Defeating DOMINO to Obtain Unfair Amounts of Service in IEEE 802.11 Hotspots

Chris Henk, Denver Brittain
cahenk@smu.edu, dbrittain@smu.edu

***Abstract*—In this paper, we present three attack methods which exploit vulnerabilities in DOMINO in order to receive unfair amounts of bandwidth from a wireless IEEE 802.11 hotspot. DOMINO purports to protect against MAC layer misbehavior techniques by implementing statistical analysis functions in access point firmware, but it suffers from serious vulnerabilities that make this claim untenable. This allows an attacker to either evade detection or eject other nodes from the network, making DOMINO improper for real world deployment. We prove that these vulnerabilities exist by exploiting them with *false flagging*, *framing*, and *virtual stations*. These attacks can be implemented with simple modifications to open source firmware and drivers on commodity hardware. Our testing shows that we can either evade detection or eject nodes 100% of the time against a standard DOMINO implementation. We conclude that DOMINO and other protocols using statistical analysis heuristics are entirely defeated by these attacks and can be exploited to make MAC layer misbehavior techniques more effective.**

I. Introduction

With the high costs of cellular data plans and ever growing number of wireless devices, community hotspots have become essential tools for consumers, schools, small businesses, and enterprises alike. IEEE 802.11 hotspots are inherently vulnerable to a number of attacks allowing for one or more nodes on a network to gain an unfair share of resources, increasing their level of service at the expense of all other nodes on the network [5]. Unless proper protection mechanisms are in place, a hotspot cannot guarantee fairness, where each node has equal access to the network. Existing defensive protocols that purport to achieve fair allocations and prevent cheating suffer from significant deficiencies that have received little attention in the literature.

One of the best known protocols to defend against MAC layer misbehavior, proposed by Raya et al, is DOMINO (Detection Of greedy behavior in the MAC layer of IEEE 802.11 public NetwOrks) [6]. However, DOMINO has significant vulnerabilities that an attacker can easily exploit with commodity hardware to defeat it. The general architecture and design philosophy of DOMINO is used in [9], [8], and [7], meaning that they, along with other DOMINO based protocols, suffer from the same deficiencies. These exploits are the result of the protocol's inability verify the authenticity of a corrupted frame. New misbehavior techniques that take advantage of this flaw make DOMINO, and similar protocols, insufficient for real world use.

In this paper, we present a new class of misbehavior techniques in which the attacker is forging frame collisions to defeat DOMINO, along with three examples. This new class can be divided into two subtypes, A) those that use the detection mechanism to eject competing nodes from the network and B) those that simulate network congestion to avoid detection. We introduce *false flagging* and *framing* as examples of subtype A and *virtual stations* as an example of subtype B.

To demonstrate the effectiveness of these techniques, we modified the open source drivers for a commodity wireless card [10]. Additionally we tested our prototype on existing wireless hotspots at Southern Methodist University. We utilized a second commodity wireless card to record MAC layer errors in addition to a standard packet capture using Wireshark. This data was then fed into a static analyzer which implements DOMINO. The output from the analyzer was used to determine what would be detected and what corrective actions, if any, would be taken had the wireless hotspots implemented DOMINO.

We were able to obtain significant amounts of additional bandwidth obtained from an unsecured network without being detected by DOMINO in attacks that did not seek to eject competing nodes. This resulted in a high level of net performance increase depending on the level of congestion and aggressiveness. When attacking with the intent to eject competing nodes, we were able to successfully eject competing nodes without being detected in DOMINO. In fact, this strategy was so effective that it performed better than if no defensive protocol had been employed and the attacker had to continue reactive jamming.

