GHOST: Concealing Vehicular Botnet Communication in the VANET Control Channel

Mevlut Turker Garip, Peter Reiher, Mario Gerla
Department of Computer Science, University of California Los Angeles
{mtgarip, reiher, gerla}@cs.ucla.edu

***Abstract*—Vehicular ad hoc networks (VANETs) are expected to play a big role in our lives in the near future; they will both improve traffic safety and revolutionize the driving experience. Their expected deployment in autonomous cars will induce attackers to design new methods to target these systems, and to organize the vehicles they compromise into vehicular botnets. Vehicular botnets enable new attacks that reveal previously unknown security flaws in VANETs. Effectively defending against such botnets requires investigation of their characteristics and of the attacks that these cooperating malicious vehicles can perform on VANETs. One important characteristic of a botnet is the way its members communicate to coordinate their attacks, with an emphasis on stealth. In this paper, we investigate alternatives for vehicular botnets to communicate to perform attacks. We design and demonstrate a VANET-based botnet communication protocol that hides itself in the ongoing network traffic over the control channel. We show via simulation that it is infeasible to detect such botnet communications due to the vulnerabilities existing in the VANET standards, and discuss possible countermeasures.**

***Keywords*—Vehicular Ad hoc Networks, VANET Security, Vehicular Botnets, Vehicular Botnet Communication, Autonomous Cars**

I. Introduction

Traffic safety is a main concern for most countries [17]. Countries mainly focus on passing new traffic regulations and improving road infrastructure to prevent accidents; however, the improvement rates are still minimal. Therefore, exploiting modern technology is considered to be the most effective approach to improve traffic safety [7]. Insufficient traffic information and slow driver reaction are the root causes of most accidents [14]. VANETs can significantly improve traffic safety by providing information about traffic conditions spanning many miles and by delivering timely alarms that effectively prevent accidents. VANET-enabled autonomous vehicles could save 30,000 lives and prevent 2.2 million car accident injuries each year in the US alone [27]. In 2020, all cars are expected to have a certain degree of autonomy allowing the on-board computer to maneuver the car [30]. Major car manufacturers such as Audi, BMW, Ford, GM, Lexus, Mercedes-Benz, Nissan, and Volvo have already started prototyping their autonomous cars [27]. VANETs will not only increase the collision avoidance capabilities of these vehicles, but also make their deployment easier and faster, since manufacturing these cars will be cheaper without the dependence on highly expensive sensors.

History has shown that as computing and communications capabilities are added to new environments, inevitably some fraction of the new equipment is compromised by attackers. Adding computational capabilities to automobiles and connecting them by VANETs should be expected to follow the same path. Already researchers have demonstrated that computers and internal networks on board of cars can be compromised and exploited, allowing varying degrees of control of the vehicles [23] [28]. In some cases, the vehicles can be compromised remotely, using V2V unlicensed network connections from a distance. They can also be attacked from incautious Internet downloading via an otherwise secure LTE. As we increase cars' cyber capabilities, the attack surface they present and the increased number of targets will certainly attract cyber attackers, and some attacks will succeed in compromising the vehicles. As the internal computers take greater and greater control over vehicular capabilities, leading eventually to autonomous cars, the reward a successful attack can harvest will only make compromising them a more attractive target. Earlier work has proven these autonomous cars can be compromised [22] [24]. A compromised vehicle of this kind not only offers computation and communication capabilities, but the ability to perform real world actions that can have severe consequences.

Some attention has been paid already to the cybersecurity of autonomous and networked vehicles [18], particular to ensuring that vehicles can provide strong authentication for their messages [21]. Existing research on these lines has been primarily oriented towards preventing vehicle owners from lying to other vehicles and roadside infrastructure. Such research offers some leverage against attackers who compromise a single vehicle. However, attackers who realize that they can compromise a single machine quickly move on to multiple machines, and to organizing them into a distributed resource of greater total power than its individual parts.

