Arpeggio: a Penetration Attack on Glossy Networks

Zhitao He
SICS Swedish ICT
Kista, Sweden
Email: zhitao@sics.se

Kasun Hewage
Uppsala University
Uppsala, Sweden
Email: kasun.ch@gmail.com

Thiemo Voigt
SICS Swedish ICT and Uppsala University
Sweden
Email: thiemo@sics.se

Abstract—Glossy networks make use of concurrent transmissions to achieve rapid network flooding in wireless networks with high reliability. They are robust against jamming and header injection attacks. We find that Glossy floods can be hijacked by a packet injection attacker to penetrate into the network and cause severe loss. We demonstrate the design of such an attacker by evaluating its effectiveness in a 30-node testbed.

I. INTRODUCTION

Lending to constructive interference and capture effect [1] in radio transmissions, the Glossy flooding service has proven to be a rapid and reliable way to disseminate information among tens of low power sensor nodes [2]. Compared with multihop routing, flooding provides an illusion of a flattened topology by concealing the network’s internal structure, thus greatly simplifying usage.

Is a Glossy network truly *flat* and *hardened*, so that it is robust against penetration?

We have recently shown that Glossy remains intact despite attempts from jamming and PHY-layer header injection attacks [3]. It suffers noticeable losses only when the attacker turns into an *insider*, i.e., runs on the Glossy protocol itself, so that bogus packets can be injected at particular time points relative to the communication slots arranged by Glossy’s precisely synchronized global schedule. In order to find an effective attack with minimal protocol dependence, we first analyze the unbuffered radio reception process used by Glossy, which plays an important role to filter out intrusive packets. We then identify the wake-up periods preceding each Glossy flooding to offer a precious time window for an intruder to inject packets into the network. The bogus frames are sent in the CC2420 radio’s cyclic transmission mode, previously exploited by us to successfully disrupt low-duty cycle, random access MAC protocols [4].

Our penetration tests lead to a surprising discovery. Glossy’s anonymous flooding service, underpinned by concurrent radio transmissions, turns out to be a double-edge sword. On one hand, the concurrent transmissions provide abundant link redundancy, often sufficient to swarm out foreign packets from a single malicious attacker. On the other hand, an attacker capable of disguising bogus frames as legitimate ones is given access to the whole network by free-riding on Glossy floods. The resultant damage of such hijacked floods turns out to be much wider in range than a full duty-cycle jammer. We call our attack *Arpeggio*, since a bogus frame ripples across the network hop-by-hop, in a similar fashion to that of a

harpist playing an ascending sequence of notes by sweeping her fingers across the harp strings. Our evaluation shows that, using a minimal level of TX power, an attacker located at the fringe of a 30-node Glossy network can cause over 50% packet loss to all nodes except a few that are in the immediate transmission range of the Glossy initiator. Three attackers can together bring down the aggregate PRR to 30%. This raises a security alarm, because the attack poses an immediate threat to all Glossy-based systems. We also show that the attacker’s relative hop distance to a legitimate flood initiator is a decisive factor on whether the attack makes a shallow dent or a deep breach. By exposing the internal structure of a Glossy network previously treated as flat, Arpeggio provides useful guidance for developers to harden specific instances of such a network.

Our main contributions are:

- Identification of the precise mechanism that bolster Glossy’s robustness against existing header injection attacks.
- Design of a packet injection attack for an unsynchronized, low power radio transmitter, that exploits Glossy floods to penetrate the network.
- Evaluation of the efficacy of the attack under various configurations.

The rest of the paper is organized as follows. Sec. II provides a background on Glossy and the Low Power Wireless bus that provides an link layer abstraction to Glossy; we then analyze the strengths and weaknesses of Glossy in Sec. III, based on which we design and implement our attack described in Sec. IV and Sec. V; we evaluate the attack in Sec. VI; after a summary of related work (VII), we conclude by Sec. VIII.

II. BACKGROUND

We first describe how the Low-Power Wireless Bus uses Glossy flooding to provide a TDMA MAC service to the upper layers. We then describe the Drizzle PHY header injection attack, whose radio transmission mechanism forms the basis of our packet injection attack.