We conclude that this new class of misbehavior techniques completely defeat DOMINO and are easily implementable by an attacker in the real world using commodity hardware. DOMINO and similar heuristic based protocols that cannot guarantee the authenticity of corrupted frames are vulnerable to these techniques and there is a high potential for abuse. This vulnerability is so substantial that an attacker exploiting it to eject other nodes will be able to cheat more effectively than if no defensive protocol had been deployed at all. Therefore, there is a clear need for a robust defensive protocol that is capable of determining if frames are authentic. This need will only increase as wireless hotspots continue to proliferate.

The remainder of this paper is organized as follows. In Sections 2 & 3 we introduce WiFi and DOMINO. Section 4 reviews related work. In Section 5 we present our methods for defeating DOMINO in detail. Section 6 covers our testing environment and results. In Section 7 we provide an analysis of the data. This is followed by a brief discussion of potential

methods to defend against the attacks previously introduced in Section 8 and our final conclusions in Section 9.

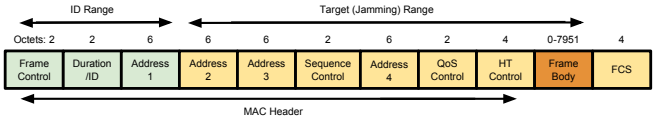
II. IEEE 802.11 MAC Layer

WiFi solves the shared medium issue inherent to wireless communication with Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA outlines, very basically, that a node may only transmit when a channel is sensed to be empty or idle.

Binary Exponential Backoff, found in section 8.4.2.24.11 of the standard[3], is the method in which nodes will decrease their probability of transmission after a collision occurs.

Nodes also have the ability to spoof MAC addresses. This is in fact, so accessible in modern hardware and software that popular operating systems such as Microsoft's Windows 10 have a setting to spoof a random MAC address when connecting to public hotspots to protect against user tracking. The ease at which a node may lie about there MAC address means that 802.11 has no mechanism to verify the authenticity of any node claiming to hold a particular address, an important consideration in advanced MAC layer misbehavior techniques.

The assumptions made by these protocols are exploitable without proper protection as they are contention based and simply assume that every node is behaving properly. The 802.11 standard itself does not offer any protection against the methods of attack presented here.



III. DOMINO

The DOMINO protocol addresses the absence of any protection against MAC layer misbehavior in IEEE 802.11. It does so by determining the deviation of each node's behavior from its expected behavior. The architects determined that a solution must meet three key requirements while also addressing the greedy node problem:

1) Seamless integration with existing wireless access points. Achieved through a statistical passive approach to traffic monitoring.
2) Compatibility with existing networks and network topologies.
3) Future applicability to 802.11 changes.

While useful to understanding the purpose of DOMINO as an applicable solution to greedy MAC layer behavior, the aforementioned requirements have little impact on the actual detection process apart from stating that it must be passive in both nature and action to prevent integration issues.

The component of the DOMINO architecture's detection applicable to our attacks is array of tests it runs in order to classify the behavior of each node on the network as either malicious or non-malicious. The tests include:

1) Identifying scrambled frames in order to detect CTS/ACK/DATA scrambling attacks.
2) Discovering backoff times that are shorter than DIFS to find a greedy node.
3) Determining which nodes have regularly sent large NAV values in order to increase their transmission's duration.
4) Tracking the maximum backoff executed by each node on the network in order to discover those lower than a threshold value.
5) Average actual backoffs are measured to detect malicious nodes without falsely accusing properly behaving ones.
6) Consecutive backoffs are also measured to detect malicious nodes without falsely accusing properly behaving ones.

DOMINO functions by periodically collecting traffic traces of all sending stations on the local network and analyzing that data with the aforementioned array of tests to classify each node's behavior. The final classification is done with respect to the average behavior of all nodes on the network. Flagging greedy nodes is a function of deviation from the averagely behaved node on the local network.

One of the major shortcomings of DOMINO is its counting of corrupted frames as a function of behavior. A node with an unusually low number of corrupted frames will be classified as malicious by DOMINO. This is a vulnerability that we will exploit later in the paper using MAC and IP spoofing.