We should thus expect that the future will bring us *vehicular botnets*, collections of autonomous and networked vehicles under the coordinated control of remote attackers. These botnets can enable various attacks that have not been investigated yet. In VANETs, as with the Internet, botnet attacks are more powerful and harder to defend against than attacks from single sources. Similar to Internet botnets, effective defense against vehicular botnets requires that we first understand the nature of the attacks they can perform and how their communication works. In this paper, we demonstrate a VANET-based botnet communication session that hides in the control channel traffic. We believe that the most successful defense against any type of botnets starts with a detailed analysis of their communication and investigation of the ways to disrupt it.

In Section II, we present the related work in VANET security and compare Internet botnets with vehicular botnets in terms of communication. In Section III, we demonstrate our vehicular botnet communication protocol. In Section IV, we show the performance of botnets in action and experimentally prove the challenge to detect them. In Section V, we discuss possible solutions to detect and disrupt botnet communication.

II. Related Work

The existing work on security of autonomous vehicles and VANETs is primarily related to bad behavior or compromises of individual vehicles, with a related body of research on privacy issues in VANETs. [21] provides a good survey of the early research on these topics. This paper mentions the possibility of multiple cooperating malicious entities in the VANET, but does not discuss them in detail. Following work on defenses against security problems in VANETs focused on ensuring the correctness of cryptographic authentication [11]. More recent work looked at tamperproof hardware in cars [1], [10], [18] and analysis of messages sent by VANET members to evaluate plausibility [2], [3], [6], [9], [13], [15], [20], [26], [33] or trustworthiness of the sender [4]. [19] assumes cooperating attackers, but does not discuss them in the context of a botnet. While the general approach of analyzing messages for plausibility and evaluating trust in nodes has promise, it does not attempt to identify cooperating attackers, or at best assumes that a majority of the data comes from honest vehicles. Therefore, none of these solutions consider vehicular botnets and investigate defenses against them.

Attackers form botnets by first compromising individual entities. More recently, a number of papers have described successful cyberattacks on particular vehicles [12], [22], [23], [24], [28]. These attacks demonstrate that existing vehicles with computer and communications capabilities are already vulnerable and can be compromised. As advanced VANETs and autonomous vehicles are developed and deployed, it is a valid assumption that vehicular botnets will be created by organizing these compromised autonomous cars. We demonstrated the first vehicular botnet attack in the literature in [8] and showed how powerful such attacks can be by causing severe congestion on any road of our choice. This particular botnet attack did not require vehicular bots to communicate with each other; however, we can expect future attacks to utilize a vehicular botnet communication protocol for collaboration. Therefore, investigating the characteristics of such communication is crucial for defending against vehicular botnets, especially considering that the most effective way to stop a botnet is taking down its communication [16]. This paper is the first that addresses vehicular botnet communications.

The functional requirements of a vehicular botnet communication are similar to Internet botnets; they need to protect the identity of botmaster and bots, avoid detection, and be resistant to the loss of several bots. However, how these requirements can be satisfied by vehicular botnets is fundamentally different than the Internet botnets. This is because the characteristics of the communication over the Internet and VANETs are fundamentally different. Unlike the Internet, in VANETs vehicles have to use a shared medium to communicate, which every other vehicle can listen to (i.e., control and service channels). In order to protect the identity of botmaster, the same approach with Internet botnets can also be used by the vehicular botnets, which is using multiple levels in their command and control (C&C) hierarchy to reach the botmaster. However, ensuring the infeasibility of getting detected during the communication among bots is much more challenging in VANETs than it is in the Internet. How we achieve to satisfy these botnet communication requirements—while at the same time being restricted by the characteristics of VANETs—is explained in detail in the subsequent sections.

III. Vehicular Botnet Communication

Creating a vehicular botnet requires building a communication infrastructure to organize and control the compromised vehicles so that the attacks can be performed cooperatively by these vehicles. Since using vehicles' Internet connections for communication among vehicular bots will raise a flag, our vehicular botnet communication needs to use the existing VANET infrastructure. Using existing communication mechanisms of the current Internet botnets for vehicular botnets is not feasible since the Internet architecture is fundamentally different than the VANET architecture. Therefore, we designed a vehicular botnet communication protocol that takes advantage of the existing VANET infrastructure to control the vehicular bots and facilitate collaboration for vehicular botnet attacks.