A. Low-Power Wireless Bus on Glossy

Glossy’s high efficiency and reliability has been testified by its usage as the main MAC layer communication primitive by several novel data dissemination protocols [5]–[8]. Glossy has also been used to transport general traffic patterns, with the help of a link layer abstraction called the low power wireless bus (LWB), which provides a convenient API for unicast

Fig. 1. Composition of slots in a LWB round. Each round begins with a slot for the schedule. There can be several data slots and few contention slots.

and broadcast together with a TDMA scheme to coordinate channel access [9].

LWB uses scheduled slots to control network-wide Glossy floods. Glossy floods use the fact that simultaneous transmissions of the same packet interfere constructively at a receiver to achieve very fast and reliable network floods. Nodes in Glossy retransmit every data packet several times to further increase robustness.

In LWB, time is split into communication rounds, which in turn are divided in slots. Every slot consists of a number of repeated floods. The schedule of slots for each communication round is prepared by a dedicated controller node called *host*. Figure 1 depicts the composition of slots in an LWB round. Each round begins with a slot for the schedule that specifies the round period, the time at the *host*, data slots and contention slots.

Nodes use data slots for sending/receiving application data. The assignment of a data slot to a node indicates that the node becomes the initiator of a Glossy flood while other nodes become receivers of the Glossy flood. In this way, all other nodes receive what the initiator node has sent and then can filter the data locally based on interest.

A schedule may contain contention slots during which any node can send its demand for slots to the *host*. In contention slots, packet collisions are possible. However, one packet typically reaches the *host* due to the capture effect [1].

At the end of each round, the *host* computes a new schedule for the next round based on the demand of the nodes. The slot allocation for a schedule is dictated by the scheduling policy at the *host* and can be based on factors such as throughput, latency and energy efficiency.

B. Drizzle header injection attack

Symbol synchronization is required for a radio receiver to correctly decode a frame. This is achieved by the receiver locking onto a SYN header detected at the beginning of a transmitted frame. Once synchronized to the symbol boundary implied by the header, the receiver proceeds to decode the subsequent frame length header, and continues to decode and store an N-byte payload into a hardware frame buffer. An 802.15.4 receiver usually automates this process for optimal performance, exposing itself for a DoS attack that we show in a previous work [4]. Comprising only a three-byte bogus header dubbed *Droplet*, a bogus frame can trigger the receiver's frame decoding/buffering process. When transmitted in the CC2420 radio's cyclic transmission mode, the droplets become a *drizzle* that repeatedly lock the receiver into futile work. The Drizzle

Fig. 2. The Drizzle attack. A receiver wakes up to an ever-present stream of bogus headers, thus deprived of the chance to receive legitimate data frames.

attack, depicted in Fig. 2, has been shown to hit hard against the low-duty cycle, carrier-sensing ContikiMAC. The strength of Drizzle lies in its compact organization of short bogus headers (spaced at $96\mu s$ intervals), making its target very susceptible to mistakenly synchronizing to one of them upon wake-up.

III. ANALYSIS

We analyze the reasons why Glossy appears very robust against interference, despite its use of the low-cost 802.15.4 radio hardware.

A. Glossy's link redundancy

Glossy's finely time-synchronized nodes collaborate in flooding the whole network by retransmitting the original message from a source node several times within a tight time window of just a few milliseconds. Since all nodes participate in the retransmissions, the probability that a copy of the message reach the destination is greatly enhanced, even if some individual links are weak. Occasional interference from coexisting radio transmitters causes negligible harm, because most of these transmissions occur outside of the narrow time windows when nodes wake up to communicate with each other according to a global schedule. When interference falls inside a wake-up window, Glossy flooding is more resilient than conventional ARQ-based unicasts, thanks to the greater link redundancy provided by all the nodes synchronously retransmitting the same packet multiple times. Moreover, Glossy's highly customized radio driver mitigates the adverse effect caused by foreign packets, as we show next.