A purported benefit of DOMINO is its ability to function undetected in that a potentially malicious network user will not know if the access point is using DOMINO until the user is penalized for greedy behavior. However, this claim is only valid if nodes are never falsely punished. A potentially malicious network user that can trick the network into punishing an innocent user can infer whether or not DOMINO is running based on whether or not the victim receives punishment.

There are, however, benefits to DOMINO. Passive traffic analysis is also greatly beneficial as it reduces the load on the AP and does not require specialized hardware. This makes implementation simple and relatively easy as it does require any major modifications to the AP itself. Additionally, it's modular design allows for a high degree of customization. Short implementation times and a diversity of detection methods make the protocol harder to defeat.

DOMINO relies on unauthenticated information and underestimates the abilities of an attacker. In particular, it fails to adequately consider the attacker using the detection and punishment mechanism itself to indirectly improve their performance. This is a symptom of the faulty distinction between a selfish attacker, those who merely aim to boost their performance at the expense of others, and a malicious attacker that aim to disrupt network service. This distinction overlooks an entire class of attack techniques were the attacker aims to improve their performance, but behaves maliciously in order to achieve it. Because DOMINO can not determine the source of message jamming, the mechanism can not reliably attribute misbehavior to the offending node. This fundamental flaw is central attack techniques presented in this paper.

IV. Related Work

The inability of DOMINO to reliably attribute jamming to the attacking node is investigated in [10]. The author notes that while the protocol protects against manipulations of interframe spaces and backoff counters, it can still be defeated by forging corrupted frames instead. The technique presented in the paper could be avoided by requiring that each node have it's own encryption key and that the header is also encrypted or signed. It could also be defeated with an analysis of the time frame over which the corrupted frames occur. The process we present for forging corrupted frames is similar to [10], but we do so in a way that prevents detection. Additionally, the discussion of the vulnerability is limited and was only proposed in theory, as testing it was outside the scope of the research. We build upon this prior research by focusing on defeating DOMINO and proving the efficacy of our methods using real world testing.

DOMINO also overlooks major flaws in the 802.11 protocol (specifically the 802.11i-2004 protocol), namely, the weak binding between WiFi Protected Access (WPA) Enterprise wireless network SSIDs and server authentication certificates[1]. [1] investigated this issue by attacking wireless networks in a similar fashion to the previously mentioned study. The authors point out that not a single victim was able to detect any attack on the wireless network due to this flaw. Simply spoofing MAC and IP addresses allows an attacker to avoid detection with ease. The study goes on to conclude that stealthy jamming of a wireless network, even one running WPA Enterprise, is entirely feasible.

Another paper addresses the issue of greedy behavior in Wireless Sensor Networks (WSNs)[4]. While this paper may refer to issues in non-slotted 802.15.4, the principles behind the greedy behavior and passive detection methods employed are extremely similar to those in the DOMINO protocol and apply very well to our research. This paper also fails to consider the methods of attack we will outline later and is vulnerable to every one of them. The protocol is far too trusting of the other nodes on the network and too passive against stealthy attacks.

While the previously presented works address the issues with DOMINO, [2] gives a possible solution to the MAC spoofing issues that are prevalent in wireless networks. The paper recommends a fingerprinting technique for detecting MAC spoofing which could make DOMINO a far more applicable solution to the greedy node problem. The paper also states that fingerprinting is entirely possible in a passive approach, without the need for specialized hardware or extensive modification to existing software architectures.

Some previous work exists on the greedy node problem, but we acknowledge that it is very limited in quantity and the available work itself may be both limited in scope and in applicability to our research. We have found, however, a few high quality sources of previous work from which to study and take into account when devising our solutions to the problem.