A. Design Overview

The VANET communication infrastructure has two different wireless channels that vehicles can use: the control and service channels. Basic Safety Messages (BSMs) are broadcast through the control channel; other non-safety applications use the service channel. Using the service channel to exchange many botnet messages among a particular set of vehicles would be suspicious, so we decided to hide the vehicular botnet communication in the plain sight. Vehicular bots will inject their messages into BSMs, which are already being flooded across the network, but no vehicle except our vehicular bots will know the existence of these secret messages and be able to decode them. We designed the injection mechanism in a way that the altered BSMs will not look suspiciously different than the normal BSMs. In addition, we do not inject botnet data into every BSM and we constantly change the injection frequency to further decrease the possibility of raising suspicion; details are explained in the subsequent sections.

	BSM Data Item	Bytes	BSM Data Item	Bytes
	Message ID	1	Heading	2
	Message Count	1	Steering Wheel Angle	1
	Temporary ID	4	Accelerations	7
24 cm	Time	2	Brake System Status	2
$15x10^{-7}$ degrees	Latitude	4	Vehicle Size	3
$15x10^{-7}$ degrees	Longitute	4	Event Flags (opt)	2
	Elevation	2	Path History (opt)	Var
0.08 degrees	Positioning Accuracy	4	Path Prediction (opt)	3
0.67 mil/hrs	Transmission & Speed	2	RTCM Package (opt)	Var

Figure 1. BSM fields where botnet messages will be injected into and the effect of the injection on the field values.

Not all the fields in a BSM can be altered without causing implausible data and raising any flags. We alter only four fields in a BSM where an injection can stay undetected based on the resulting value of each field. Figure 1 shows the content of a BSM according to the IEEE 802.11p standards [32]. The fields that we use for injection are colored green: "Transmission & Speed" for the vehicle's current speed, "Positioning Accuracy" for the noise in calculation of the vehicle's heading relative to true north, "Latitude" and "Longitude" for the vehicle's GPS location. We inject half a byte to each field by replacing the

least significant 4 bits of the field. We limit the size of each injection to half a byte so that the changes due to the injection in the field values are not suspicious. As shown in Figure 1, our injection can change the value of the speed field at most 0.67 miles/hour, positioning accuracy field at most 0.08 degrees and GPS position at most 24 centimeters (max. on equator), which are completely natural variations even without an injection.

B. Vehicular Botnet Message

The size of each vehicular botnet message is 2 bytes, which will be divided into four equal parts to be injected into a BSM. Each botnet message has 3 fields as shown in Figure 2:

Figure 2. Botnet message that will be injected into BSM.

Stream/Attack ID: In a vehicular botnet, the same botnet communication infrastructure has to be used for different attacks, sometimes even simultaneously. Each bot should be able to separate data streams from each other based on which attack each stream belongs to. Therefore, we have a stream/attack ID field in the botnet message which specifies which stream this message belongs to so that bots can separate different data streams. We allocated 3 bits in the botnet message for the stream ID, which supports performing up to 8 different vehicular botnet attacks simultaneously.

Character Offset: Due to the limited size of injection in order to stay under the radar, each botnet message has only a single byte allocated for the payload. The data that a bot needs to send is divided into individual characters, each of which is sent with a separate botnet message. Since BSMs altered by these botnet messages might arrive out of order at the receiver bots, we allocated 5 bits in the botnet message to hold the character offset so that the data received can be reconstructed with the original order. Therefore, at most 32 characters can be sent in a single stream due to 5-bit character offset; multiple streams can be used if larger data needs to be sent, decreasing the number of simultaneous attacks possible.

C. Communication Secrecy

Our vehicular bots do not communicate with each other through the Internet, a separate wireless frequency or an individual service channel in the VANET, existence of which might get detected. Our botnet communication is completely hidden in BSMs, which are broadcast and flooded to the entire VANET. Detecting the existence of secret messages embedded in BSMs is difficult since the changes in the field values after injections are negligible and expected due to the noise in sensor readings. Our mechanism discussed in the subsequent sections is designed to provide confidentiality if anyone ever suspects that vehicular botnets use BSM broadcasts to communicate.