B. Radio receptions: fully buffered, partially buffered, and unbuffered

The usual procedure for packet reception in low power radios involves usage of a hardware frame buffer in the radio. The radio decodes and buffers a whole packet in its RX FIFO; it generates an interrupt to wake up the MCU; the latter then reads the buffered packet via the SPI bus into its RAM for frame decoding and other upper layer protocol processing. Because this procedure is entirely sequential, it is safe and simple to implement; benefiting from deep sleeping modes of modern MCU's, it also minimizes power consumption.

An alternative approach that reduces latency and increases throughput allows an incoming packet to be only partially

buffered. By programming the radio to wake up the MCU as soon as a portion of the packet is decoded, a two-stage pipeline is created to parallelize radio reception and MCU bus read. Due to data rate difference between the air interface and the SPI bus (the latter being higher), however, the pipeline would stall if the radio's RX FIFO underflows. Optimizing the pipeline throughput based on an estimated packet length and handling occasional underflows add considerable software complexity as well as a degree of performance variation to the device driver. The MCU's duty cycle also increases when the pipeline stalls, leading to energy waste.

To achieve extremely low latency, Glossy's rapid network-wide flooding design requires a minimal transition time between a packet's reception and retransmission. Moreover, the protocol's reliance on concurrent radio transmissions imposes a very small tolerance in software processing latency, equivalent to just a fourth of a symbol period ($0.5\mu s$), during packet reception. Therefore, the Glossy radio driver adopts an even more aggressive approach than the partial buffer method in reading the radio's RX FIFO buffer. The radio is programmed to raise an interrupt as soon as the 802.15.4 PHY-layer header is received, so that the MCU starts to poll a hardware pin to read subsequent bytes one after one as soon as they arrive. This results in a sequence of single-byte SPI read accesses. Consequently, a higher overhead than a normal multiple-byte access is incurred, mainly by the repeated transmissions of the same SPI address to retrieve each data byte. Because the MCU is occupied throughout the whole reception process, which can last over $4ms$ for large packets, extra energy is consumed and other pending tasks are locked out temporarily.

Fig. 3 illustrates the three modes of radio reception.

Fig. 3. Three modes of radio reception: fully buffered, partially buffered, and unbuffered. Glossy's unbuffered reading minimizes software delay, at the cost of extra CPU overhead.

Unbuffered reception is rewarded by minimal packet retransmission latency. Immediately after the last byte is received, a relay counter field is updated, then the whole packet is retransmitted.¹

C. Glossy's robustness against coexisting transmitters

Glossy's unbuffered radio reception includes header checking that can result in premature termination of the process. The first two received bytes, namely the payload length byte

and Glossy application header byte, are decoded in situ so that the radio can terminate an ongoing frame reception. Foreign packets with a mismatching size or missing a proper app header are thus canceled early during decoding. Without header checking, any received packets would be retransmitted, which would interfere destructively with the ongoing flooding. The result is a certain degree of robustness against random interference.

We previously noted that injection of 802.15.4 PHY headers by the Drizzle attack has negligible adverse effects on Glossy [3]. In light of the new analysis, we suspect the main reason is the header checking mechanism that terminates radio reception of foreign packets. To verify our conjecture, we launch the Drizzle attack at the fringe of a 30-node Glossy network, and collect statistics of both packet losses and header/checksum errors.² The network consists of a source node, a host (scheduler) node, and 28 receiver nodes; they transmit at the maximum $10dBm$ power level; The attacker transmits at $-25dBm$. 100 packets are sent in each test run. With the maximum retransmission counter set to three, a Glossy flood creates a total number of approximately $100 \times 30 \times 3 = 9000$ transmissions. Over 8 runs, the attacker ramps up its attack duration, from 0 % to 100 % of the run period. Fig. 4 shows our Drizzle attack is futile, causing very few packet losses. However, one of the receiver nodes located close to the attacker records a large amount of header errors (Fig. 4(b)). But thanks to Glossy's early termination of foreign packets, that receiver has received all legitimate packets. On the other hand, the barrage of bogus headers is completely unnoticed by the other nodes due to limitation of the attacker's transmission power (Fig. 4(c)).