V. Three Approaches for Defeating DOMINO

False flagging, framing, and virtual stations are designed with two outcomes in mind. A) Introducing "noise" from the perspective of a defensive protocol equipped access point and B) dissociating the node's behavior from itself by attributing it's behavior to other nodes, be they real or virtual. The noise effect defeats heuristics by degrading the accuracy of their statistical methods to a point where their false positive rate is too high to be used in a real world deployment. While this could be countered with better analysis on the part of the heuristic, it is infeasible to perform such complex operations on the restricted hardware resources of wireless access points. The second effect, exploits a fundamental flaw in existing solutions. No matter how advanced a heuristic may be or unaffected a non-heuristic method is to noise, it will be unable to accurately identify malicious nodes if it can not reliably attribute the misbehavior to the correct *physical* node. This is achieved by exploiting the fact that the IEEE 802.11 MAC layer has no mechanism for fully authenticating a physical device.

A. False Flagging

Internet traffic is inherently bursty in nature. An intelligent attacker will sparingly jam traffic, only doing so when it is advantageous to them. By keeping track of each nodes approximate backoff counter, an attacker can selectively jam packets when the backoff timers of competing nodes are low. By doing so, and potentially even disregarding their own backoff timer, the node will ensure that competing devices have larger contention windows and thus will achieve higher bandwidth allocation. By the same logic, the node will allow traffic through if backoff timers have already been elevated and their is nothing in its send queue. This sort of behavior, while more intelligent than jamming any and all traffic, leaves artifacts that can be tracked. As explained in [6], while transmission rates themselves can not be reliably used to detect cheating the rate of collisions and average delays can be. At the cost of some power (for transmitting additional wireless signals) a node can opportunistically leverage idle time by issuing an RTS without any actual data to send then jamming the CTS meant for it. While this can lead to a decline in performance when utilization is higher (e.g. a large file transfer) the effect is quite small and still results in a large net gain in performance when compared to not cheating. This creates noise for the heuristic and increases the difficultly in identifying a cheating node.

B. Framing

Framing allows the attacker to perform two useful functions. It can be used as an intelligence gathering mechanism or as an offensive tool to gain bandwidth. Our method for framing consists of choosing a node on the local network at random, spoofing our MAC IP addresses to mirror those of the node we are framing, and then aggressively jamming every other node on the network.

This random victim selection and jamming effectively increases the error rate of the victim node high above that of the other nodes, triggering one of DOMINO's passive traffic analysis functions. It will also trigger the backoff functions as we will not allow the victim node to backoff between transmissions.

After one of DOMINO's functions is triggered, the victim node will be subject to penalization. Once this penalization has occurred, in the form of significantly lowered data rate or complete removal from the network, another random node is selected as our next victim. This process of node selection and jamming continues until there are no more nodes on the network to jam and we have secured all the bandwidth on the local network, or we are satisfied with the massive decrease in total traffic on the network.

Framing another node for MAC layer misbehavior has proven to be an effective method of indirectly gaining more bandwidth as nodes are progressively banned from the network following their framed jamming periods.

C. Virtual Stations

Virtual stations are a far more elegant attack technique that evades detection entirely by establishing several simultaneous connections to the access point using spoofed MAC addresses. These spoofed nodes can be used for multiple purposes:

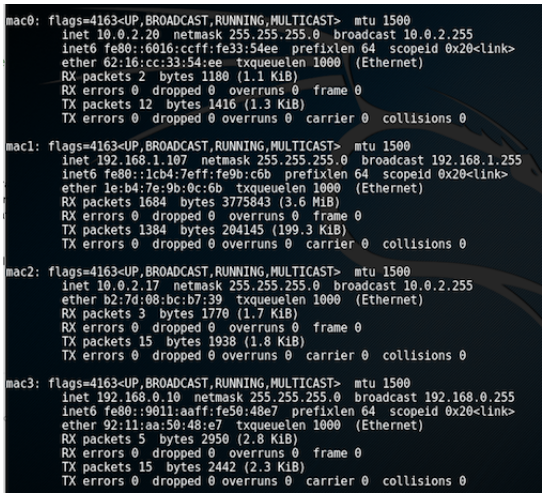
1) Virtual nodes may be used to normalize the effect of a misbehaving node. Increasing the number of nodes on the network will absolutely increase the number of non-forced collisions, making a misbehaving node appear relatively moderate in comparison to the other nodes on the network. As long as it does not deviate from DOMINO's "standard" for this network's behavior, it will not be penalized.
2) Virtual nodes may be used to minimize the impact of DOMINO ejecting a misbehaving node. If multiple spoofed nodes are controlled, having one thrown out will not deter the tech savvy attacker at all as more nodes with newly spoofed MAC addresses can be generated at any time.
3) Virtual nodes may be used to load balance the traffic received by each node. This method completely circumvents DOMINO as it does not engage in any method of jamming or MAC layer misbehavior. By load balancing traffic across all virtual nodes using the Round Robin[1] scheduling algorithm, an unfair amount of bandwidth will be gained by the attacker.

