening in on voice/chats, or taking remote control of a ship's thrusters, engineering stations, or turrets.

Details on mining/trading these components, as well as other aspects of this system are detailed below.

Tradable/Minable Commodities

In order to fit in with the existing and proposed economic aspects of Star Citizen, apart from the ship-specific frequency hopping patterns needed for remote attacks, all off the components needed to successfully digitally attack another player, are effectively just another piece of information. These can be bought, horded, sold, and used by the players. With the exception of the payloads, they would also be "minable" by the players, allowing them to be created dynamically which would aid in the player-generated economy. The following are the four types of hacking-related commodities:

- Firmware
- Ship Profiles
- Exploits
- Payloads

Firmware

There would be potentially three different types of firmware, but they would all behave more-or-less the same. Ship firmware would be specific to a particular brand and model of ship. Component firmware would be specific to individual ship components, again divided by component manufacturer as much as is appropriate for the lore of the game. Firmware modules are attained via ownership of the item in question, or by physically/digitally stealing it with the hacking techniques outlined above. Thus, all players automatically get access to the firmware for ships/items they own, perhaps by using a mobiGlass app obtained from a seedy character or mission giver. It follows then, that every player would have the mobiGlass firmware and their starting ship's firmware to begin with. Firmwares can also be purchased using the in-game currency.

Obtaining the firmware is a prerequisite for generating ship profiles, extracting encryption keys, and mining for exploits. It is not necessary to have the firmware in order to carry out an attack as long as the appropriate encryption keys, exploits, and attack payloads are owned by the attacker. However, if a player doesn't have a ship profile, the firmware can be used to get the necessary keys.

The firmware would also contain a version number that can be used to identify which exploits it is vulnerable to. See the exploit section below for details.

Ship Profiles

Ship profiles simply contain a particular ship's encryption keys along with the set number of frequency hops and bytes per hop used by that manufacturer's/model's ship. These are what is required to begin a remote or wireless attack on a target ship. They can be bought and sold just like any other tradable piece of information in the game. If one has obtained the firmware for the target ship, the keys can be extracted with a modest amount of compute power and the number of frequency hops and bytes per hop values manually discovered with **EtherShark**. Once a player has discovered these three data items, they can be packaged into a ship profile which can be traded.

Having the number of frequency hops and bytes per hop specific to a particular manufacturer and/or ship model would add another layer of differentiation to the ships themselves. For example, a military focused company such as Aegis may use a large number of hops and small amounts of data per hop, increasing the difficulty of the attack. On the other hand, a manufacturer such as Drake, who seem to prioritize just making it work, might use only one or two hops with large chunks of data per hop making them a more enticing target.

Exploits

Exploits can be generated or purchased. The exploits themselves have varying levels of "power." As with real world exploits, some are merely denial-of-service (Dos) attacks, some data leakage, some data corruption, and some allowing for remote code execution and full compromise, including persistance after disconnect. Exploits are specific to a specific component, whether that's a ship component, or add-on component of a higher quality. Some types of exploits are specific to remote attacks, where some might be physical access only. For remote attacks, there might be some exploits that require **EtherShark** to merely inject an exploit packet into the target radio. Some attacks might allow for the attacker to act as a man-in-the middle, thus requiring not only the attacker to lock onto the target signal, but also wait for a specific type of packet to traverse to/from the victim in order to modify it and thereby exploit the vulnerability. Clearly, the more powerful the exploit, the more difficult the attack can be.

Exploits can be chained together and/or used in parallel to provide more access to other systems or improve the power of the attack. For example, one exploit that grants only limited readability might be used in conjunction with one that allows writability, enabling certain values on a target's system to be both looked at and modified. Or one exploit that grants remote code execution on one system could be used to run an exploit that grants readability of a connected system's weapons systems, enabling the attacker to remotely discover what kinds and how many torpedoes or missiles the target has available.

The generation of exploits would be done by automated vulnerability discovery and exploit generating systems, (and again, I'll propose the names: **Terran fuzzy lop** and **SANgr**). These systems are almost entirely automated, apart from the player loading the firmware of the target device, with exploit generation of various types being controlled by game designer set probabilities. The ships with more computing power such as the Drake Herald and Crusader's Star Runner would be much better suited to finding exploits and quicker than ships without dedicated computer servers. The more powerful exploits should generally be less common and therefore more difficult to generate. Similarly, components of a higher quality should be less likely to have exploits found for them.