Even though one cannot guess which and how many fields bots are injecting their messages into and the size of each injected botnet message by just monitoring BSM exchanges, we still decided to assume that this information can be obtained just in case police might apprehend one of our bots and access the source code of our vehicular botnet software.

In Section III-F, we show that, even in this situation, our mechanism can still ensure the confidentiality of the exchanged botnet messages by the use of tokens and periodic password updates. In the next few sections, we describe how a compromised vehicle joins the vehicular botnet and the mechanism by which it periodically mutates the injection scheme.

D. Initialization of Bots

Bots do not use their Internet connection to communicate with the other bots. However, they will use it for occasional communications with the botmaster (attacker who owns the botnet). There are two packet types for this communication between the botmaster and bots: *password request* packet and *password response* packet. These packets are used by the botnet not only for the initial setup, but also for synchronization and confidentiality purposes even after the initial setup, which is further explained in Section III-F.

When a vehicle first gets compromised, our botnet software is downloaded onto the vehicle. The software comes with a unique token that is different for each compromised vehicle; these unique tokens are generated by the botmaster and can only be used once and only for the initial setup. The compromised vehicle will send a *password request* to the botmaster for the first time with this token and its "Temporary ID" in order to obtain a unique bot ID and the current password in the botnet, and register its "Temporary ID". This temporary ID is a unique identifier for each car as required by the IEEE 802.11p standards [32] (see Figure 1). After receiving this *password request*, the botmaster will then check its *unused token database* and confirm if the token exists. If it exists, the botmaster will generate a unique bot ID, add the (bot ID, token) pair to its *used token database*, add the (bot ID, temporary ID) pair to its *bot database*, and send a *password response* that contains this bot ID, the current password in the botnet and a list of all the temporary IDs in the *bot database* back to the compromised vehicle. Then, the vehicle overwrites its *bot list* with this list sent by the botmaster.

E. Botnet Message Broadcast

The current password in the botnet is used to configure the injection mechanism. It is constantly updated by each bot at the same time, thus, the injection configuration mutates with it. Since the next password is generated from the previous password and every bot updates its password at the same time in a distributed manner, all the bots will be using the same injection mechanism at all times despite the constant mutation. Bots synchronize their times based on the GPS timestamp, avoiding issues of clock synchronization. The *password structure* in each bot is designed as follows:

Password: The current password in the botnet. All the following values are set based on the current value of the password.
Injection order: This determines which permutation of the half bytes of botnet message will be injected to the four fields (e.g., fourth half byte is injected to the longitude field while first is injected to the speed field in the BSM).
Encoder/Decoder ID: This specifies which encoder-decoder function pair should be used by both sender and receiver bots to encode the character sent and decode the character received.
Injection Frequency: This determines the frequency at which injections on BSMs will occur (e.g., inject every fifth BSM).

Next Password Update Time: This specifies the time that current password must be updated by each bot so that all bots can switch to the same configurations without any coordination.

Each bot continuously checks BSMs, at a frequency determined by current configuration to extract any injected character. First, each bot will check if the temporary ID associated with the BSM is in its *bot list*, which is the only way to understand whether the BSM is coming from a bot or not. If it exists in the list, then each bot will extract the 2-byte botnet message according to the current active injection configurations, and check whether it is a valid message or not. An invalid botnet message is indicated by setting all its bits to 1. The reason that this special botnet message exists is that every bot has to inject something to the BSM when it is time to inject, regardless of whether there is a character to send in the bot's message buffer. Therefore, each bot needs to inject this special botnet message (all bits are set to 1) if all of its message buffers are empty, to let other bots know that it did not have any character to send. If the extracted botnet message is valid, it will be placed at the correct offset in each receiver bot's message buffer belonging to the attack specified by the *Stream/Attack ID* in the botnet message. If not, the extracted botnet message will be discarded.

When botmaster wants to send a message to all of the bots, it chooses one or more representative bots based on the sparsity and number of the bots. It sends the message to the representative bot(s) over the Internet along with the associated *Stream/Attack ID*. The representative(s) will break the message into individual botnet messages with the corresponding character offsets and attack ID, and inject one botnet message at a time to BSMs according to the current active injection configurations. Received botnet messages will be saved by the other bots in their associated message buffers. When the next injection time comes, the other bots will then randomly choose a character among all of the received characters so far in their message buffers, and broadcast it like the representative(s) according to the configurations at that time.