All of the previous uses listed for virtual stations will result in a gain of unusually high amounts of bandwidth for a "single" node on the network. Even if DOMINO discovers and ejects one of the cheating nodes, others can be generated and the unfair share of bandwidth still remains allocated to the attacker.

VI. Prototype Development and Testing

We developed a prototype for our malicious node by modifying open source drivers created by M. Vanhoef and F. Piessens. Their original code[2] as well as our modified implementation[3] are publicly available for download (some portions are redacted to prevent abuse, but are available to researchers upon request).

Virtual stations can be generated via simple shell commands in Linux. Generating a newly spoofed MAC address and then running DHCP to configure a valid local IP address is trivial (a static IP can also be configured). Connecting all the virtual nodes to the network is done by simply turning on the newly created wireless interfaces. Traffic load can then be distributed across these nodes using a round robin scheduling algorithm, or, for a different attack method, one of the nodes can be set to reactively jam and gain excessive amounts of bandwidth for itself.



Multiple wireless interfaces are shown above with spoofed mac addresses. mac2 and mac0 have used Dynamic Host Configuration Protocol (DHCP)[4] to configure their local IP addresses while mac1 and mac3 have been given static IP addresses. Kali Linux (32bit Debian) was the Linux distribution used for this test.

Reactive jamming was accomplished by editing the open source firmware of a TP-Link wireless adapter (our implementation was again, based off of M. Vanhoef and F. Piessens work). The modifications were then compiled into a new version of the firmware and loaded onto the device. Our test network consisted of the following four different devices:

1) MacBook Pro running MacOS and a VM running Ubuntu
2) MacBook Pro running MacOS
3) Dell XPS running Windows 10 and a VM running XUbuntu (used for loading firmware and jamming)
4) Netgear router

We collected our data via WireShark for both standard network behavior without the presence of jamming and network traffic under reactive jamming (specifically jamming data frames) using our modified firmware. Ordinary network behavior is shown below as a screenshot from our WireShark monitoring.

[1]Round Robin is a scheduling algorithm in which each process or function is given an equal sized time slice (quantum) and the functions are then executed in a circular fashion until complete. This method is effectively starvation free.

[2]https://github.com/vanhoefm/modwifi

[3]https://github.com/Ranind/modwifi

[4]Dynamic Host Configuration Protocol (DHCP) automatically provides an IP host with its IP address and other configuration details such as the subnet mask and default gateway.

```
Reply from 192.168.1.3: bytes=32 time=1ms TTL=64
Reply from 192.168.1.3: bytes=32 time=10ms TTL=64
Reply from 192.168.1.3: bytes=32 time=3ms TTL=64
Reply from 192.168.1.3: bytes=32 time=2ms TTL=64
Reply from 192.168.1.3: bytes=32 time=2ms TTL=64
Reply from 192.168.1.3: bytes=32 time=1ms TTL=64
Reply from 192.168.1.3: bytes=32 time=4ms TTL=64
Reply from 192.168.1.3: bytes=32 time=3ms TTL=64
Reply from 192.168.1.3: bytes=32 time=109ms TTL=64
Reply from 192.168.1.3: bytes=32 time=2ms TTL=64
Reply from 192.168.1.3: bytes=32 time=2ms TTL=64
Reply from 192.168.1.3: bytes=32 time=2ms TTL=64
```