D. Glossy fends off bogus application header injection attacks

How about injection of legitimate Glossy application headers? They should be able to bypass the simple header checking. Would a large amount of such payload-free headers destabilize the tight retransmission rounds of Glossy floods? We modify the Drizzle attack, extending the encapsulated "droplets" by an extra byte consisting of a legitimate Glossy application header. As the close-to-perfect PRR's in Fig. 5(a) indicates, this attack hardly dents the peripheral of the network, let alone penetrate it. We can observe however a remarkable shift of Node 202's error pattern, now characteristic of a proportional increase in CRC errors as the attack duration prolongs (5(b)). This shows that a significant number of bogus headers manage to pass the header checking. However, the data redundancy provided by the numerous Glossy retransmissions is sufficient to compensate the bandwidth loss by this single receiver with high CRC errors.

This unsuccessful attack becomes the stepping stone for our next attempt, which proves to be highly effective.

¹Glossy makes a further optimization to reduce software delay, letting a packet's last 8 bytes to automatically queue into the RX FIFO and raise an interrupt on completion.

²Detailed testbed set-up is available later in the Evaluation section

Fig. 4. Futile Drizzle attack against 30-node Glossy network. Attack duration increases from 0% to 100% over 8 runs, in 16% steps. Node 202 is exposed to an extraordinary amount of LEN header and Glossy header errors as the close-by attacker ramps up its attack duration, but still retains a 100% PRR, thanks to early termination of such errors and abundant retransmissions from other nodes.

Fig. 5. Unsuccessful Glossy application header attack against a 30-node Glossy network. Attack duration increases from 0% to 100% over 8 runs, in 16% steps. Node 202 experiences an increasing number of CRC errors as the attacker prolongs its transmission, but still achieves a 99.6% PRR, thanks to abundant retransmissions from other nodes. The latter do not observe any anomalies, as they are located outside of the attacker's range.

IV. ATTACKER DESIGN

The insight obtained from the previous analysis points us towards further extension of the header inject attack in order to pass the CRC check. By forging bogus packets and injecting them at the fringe of a Glossy network, we hope to see them propagate across the whole network by surfing on the waves

of Glossy floods.

A. Attack model: a time window to jump onto a flood

The timeline of a Glossy flood depicted in Fig. 6 exposes a time window for our attacker to launch. Despite all nodes in a Glossy network wake up at the same time to participate in concurrent transmissions, they join the flood at different time slots, based on their hop distance from the initiator node. The idle wake-up time for immediate neighbors of the initiator is very small, consisting of a short guard time ($500 \mu s$) and a few tens of CPU cycles of software delay. However, receivers located at hop two need to wait until the initial packet is relayed by hop-one neighbors. This leaves open a full relay slot, of a few ms , for the attacker to inject a frame to hop two receivers after they wake up. Nodes located further down the flooding path have even longer idle listening time, proportional to their hop distance to the initiator. During the time these nodes wait for a flood from upstream, they are vulnerable to other packet sources.

Fig. 6. The Glossy flooding process wakes up all receivers at the same time, leaving receivers more than one hop away from the initiator a considerable idle listening time window. Its length is equivalent to one or multiple full packet time slots.

B. Bogus data packet

An LWB data packet consists of an 8-byte header: an 8-bit Glossy application header is followed by two 16-bit addresses, a 8-bit payload length, an 8-bit counter for packets in the queue, and finally an 8-bit data option. Without studying the source code in detail, we simply use a radio sniffer to collect a set of packet samples and craft a data packet that consists of a legitimate 8-bit header but no payload. We append a 16-bit CRC checksum to the frame, so that it can pass the CRC check on radio receivers. For easy distinction between our packet and packets sent by other nodes, we use a source address different from that of the authentic source node of our test application. The application accepts both at the receiver nodes, but only counts packets originated from the authentic source node as correct receptions in its statistics counter. The total length of the bogus frame, including an 802.15.4 header, becomes 13 bytes. It is therefore considerably longer than the 3-byte “Droplet” used in Drizzle, but still quite short compared with a normal data packet, which can be over 130 bytes including the preamble. Transmission of this bogus frame in tight packs,

using the cyclic TX mode of the CC2420 radio, is illustrated in Fig. 7.