Once an exploit is generated, it can either be used by the player, given/sold to other players, or sold to either the manufacturer or the UEE under a bug-bounty program. Just as in real life, the manufacturers likely wouldn't pay as much as more… shady players/characters, but it would provide a revenue stream for morally upright players with computing resources to spare. It allows players to decide what kind of player/person they want to be in the game; white-hat, black-hat, or gray.

One of the advantages of this system is that it can be self-regulating in the live game environment. If a particularly powerful exploit becomes used too frequently or against high-profile targets, it becomes known to the UEE and/or the manufacturer, who will then issue an update/patch, thereby preventing that particular exploit from ever being used again. Manufacturers will also occasionally fix vulnerabilities on their own, regardless of how often it is used. This prevents both abusive players from ganging up on unsuspecting players and discourages hording of mined or purchased exploits. However, ships/components might only get the updated firmware once they go into a service station for repairs or are returned via an insurance claim.

Payloads

Payloads are the programs that actually makes use of the exploits; exploiting a system is pointless unless you can actually do something with it. Payloads are the only tradable commodity that cannot be generated by the players. This is because they will likely need some specific implementation by the game designers for each one. For example, an interface to pilfer data from a victim's data banks, remotely control a turret or ship console, open bay doors, or steal a ship from a Hurston ship retrieval console would all need a specific UI to be implemented in the game. These are not likely to be things players could reasonably be expected to make in game. The players would be able to buy and sell payloads, and/or obtain them from mission givers throughout the game. As a digital good, once acquired, the player can keep them indefinitely, allowing dedicated players focused on hacking to amass a significant arsenal of digital mayhem.

The one type of attack that wouldn't require a specific payload would be a wireless DoS attack, as all that's required is a powerful enough radio transmitter and the appropriate signal lock obtained through **EtherShark**.

The payloads and their effects on the target would have varying levels of detectability by the victim. Some payloads might be changing settings on various ship consoles, moving turrets, firing weapons, or adjusting power levels. Other payloads might be more difficult to detect. Finding the balance between how brazen the attack should be and how long you wish to go unnoticed should will depend greatly on the situation and will require skill to determine where that point is.

If a target discovers an active exploit/payload running on his/her ship or component (mobiGlass included), a full power cycle should be enough to purge the intrusion, forcing the attacker to re-connect and start over. For remote attacks, power cycling the offending radio should be sufficient to disconnect all but the most powerful of exploits that enabled some level of persistent storage and re-connect capabilities. For example, if a remote attack targeted at a thruster pack that allowed the attacker full control of the target ship's movement, if the pilot turned off the SAN bus attached radio, ship movement would be returned to the pilot. If, however, the pilot turns the radio back on, this particular exploit/payload combination would reconnect itself with the attacker once again. A full system shutdown of the ship would clear the attack, but would obviously leave the target vulnerable. These types of scenarios could lead to some very interesting and dynamic game play.

EtherShark

EtherShark is the proposed tool for managing much of the remote attacks. There are may ways of implementing the proposed game mechanics, so this section will simply focus on the proposed frequency hopping decoding mechanisms as demonstrated in the attached schack program.

Each target ship model would have a specific number of frequencies radio transmissions jumped between, and a set number of bytes per frequency. It is up to the player to with the appropriate manufacturer encryption keys to determine which frequency each hop is using, the duration of that hop, and if necessary, the offset into the first packet captured. When tuned to the wrong frequency hop-pattern, the ‘decoded’ data is gibberish and unreadable. In order to aid the player in decoding this, all radio data packets should have a semi well-formed format. The format used for the attached schack demo is as follows:

Field	**Size (b)**	**Description**
Packet Header	2	Denotes the beginning of a message. Consists of the ASCII characters: SOH STX (Start of heading, Start of text - 0x01 0x02).
Manufacturer Signature	4	A signature unique to each manufacturer, e.g. AEGS, ANVL, AOPO, BANU, CNOU, CRUS, DRAK, ESPR, KRIG, MISC, etc...
Component Signature	5	A signature unique to each component. Most remote attacks will likely be first detecting packets from the RADIO device.
Sender	n	Field denotes the sender of the data. TOML formatted field beginning with “sender=” and ending with ‘\n’
Receiver	n	Field denotes the receiver of the data. TOML formatted field beginning with “receiver=” and ending with ‘\n’
Type	n	The type of the transmission, which may aid in understanding the next field. The type, TOML formatted beginning with “type=” and ending with ‘\n’, is one of a fixed number of types, e.g. TEXT, VOICE, DATA, NAVIGATION, SCAN, PING, ACK, TELEMETRY, DRONE_C&C, etc...
Data	n	TOML formatted field beginning with “data=” and ending with ‘\n’. Data format is dependent on packet type field.
Packet Footer	2	Denotes the end of the message. Consists of the ASCII characters: ETX EOT (End of text, End of transmission - 0x03 0x04).

While this is what is used in the schack demo, improvements could easily be made without sacrificing enough structure to allow players to decode the messages. For example, the data field really shouldn’t use any special character to denote the end of the field, rather the packet should have another field of a known size that defines the length of the data field.

In any case, using only this information, it is possible to manually scrub through all allowable frequencies using only a single frequency to find bits and pieces of a collected packet that are decipherable. Using that information, it is simply a matter of looking at the data to determine the number of bytes per frequency. Initial packet offset can be set by searching first for the packet header and adjusting the display until the header is at the beginning of the decoding sequence. This means the

first packet may end up being discarded, but in an actual implementation this would be both unlikely and irrelevant except where speed might be of utmost concern.

Once the first frequency is discovered additional frequencies can be added to the decoding sequence and scrubbed through until the entire sequence is readable. While possible to do this with a brute force approach, it can become quite tedious, so several tools were added to aid in the frequency discovery. Firstly, the raw data is displayed. Because the frequencies are simulated by a simple xor operation, players who watch for new incoming packets can deduce what particular bytes in the message are supposed to be and xor the expected value with the codded value to obtain the frequency for that piece of the message. Skilled and technical players familiar with ascii encoding and binary operations will be rewarded by being able to nearly read the appropriate frequencies right from the raw data.

Secondly, for the more visually inclined players, spectral analysis graph is displayed that shows the frequency spectrum. This chart will show spikes that get taller at the frequencies the player has tuned **EtherShark** to. With a little trial-and-error, the appropriate frequencies can quickly be sussed out.

There will be the possibility of some very tricky combinations, such as a 4 byte-per-hop pattern with frequencies one and two set to the same frequency, leading to the assumption that the sequence is using an 8-byte pattern. Skilled players will be able to identify these conditions to quickly rule out false positive readings.

Using xor to simulate badly tuned frequencies may be computationally simple, but might not achieve the goal of appearing feasible to the technically inclined. My experience with signal analysis is quite limited, so I’m sure others can find more appropriate solutions. And overall, this is, of course, but one possible implementation that uses this concept. Many improvements could surely be found.

Example Game Loops

These are just a small sample of game play possibilities that could occur with the proposed system.

- Solo player waits near a landing pad for a target to land and its crew to leave the ship. Player cuts a hole in the hull gaining access to SAN bus, runs an exploit/payload to open the doors, enters and steals the ship/cargo.
- A solo player parks a Herald hidden in the mountains, just within range of a mining station. Remotely hacks the data banks of landing ships looking for high-valued cargo runs. The nav computer is also tapped to tell the player where the target might be heading. This information can either be sold to pirates, or forwarded on to other players who will actually perform the intercept.

- A multi-crew ship tails a cargo ship in quantum travel. The pilot keeps the attackers just inside signal range, but far enough away to avoid detection. The secondary player works **EtherShark** until signal lock is obtained, then executes an exploit against the cargo ships navigation system to cause the ship to over or under shoot the intended exit point.
- Players aboard a disabled ship that is in the process of being boarded hack the attackers ship, taking over remote control of the enemy ship turrets, firing upon the would-be hijackers EVAing across.
- Many many more… let the imagination run wild.

Other Random Thoughts Ideas

- EMPs would disrupt attacks
- Gives a(nother?) reason for having all the compute power some ships sport.
- Once physical access is gained, it might be possible to leave a radio dongle tap that allows direct SAN bus access remotely via radio. This of course would itself be vulnerable to attack.
- Exploit/payload management tool might be called StarSploit.