Although collisions are ordinary in any wireless network, before mounting our attack, the network clearly appears to be devoid of issues and functioning normally. Global Internet access was possible on all devices connected to the test network. Our capture filter for WireShark explicitly filtered out all control frames as they are not applicable or useful in this context. Both MacBooks were set to repeatedly ping each other in order to generate some additional network traffic. The pings were delivered successfully (highlighted in pink in the wireshark output) and had low latency, single digit range in milliseconds.

After jamming the data frames of all other nodes, ordinary network behavior appeared to be much different. Collisions and error rate of the attacker increased drastically, but its pings continued to be delivered in an ordinary amount of time. Pings from other nodes, however, were either unable to be delivered or had multiple thousand millisecond delays. The error rate of the victim was extremely low, while that of the attacker was extremely high, despite his actual ability to use the network.

```
64 bytes from 192.168.1.4: icmp_seq=1408 ttl=64 time=4809.675 ms
64 bytes from 192.168.1.4: icmp_seq=1409 ttl=64 time=4325.435 ms
Request timeout for icmp_seq 1414
64 bytes from 192.168.1.4: icmp_seq=1410 ttl=64 time=5601.917 ms
64 bytes from 192.168.1.4: icmp_seq=1410 ttl=64 time=5602.188 ms (DUP!)
Request timeout for icmp_seq 1416
Request timeout for icmp_seq 1417
64 bytes from 192.168.1.4: icmp_seq=1411 ttl=64 time=7233.831 ms
64 bytes from 192.168.1.4: icmp_seq=1411 ttl=64 time=7236.177 ms (DUP!)
Request timeout for icmp_seq 1419
Request timeout for icmp_seq 1420
64 bytes from 192.168.1.4: icmp_seq=1412 ttl=64 time=9956.165 ms
Request timeout for icmp_seq 1422
64 bytes from 192.168.1.4: icmp_seq=1413 ttl=64 time=10773.371 ms
Request timeout for icmp_seq 1424
64 bytes from 192.168.1.4: icmp_seq=1414 ttl=64 time=11870.256 ms
64 bytes from 192.168.1.4: icmp_seq=1415 ttl=64 time=11293.066 ms
Request timeout for icmp_seq 1427
64 bytes from 192.168.1.4: icmp_seq=1416 ttl=64 time=12763.530 ms
Request timeout for icmp_seq 1429
64 bytes from 192.168.1.4: icmp_seq=1417 ttl=64 time=13258.061 ms
64 bytes from 192.168.1.4: icmp_seq=1418 ttl=64 time=13101.294 ms
64 bytes from 192.168.1.4: icmp_seq=1419 ttl=64 time=12939.516 ms
64 bytes from 192.168.1.4: icmp_seq=1420 ttl=64 time=12849.651 ms
64 bytes from 192.168.1.4: icmp_seq=1420 ttl=64 time=12853.852 ms (DUP!)
64 bytes from 192.168.1.4: icmp_seq=1421 ttl=64 time=12945.744 ms
64 bytes from 192.168.1.4: icmp_seq=1421 ttl=64 time=12950.630 ms (DUP!)
```

This extreme increase in data errors caused massive ping delays, latency increased from single digits to well thousands of milliseconds and even made numerous pings fail entirely to deliver. The global Internet was unreachable from the jammed MacBooks, all attempted page lookups failed to load. This full jamming attack then, is equivalent to a Denial of Service (DOS)[5] attack on all nodes of the network as it effectively prevents any data frame traffic from moving between the devices and the AP.

[5]A Denial Of Service (DOS) attack seeks to make network resources unavailable to legitimate users of said network.