Fig. 7. Injection of a bogus data frame to generate a Glossy flood. The frame is relayed over multiple hops to reach a range far outside of the attacker's radio transmission range.

Later we will show that this continuous stream of frames has a rather high likelihood to be detected by an awakened Glossy node and to be mistakenly retransmitted to other nodes. We do not assume the attacker has superior TX power: it is very likely to be overwhelmed by a concurrent Glossy transmitter. But if accepted by one or more Glossy receivers, the bogus frame is then retransmitted at the same TX power as legitimate frames, with greatly enhanced probability of being flooded across the whole network.

C. Bogus LWB schedule

One other important packet type underpins a Glossy-based LWB network: the schedule packets broadcast periodically from the host node. They carry global time information used by all nodes to schedule their data transmissions: host address, current time at host, number of data slots in current round, round period, number of contention slots. Nodes buffer the application data packets locally until they have data slots allocated in the schedule. As the buffer is limited in size, new application data packets are discarded when the buffer is full. It is simple for us to forge bogus schedule frames by modifying the content of a sniffed packet. In an attempt to make the network unstable, we send strings of 9-byte long synch packets that indicate zero available slots, in the *Arpeggio* manner. The frame format is shown in Fig 8.

Fig. 8. Injection of a bogus LWB schedule frame. This would reset the network-wide schedule compiled by all nodes.

Fig. 9. The timing configuration used for schedule, data and contention slots and schedule computation.

V. IMPLEMENTATION

In this section we present some details on the implementation that we use for the evaluation in the next section.

A. LWB application

To evaluate the effectiveness of the attacks, we implement a test application. The application runs on an LWB network that contains an LWB host, a source node, and multiple receiver nodes. The host node defines the schedule so that all possible data slots are allocated to the source node, i.e., the source node uses all data slots to send data packets in each round. Glossy implicitly enables all receiver nodes in the network to receive these packets. Each application packet carries a sequence number to identify the packets that are lost during an attack. We use serial data output and general purpose input/output pins of the Tmote Sky nodes for debugging and obtaining packet loss information to assess the spread of the attack. A typical test run consists of 100 data frames, each 28-bytes in size, sent from the source node at 125 ms intervals. We collect packet reception ratio and three error statistics (CRC, LEN header, and Glossy app header) after each run. The RSSI and LQI of each received frame are also recorded.

B. LWB timing

We use the Tmote Sky implementation of Glossy and LWB for Contiki OS in our experiments. We configure Glossy to retransmit each packet maximally three times during schedule, data and contention slots. In order to adhere to the timing constraints, we configure Glossy and LWB with the timing parameters shown in Figure 9. We set the duration of each LWB data slot to 30 ms which is sufficient for three re-transmissions of a 127 byte large packet. Glossy disables hardware interrupts that may interfere with Glossy's tight timing constraints. Therefore, we use a 4 ms gap between two slots in order to allow the execution of hardware interrupts used by Contiki's internals. Note that nodes turn off their radios during this time gap. Therefore, attacker's bogus packets during this time gap do not affect the packet reception in the subsequent slot.

C. Attacker

We implement the attacker on Z1 and Tmote Sky sensors. Both carry an MSP430 MCU and a CC2420 radio. We develop the software as an application running on the Contiki OS for

these platforms and modify the CC2420 radio driver to support the cyclic transmission mode needed for Arpeggio. When the radio transmits in this mode continuously, it is quite hard to track its actual status, as all its IO pins are stuck at the same level. We therefore rely on a separate sniffer node to capture the attacker's transmissions.