F. Discussion on Secrecy and Consistency

Our botnet communication is resilient to attempts by the authorities to eavesdrop on the messages being exchanged among bots. Without knowing the current password, no one will be able to guess the current active injection configurations and reconstruct the botnet message. Before one might figure out the current configurations by brute force, they will already have changed due to the constant password updates. Successful brute force in a timely manner is infeasible because one needs to consider every BSM as injected, try all x! * y combinations (x=number of fields, y=number of encoding/decoding functions) and filter out the combinations that result in invalid messages before the password is updated.

One can try to brute force a token to be able to tap the botnet communication. However, each token is created in the moment of compromising a vehicle and immediately used by the compromised vehicle to join the vehicular botnet. Therefore, window of opportunity to brute force this token is extremely small, making this approach infeasible.

The only problem that might occur is that police might apprehend one of our bots and access the botnet communication software. Police can obtain the token associated with this captured bot, but due to not allowing token to be used more than once, the token cannot be used by the police car to join the botnet. However, police can still use the captured bot itself to eavesdrop on the communication; therefore, we implemented a technique to remove the apprehended bots from our vehicular botnets. The botmaster generates a new random password every 13 seconds, which is independent from the previous password. It sends a *password response*, which contains this password and the list of current bots' temporary IDs, to all the bots in its *bot database*. When the botmaster detects that one of the bots got apprehended, it will remove the bot from its *bot database* and its associated token from the *used token database*. Therefore, the password generated by the botmaster will not be sent to the apprehended bot, and the bot list that is sent to all the other bots will not contain the captured bot. This way, the apprehended bot will be out of sync with the other bots and will not be able to extract botnet messages properly. As a result, even if police obtains the current password in the botnet communication, it will be obsolete in at most 13 seconds after the apprehension is detected. How a botmaster can detect the apprehension of bots is for future work. One way would be making the bot send a message to the botmaster if the executable of its botnet communication software is accessed for reading. Read—rather than execute—access will indicate an attempt for reverse-engineering.

Between these random password generations, the botmaster keeps updating its password based on the previous password periodically like all the other bots do. Bot vehicles, which were turned off by their owners for some time because they were parked, will miss several password updates and not be able to communicate with the other bots. Therefore, when they get turned back on, they will send a *password request* to the botmaster with their valid bot IDs to receive the current active password that the other bots are using in order to catch up with the current botnet communication.

As aforementioned, bots use their *bot list* and the temporary IDs associated with received BSMs to determine if they are sent by the bots. However, as mandated by the IEEE 802.11p standards, every vehicle updates its temporary ID periodically for privacy reasons. Therefore, every time a bot updates its temporary ID, it will send a *password request* to the botmaster with its bot ID and the updated temporary ID. The botmaster then will update its *bot database* so that it can be shared with the other bots through the periodic *password response*.

IV. Evaluation

We used Veins [25] (which combines the SUMO and OMNeT simulators) to conduct experiments to evaluate our vehicular botnet communication. SUMO is responsible for simulating realistic vehicular traffic while OMNeT is used to simulate the communication capabilities of the vehicles with IEEE 802.11p integration [32].

	Average	Max	Theoretical Max
Latitude (d)	5.02E-07	7.67E-07	15E-07
Longitude (d)	5.25E-07	7.73E-07	15E-07
GPS Position (cm)	8.08	12.11	24
Positional Accuracy (d)	0.02	0.03	0.08
Speed (Miles/Hour)	0.29	0.40	0.67

Figure 3. Changes in the values of the injected fields during the experiment

First, we show that it is infeasible to detect our botnet communication. Figure 3 shows the average changes on the

four BSM fields due to the injection of botnet messages, as well as the maximum changes observed during our experiments. It can be seen that actual changes on these fields are much lower than the theoretical maximum values that we anticipated in Section III-A. Changes in the speed values stay under the speedometer errors tolerable by the UN regulations [29]. Changes in the GPS position and positional accuracy are much lower than the expected errors in the sensor readings (≈1 meter for the state-of-the-art GPS sensors [31] and 1 degree for the magnetic compasses [5]). These results show that it is infeasible to detect our botnet communication.