VII. Effectiveness of New Misbehavior Techniques

Our implementation was able to obtain substantial real world performance gains at the expense of other nodes in both unprotected and protected networks. The most effective method of misbehavior differs depending the form of MAC layer misbehavior protection, if any, employed by the AP of the victim network.

By jamming only the data frames of nodes whose MAC address does not match our jamming host device or the AP, we can gain unfair amounts of bandwidth by forcing the backoff times of other nodes to be artificially high through reactive jamming. In a network not employing any form of MAC layer cheating protection (such as DOMINO), simply jamming all data frames not belonging to our device or the AP is sufficient as there is no penalty for cheating in this way.

Should DOMINO be present, however, reactively jamming only when necessary and additionally sending scrambled frames when we are not actively transmitting or receiving will raise our error rate and backoff times to meet DOMINO's standard network behavior model. This False Flagging attack effectively allows our device to gain unfair amounts of bandwidth when desired without risk of detection as it will not deviate from DOMINO's aforementioned behavior modeling in any way significant enough to call for penalizing action.

Framing another node for misbehavior is as simple as spoofing its MAC address before initiating a full jamming attack on all other nodes. This attack raises its error rate far above the standard rate defined by DOMINO and will be flagged and penalized quickly. Once all other nodes have been penalized and/or removed from the network, the malicious node is free to take full advantage of the channel bandwidth gained. As the figure below shows, the attacker succeeded in appearing as a victim by using false flagging to generate fake corrupted frames. The victim of framing was excluded from the reactive jamming, so it's error rate remains extremely low. As a result of the statistically significant difference between the error rates, the victim is classified as misbehaving and is ejected from the network.

Error Rates (Low Implies Cheating)				
	Access Point	Victim	Cheater	Bystander
Error Rate	0.9%	1.09%	87.66%	88.32%
# Frames	21590	1564	3121	75040

Note: Access Point will never be classified as misbehavior

The virtual stations approach entirely eliminates any possibility of detection from DOMINO if no jamming is used. Simply combining the bandwidth allocated to each node causes a massive increase in total bandwidth given to the owner of all the virtual nodes on the network.

VIII. Potential Methods for Detecting Forged Frame Collisions

A combination of extended history tracking with hardware and software authentication mechanisms have the potential to mitigate the concerns posed by these new misbehavior techniques in a low cost package that is suitable for real world deployment.

Virtual station based attacks can be mitigated by using multiple access points to triangulate the approximate location of nodes on the network. If too many nodes are in too small of a predicted space, they are flagged as suspicious and their activity tracked closely. Their IP addresses will also, most likely, have been configured by DHCP at approximately the same time. If an excessively large number of nodes were suddenly instantiated in too small a space, with IP addresses that were all configured at approximately the same time, those nodes would be flagged for suspicious activity.

Framing attacks can be defeated simply by employing some form of authentication between the AP and the node. A challenge-response or Public Key Infrastructure (PKI)[6] of some sort will prevent a node that is spoofing the MAC address of another node that already exists on the network from gaining access. The AP can simply refuse connections to inauthentic requests from a spoofed MAC address that it knows already exists on the local network.

Device type detection can also be used in order to prevent MAC address spoofing. If a table of current devices on the network is saved by the AP containing the MAC address, device type, as well as a unique device information string such as the string returned when querying the Win32_BIOS class in System.Management on a Windows machine, which has an extremely high chance of being globally unique.

False flagging attacks can be mitigated with RTS/CTS scrambling identification and close tracking of nodes transmitting these malformed frames. Discovering intentional self jamming attacks by reading the data within the frames to determine when the jamming is beneficial versus when it is simply a false flag, leading DOMINO to believe that its overall average behavior does not deviate too greatly from the defined standard expectation of nodal behavior on the local network.