To be able to alter the attacker's multitude of parameters, such as the content of the attack frame at runtime, we develop a sophisticated set of serial debugging commands. For example, our program stores a static array of commonly used frames, ranging from the original "Droplet" header to LWB's schedule packet. These frames can be selected at runtime. Furthermore, we allow the user to key in frame snippets from the serial port, and then assemble a new frame by combining the snippets with a frame in the static array. Our usual software development workflow involves setting up a small 5-node Glossy network in our office and then launching a sole attacker and a sniffer to observe the effects, altering certain parameters along the way. When satisfied, we upload the code to the 31-node Flocklab testbed to run a full test, then collect the log files to perform some analysis.

Because the attacker program is unsynchronized with the test application, we need some means to automatically launch attack runs at approximately the same time when the application's source node starts data floods, so that repeatable results can be attained. We therefore initialize the attacker to be in a listening mode, only to switch to transmission mode when it detects one of the first five sequence numbers in a 100-packet test run. In order to derive the relationship between attack duration and packet loss, the attacker ramps up its attack duration in a number of linear steps after each run, from 0% to 100% of the total run time.

VI. EVALUATION

We conduct experiments in the FlockLab testbed [10] using 31 Tmote Sky sensor nodes. Figure 10 shows the topology of FlockLab. All experiments are conducted on channel 26, the one with lowest noise level.

A. Testbed Setup

We select node 201 to be our primary attacker, as it is located away from the main building and has the longest average distance to all other nodes. When transmitting at -25 dBm power, this attacker can only reach its closest neighbor, Node 202, located about 15m away. Node 33 hosts our global LWB scheduler; We use Node 4 as an LWB broadcast source to initiate all Glossy data floods. Both have rich connectivity with multiple neighbor nodes, and are safely outside of the attacker's transmission range. All Glossy nodes transmit at 10 dBm, the maximum level allowed for CC2420. We want to stress that the main objective of our test configurations is revealing insights of the operation of a compromised Glossy network from a few perspectives, rather than inflicting maximum packet loss or causing complete service disruption. Therefore, the attacker only launches radio transmissions after the network enters a stable state, when all nodes are

Fig. 10. The topology of the Sky nodes in FlockLab.

synchronized to the global schedule. A malicious attacker may, however, use the same method, albeit with a more powerful radio transmitter and more precise timing, in an attempt to force the network into perpetual bootstrapping.

B. Bogus Data Packet Attack

Over 8 successive test runs spaced at 30s intervals, the attacker transmits a sequence of bogus data packets in Arpeggio mode, stepping up the sequence length by two seconds per run. The attacker remains idle in the first run, so that we can use the resultant 100% PRR as our baseline for the subsequent runs with high confidence. In the last run, the attacker is switched on shortly after the first detected data packet and remains active until the last one is sent. The overall PRR aggregated from the 28 receiver nodes over the test runs are shown in Fig. 11. We can see that the average PRR drops successively from 100% to 56.9%. This is very impressive compared with our previous two futile attempts with Drizzle and Glossy app headers.

Investigation into the large variations among nodes leads us an interesting discovery, which we explain using the individual PRR's of three nodes in Fig. 12: Node 202, the attacker's close neighbor; Node 2, a close neighbor of the source node; and Node 7, a node at far east corner many hops away from both the attacker and source. Despite located over 50 away from each other, Node 202 and Node 7 display very similar loss patterns over the runs, whereas Node 2 is unaffected by the increasing intensity of the attack.

Fig. 11. Aggregate packet reception ratio (PRR) over 8 successively runs with increasingly longer attacks launched from Node 201. PRR values are averaged among all 28 receiver nodes. The two error bars per column indicate the standard deviation and maximum/minimum values across the nodes. The aggregate PRR decreases monotonically as the attack intensifies.

Zooming further into the last run in Fig. 13, we observe a striking semblance between Node 202 and Node 7 in the actual sequence numbers missing among their received packets, despite they probably receive those packets from two completely disjoint sets of Glossy neighbor nodes, implied from the lack of correlation between their RSSI and LQI values. This is a strong evidence that our bogus data packets successfully ride on some of the Glossy floods, hijacking the concurrent transmissions for propagate themselves across the network. A scan in the serial output log file confirms that the original bogus data packets, embedding a distinct source address, do reach the majority of receiver nodes.