Packet Loss (%)	0.0	0.5	1.0	1.5	2.0
Average Sync Time (s)	20.43	20.84	18.02	20.91	22.80

Figure 4. Average synchronization times with different packet loss rates

Our botnet communication is resilient to packet loss. We tested our systems with different packet loss rates and measured, for each rate, the average time it takes for a bot to receive the complete message associated with an attack (average sync time). Figure 4 shows average sync times mostly increase as the loss rate increases, but the changes are negligible; bots receive complete message in a timely manner.

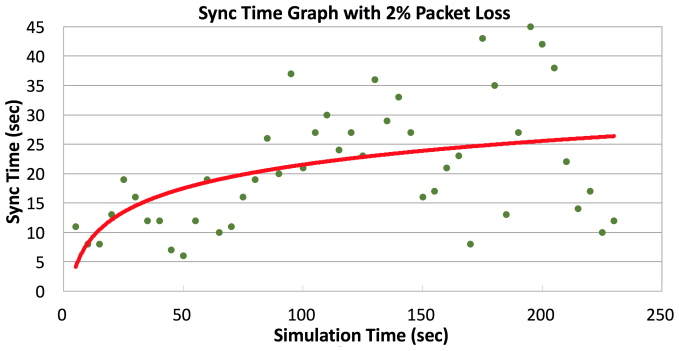

Figure 5. Sync time graph when the packet loss rate is 2%

Figure 5 shows how resilient our botnet communication is to packet loss and explains the reason of the results in Figure 4. Sync times increase throughout the simulation as more packets are lost, but our system quickly adapts to the lossy links and stabilizes at acceptable values. It stabilizes sync times by a randomization strategy we designed for advertising botnet messages. For each botnet message, bots broadcast a character at a random index in their message buffer so that the variety of characters being advertised at each time is maximized. As a result, the probability for a bot to receive a character which it has not received yet will increase and sync times will improve compensating the existing packet loss rate.

Bot Percentage (%)	5	10	20
Average Sync Time (s)	11.10	12.96	20.43

Figure 6. Average synchronization times with different botnet percentages

We also investigated how scalable our botnet communication is when a large number of bots use the botnet communication, by testing our system with different bot percentages. Figure 6 shows the average sync times when 5%, 10% and 20% percent of the vehicles in the simulation are bots. As expected, the average sync time increases as the bot percentage increases due to the computation overhead and packet loss introduced by this overhead. However, the increase is logarithmic and therefore tapers off before exceeding acceptable sync times.

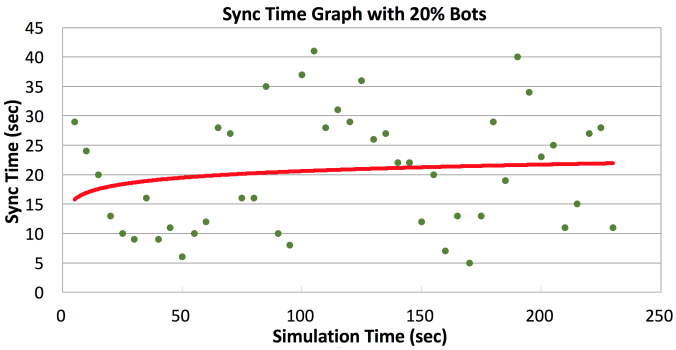

Figure 7. Sync time graph when 20% of the cars in the simulation are bots

Figure 7 shows how fast our botnet communication adapts to the overhead caused by the large number of bots when the bot percentage is 20%. It stabilizes sync times this quickly by exploiting the large number of bots for the aforementioned randomization strategy; when the number of bots is high enough, this strategy successfully compensates for the packet loss induced by the overhead. Also, the stabilization of the sync times in Figure 7 is much faster than it is in Figure 5 with high packet loss, since the effectiveness of our randomization strategy increases as the number of bots gets higher.