IX. Conclusions

DOMINO is a useful protocol as it prevents large scale cheating from the logical cheating perspective. It fails to consider, however, MAC layer misbehavior in an illogical fashion (intentionally raising its own backoff and error rate in a negative way) in order to defeat the protection granted by the protocol. As DOMINO does not authenticate clients by anything more than a MAC address, these MAC addresses can be spoofed and other nodes can be framed for cheating and ejected from the network. Multiple virtual stations may also be employed to completely circumvent DOMINO and avoid detection entirely. These stations can also be used as scapegoats, with each one being banned from the network for cheating, but others taking its place soon thereafter. The device itself will still benefit from the massive increase in bandwidth as more virtual stations may be generated with ease.

DOMINO is flawed but can be fixed. Solutions and possible defences against each attack included in this paper are feasible without large hardware or software modifications in most cases. False flagging can be prevented with an increased level and depth of passive traffic analysis by discovering intentional scrambled frame transmissions and backoff times rather than observing only a standard deviation with respect to the expected behavior of the node. Framing can be prevented with device type detection and near globally unique identifiers passed by each device on the network in addition to a MAC address to prevent MAC spoofing attacks. Authentication can also be achieved with a PKI between the AP and device on the network. A secret key can be generated based on device specific values that will prevent MAC spoofing as any device that cannot properly identify itself will be removed from the network. The virtual stations attack will, unfortunately, require additional hardware in the form of multiple APs. These will be used to triangulate each nodes approximate physical position within the network's domain. If an excessive number of nodes appear in an area far too small to host that number of machines, they will be properly grouped and penalized.

[6] A set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption.

References

[1] Aldo Cassola, William Robertson, E. K. G. N. A practical, targeted, and stealthy attack against wpa enterprise authentication. *Northeastern University College of Computer and Information Science* (2013).

[2] Idland, C., Jelle, T., and Mjølsnes, S. F. *Detection of Masqueraded Wireless Access Using 802.11 MAC Layer Fingerprints*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 283–301.

[3] Masayuki Ariyoshi, William Bartley, T. B. C. W. D. J.-P. F. A. G. P. H. R. H. H. J. K. R. M. G. J. G. J. H. J. L. K. D. J. L. T. L. H. L. O. L. T. O. G. R. J. W. R. S. S. M. S. C. S. P. W. H. W. D. W. Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE* (2012).

[4] Mokdad, L., Abdelli, A., and Ben-Othman, J. Detection of greedy behavior in wsn using ieee 802.15 protocol. In *Proceedings of the 2014 IEEE 22Nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems* (Washington, DC, USA, 2014), MASCOTS '14, IEEE Computer Society, pp. 106–111.

[5] Pelechrinis, K., Iliofotou, M., and Krishnamurthy, S. V. Denial of service attacks in wireless networks: The case of jammers. *IEEE Communications Surveys Tutorials 13*, 2 (Second 2011), 245–257.

[6] Raya, M., Aad, I., Hubaux, J. P., and Fawal, A. E. Domino: Detecting mac layer greedy behavior in ieee 802.11 hotspots. *IEEE Transactions on Mobile Computing 5*, 12 (Dec 2006), 1691–1705.

[7] Serrano, P., Banchs, A., Targon, V., and Kukielka, J. F. Detecting selfish configurations in 802.11 wlans. *IEEE Communications Letters 14*, 2 (February 2010), 142–144.

[8] Tang, J., Cheng, Y., and Zhuang, W. Real-time misbehavior detection in ieee 802.11-based wireless networks: An analytical approach. *IEEE Transactions on Mobile Computing 13*, 1 (Jan 2014), 146–158.

[9] Toledo, A. L., and Wang, X. Robust detection of selfish misbehavior in wireless networks. *IEEE Journal on Selected Areas in Communications 25*, 6 (August 2007), 1124–1134.

[10] Vanhoef, M., and Piessens, F. Advanced wi-fi attacks using commodity hardware. *Proceedings of the 30th Annual Computer Security Applications Conference* (2014), 256265.