(a) Node 202

(b) Node 7

Fig. 13. Packet losses in a single run of bogus data attack by Node 202 and Node 7. The two nodes miss many common packet seqno's, an indication that they are overwhelmed by the same Glossy floods contaminated by the attacker.

show the aggregate PRR's of these well-defended nodes, in comparison with the aggregate PRR's of the other severely compromised nodes in Fig. 14 and Fig. 15.

Fig. 12. Individual PRR over 8 successively runs by three receiver nodes located at very different locations relative to the attacker and source node.

Nevertheless, our evaluation also reveals a few receiver nodes, all located close to the source node, appear intact from infiltration. Theses close neighbors of the source node form a strong stronghold around it.

C. Multiple Attackers

We upgrade our bogus data attack by converting two more nodes, Node 3 and Node 17, into Arpeggio transmission. A repetition of the eight test runs show an elevated degree of damage. The average PRR drops below 30% in the last run, and some nodes suffer over 90% loss. However, the stronghold of five nodes around the source remains relatively intact. We

Fig. 14. We identify a set of five nodes located closely around the source node that repel the attackers.

Fig. 15. Reception at 21 receiver nodes are heavily compromised under the simultaneous attack of three Arpeggio nodes.

The distinct partition of the receiver nodes into these two groups proves the correctness of our attack model: one-hop neighbors of the initiator have a short wake-up window that is hard to capture by an outsider node, but other downstream nodes in the flood have a large likelihood to wake up to a bogus frame encapsulated in an infinite Arpeggio stream.

The scenario becomes entirely different if we attempt to attack the stronghold directly. We repeat the single node, bogus data attack by launching it from Node 1 instead. We observe only a very minor drop in PRR ($< 2\%$) across all nodes over the whole eight test runs. Our penetration attack thus unravels the dynamic internal structure of Glossy floods, implying that the strength of its defense might shift around in unpredicted fashions, as nodes rotate their roles between initiator and receiver.

D. Bogus LWB Schedule Attack

We launch the second variant of Arpeggio, embedding fake schedule information in bogus LWB packets. Fig. 16 shows that we again achieve a high degree of infiltration to the network, reducing the average PRR to 48.4% in the last run. The amount of header errors and CRC errors detected by the nodes are not significantly higher than those in normal Glossy floods, as shown in Figure 17.

Fig. 16. Aggregate packet reception ratio (PRR) over 8 successive runs with increasingly longer attacks launched from Node 3.

Fig. 17. Error statistics collected from the LWB schedule injection attack

VII. RELATED WORK

We have shown that Glossy is very robust against several attacks on its availability [3]. Besides being robust against attempts to break Glossy's constructive interference by not obeying Glossy's timing constraints, they have also shown that Glossy is robust against jamming and the Drizzle attack. Motivated by their results, we investigate deeper why Glossy is robust against the Drizzle attack and modify it to be more effective.

There are also other attacks against low-power wireless networks. Wood et al. present several denial-of-service attacks on wireless sensor networks and ways to map the jammed region [11]. Along the same lines, Xu et al. present several attacks on wireless networks and complementary mechanisms to detect them [12].

Yang et al. have presented LearJam, an energy-efficient attack against duty-cycled networks [13]. In LearJam, the attackers learn the transmission period and then jam the wireless channel when nodes are transmitting. This way, the attackers save energy. Similar attacks on LWB would be possible if the host reuses the same schedule over many rounds. The authors also present ways of mitigating the jamming attack by rescheduling transmission patterns. Tiloca et al. have presented JAMMY [14] that varies the schedule in TDMA-based sensor networks. Their approach is decentralized in that each node computes the slot allocation of the next superframe in a distributed and autonomous fashion.