V. Possible Countermeasures

The success of our vehicular botnet communication protocol depends on the infeasibility of detecting the communication. Even though we implemented mechanisms to prevent others from eavesdropping on the botnet messages, authorities can still remove the bots from the network if the botnet communication gets detected in any of the bots' messages. However, currently no one can accuse any vehicle of participating in a botnet communication since the changes in the values of injected BSM fields are negligible and even less than the natural variations in these values due to noisy sensor readings (see Figure 3). The reason that the injections go unnoticed is how the VANET standards define BSM fields. The precision of the fields that we chose to inject and the allocated number of bytes for them in BSMs help half-byte injections not to change the values noticeably. For example, "Transmission & Speed" field in the BSM counts in the unit of 0.02 meters/sec (0.04 miles/hour). There is no reason to have a precision this high for a value that is already very noisy. Therefore, one defense against our botnet communication would be to change the standards for BSMs to get rid of the unnecessary precisions and to avoid allocating more bytes for the fields than required. For instance, decreasing the size of the injected fields by half a byte will decrease their precision by a factor of 16, which is sufficient to make the injections detectable.

If the injections on the BSMs begin changing the field values more than the expected natural variations, machine learning techniques could be used to detect anomalies in the values of these BSM fields. Some anomalies can be as simple

as an acceleration of 20 miles/hour in one second, but detecting some of them may require monitoring the exchanged BSMs in the whole network and learning for a significant amount of time. After an anomaly is detected in the BSMs of a vehicle, it can be put under further surveillance to identify the other bots, or can be immediately removed from the network and apprehended by the authorities. Though, it is not a straightforward task to remove a vehicle from the network. The most effective way to achieve this is a question for future research. Maybe a reputation-based security system can be used to score each vehicle based on its trustworthiness.

An additional approach can be analyzing the Internet traffic from/to individual vehicles to identify the botmaster, or to at least identify the bots that have suspicious Internet traffic if the botmaster is untraceable. Identifying and shutting down the botmaster is the most effective way to disrupt a botnet communication; however, it is not trivial to do so. A lot of earlier research has been conducted on this subject in the context of Internet botnets; perhaps similar mechanisms can deal with the vehicular botnets.

VI. Conclusion

In this paper, we demonstrated the first vehicular botnet communication in the literature. We argued that the most effective defense against vehicular botnets, as all other types of botnets, is investigating the characteristics of the communication they use to perform their attacks and designing defense mechanisms to disrupt it. We prototyped one such communication mechanism to allow us to study it. We showed that it is infeasible to detect our botnet communication due to the existing vulnerabilities in the VANET standards. We demonstrated the resilience of our botnet communication to lossy channels and that it is scalable even if there are a large number of bots on the map. We discussed existing vulnerabilities in the VANET standards and possible countermeasures. We believe that vehicular botnets can facilitate dangerous attacks performed on VANETs, and that much more research needs to be done to effectively fight against it. This paper serves as an important first step toward designing a general defense that can prevent all types of vehicular botnet attacks since they all will have to use a botnet communication protocol.

VII. Acknowledgement

We would like to thank Paul Hyungmin Kim for his great work contributing to the implementation and experimentation of the botnet communication, and his valuable feedbacks.

References

[1] The evita project. http://www.evita-project.org, 2008.

[2] N. Bismeyer, S. Mauthofer, K. M. Bayarou, and F. Kargl. Assessment of node trustworthiness in vanets using data plausibility checks with particle filters. In *IEEE VNC*, 2012.

[3] N. Bismeyer, C. Stresing, and K. Bayarou. Intrusion detection in vanets through verification of vehicle movement data. In *IEEE VNC*, 2010.

[4] F. Dotzer, L. Fischer, and P. Magiera. Vars: A vehicle ad-hoc network reputation system. In *IEEE WoWMoM*, 2005.

[5] B. Edwards. Magnetic compass accuracy, sighting, and triangulation. http://sectionhiker.com/magnetic-compass-accuracy/, 2014.

[6] M. El Zarki, S. Mehrotra, G. Tsudik, and N. Venkatasubramanian. Security issues in a future vehicular network. *European Wireless*, 2, 2002.

[7] L. Evans. A new traffic safety vision for the united states. *American Journal of Public Health*, 93(9):1384–1386, 2003.

[8] M. T. Garip, M. E. Gursoy, P. Reiher, and M. Gerla. Congestion attacks to autonomous cars using vehicular botnets. In *NDSS*, 2015.

[9] P. Golle, D. Greene, and J. Staddon. Detecting and correcting malicious data in vanets. In *ACM VANET*, 2004.

[10] G. Guette and C. Bryce. Using tpms to secure vehicular ad-hoc networks (vanets). In *Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks*, 2008.

[11] J. Hubaux, S. Capkun, and J. Luo. The security and privacy of smart vehicles. *IEEE Security & Privacy*, 2(3):49–55, 2004.

[12] I. Roufa et al. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *19th USENIX Security Symposium*, 2010.

[13] T. H. J. Kim, A. Studer, R. Dubey, and X. Zhang et al. Vanet alert endorsement using multi-source filters. In *ACM VANET*, 2010.

[14] S. G. Klauer, F. Guo, J. Sudweeks, and T. A. Dingus. An analysis of driver inattention using a case-crossover approach on 100-car data: Final report. *U.S. Department of Transportation No. HS-811 334*, 2010.

[15] T. Leinmuller, E. Schoch, and F. Kargl. Position verification approaches for vehicular ad hoc networks. *IEEE Wireless Communications*, 13(5):16–21, 2006.

[16] Y. Nadji, M. Antonakakis, R. Perdisci, D. Dagon, and W. Lee. Beheading hydras: Performing effective botnet takedowns. In *ACM SIGSAC Conference on Computer & Communications Security*, 2013.

[17] W. Odero, P. Garner, and A. Zwi. Road traffic injuries in developing countries: a comprehensive review of epidemiological studies. *Tropical Medicine & International Health*, 2(5):445–460, May 1997.

[18] B. Parno and A. Perrig. Challenges in securing vehicular networks. In *ACM Hotnets IV*, 2005.

[19] J. Petit, M. Feiri, and F. Kargl. Spoofed data detection in vanets using dynamic thresholds. In *IEEE VNC*, 2011.

[20] R. K. Schmidt et al. Vehicle behavior analysis to enhance security in vanets. In *IEEE V2VCOM*, 2008.

[21] M. Raya and J. P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[22] S. Rosenblatt. Car hacking code released at defcon. http://news.cnet.com/8301-1009_3-57596847-83/car-hacking-code-released-at-defcon/, 2013.

[23] S. Checkoway et al. Comprehensive experimental analyses of automotive attack surfaces. In *20th USENIX Security Symposium*, 2011.

[24] G. Smith. Driverless car could be hacked by 14-year-old from indonesia, senator warns. http://www.huffingtonpost.com/2013/05/17/driverless-car-hack_n_3292748.html, 2013.

[25] C. Sommer. Veins: Vehicles in network simulation. http://veins.car2x.org, 2015.

[26] H. Stubing, J. Firl, and S. A. Huss. A two-stage verification process for car-to-x mobility data based on path prediction and probabilistic maneuver recognition. In *IEEE VNC*, 2011.

[27] C. Tannert. Self-driving cars: Inside the road revolution. http://www.fastcompany.com/3022489/innovation-agents/self-driving-cars-let-go-of-the-wheel/, 2014.

[28] A. Tutu. Tesla model s vulnerable to cyber attacks. http://www.autoevolution.com/news/tesla-model-s-vulnerable-to-cyber-attacks-79407.html, 2014.

[29] UN Economic Commission for Europe. UN vehicle regulations. http://www.unece.org/trans/main/wp29/wp29regs21-40.html, 2016.

[30] D. Undercoffler. 54 million self-driving cars will be on the road by 2035, study finds. http://articles.latimes.com/2014/jan/02/autos/la-fi-hy-autos-ihs-autonomous-cars-study-20140102, 2014.

[31] U.S. Department of Defense. Gps standard positioning service performance standard. http://www.gps.gov/technical/ps/2008-SPS-performance-standard.pdf, 2008.

[32] Wireless LAN Working Group. Wireless access in vehicular environments. *IEEE Standards*, July 2010.

[33] G. Y. Yan, G. Choudhary, M. C. Weigle, and S. Olariu. Providing vanet security through active position detection. In *ACM VANET*, 2007.