Other energy-efficient attacks have been presented by Li et al. [15] who discuss optimal attack and network defense strategies for single-channel wireless sensor networks. Law et al. demonstrate attacks against several MAC layers showing that it takes little effort to implement effective jammers even without detailed knowledge of the MAC protocols [16]. Wilhelm et al. present reactive jamming that monitors the wireless channel and jams only when the channel is busy [17]. EDJam tries to avoid energy consumption for channel monitoring by getting knowledge about transmission periods in advance [18]. While these attacks are energy-efficient, they would likely have the same effect as other jamming attacks on Glossy in that the effect would be constrained to the local area around the jammer as shown in our jamming attacks against Glossy [3].

VIII. CONCLUSIONS

We successfully demonstrate a packet injection attack that can penetrate deeply into Glossy networks, by free-riding on the concurrent transmission mechanism of Glossy floods. We achieve a reduction in packet reception ratio much greater than previous attempts using jamming or header injection attacks. Our results reveal that Glossy networks' perceived robustness against malicious attacks need re-examination, and we encourage network developers to bolster their defenses.

ACKNOWLEDGMENTS

This work has been supported by Sweden's innovation agency VINNOVA.

REFERENCES

- [1] K. Leentvaar and J. Flint. The Capture Effect in FM Receivers. *IEEE Transactions on Communications*, 24(5), 1976.
- [2] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with Glossy. In *Information Processing in Sensor Networks (IPSN)*, 2011.
- [3] Kasun Hewage, Shahid Raza, and Thiemo Voigt. An experimental study of attacks on the availability of glossy. *Computers & Electrical Engineering*, 41:115–125, 2015.
- [4] Zhitao He and Thiemo Voigt. Droplet: A new denial-of-service attack on low power wireless sensor networks. In *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*, pages 542–550. IEEE, 2013.
- [5] Yin Wang, Yuan He, Xuwei Mao, Yunhao Liu, and Xiang-yang Li. Exploiting constructive interference for scalable flooding in wireless networks. *Networking, IEEE/ACM Transactions on*, 21(6):1880–1889, 2013.
- [6] Manjunath Doddavenkatappa, Mun Choon Chan, and Ben Leong. Splash: Fast data dissemination with constructive interference in wireless sensor networks. In *NSDI*, pages 269–282, 2013.
- [7] Shuo Guo, Liang He, Yu Gu, Bo Jiang, and Tian He. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. *Computers, IEEE Transactions on*, 63(11):2787–2802, 2014.
- [8] Wan Du, Jansen Christian Liando, Huanle Zhang, and Mo Li. When pipelines meet fountain: Fast data dissemination in wireless sensor networks. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 365–378. ACM, 2015.
- [9] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. Low-power wireless bus. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 1–14. ACM, 2012.
- [10] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proceedings of ACM/IEEE IPSN*, 2013.
- [11] Anthony D. Wood and John A. Stankovic. A Taxonomy for Denial-of-Service Attacks in Wireless Sensor Networks. In *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. 2004.
- [12] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '05*, pages 46–57. New York, NY, USA, 2005. ACM.
- [13] Zequ Yang, Peng Cheng, and Jiming Chen. Learjam: An energy-efficient learning-based jamming attack against low-duty-cycle networks. In *IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 354–362. IEEE, 2014.
- [14] Marco Tiloca, Domenico De Guglielmo, Gianluca Dini, Giuseppe Anastasi, and Sajal Das. JAMMY: a Distributed and Self-Adaptive Solution against Selective Jamming Attack in TDMA WSNs. *accepted for IEEE Transactions on Dependable and Secure Computing*.
- [15] Mingyan Li, Iordanis Koutsopoulos, and Radha Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *26th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1307–1315. IEEE, 2007.
- [16] Yee Wei Law, Lodewijk Van Hoesel, Jeroen Doumen, Pieter Hartel, and Paul Havinga. Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, pages 76–88. ACM, 2005.
- [17] Matthias Wilhelm, Ivan Martinovic, Jens B. Schmitt, and Vincent Lenders. Short paper: Reactive jamming in wireless networks: How realistic is the threat? In *Proceedings of the Fourth ACM Conference on Wireless Network Security*, pages 47–52. ACM, 2011.
- [18] Guobin Liu, Jiaqing Luo, Qingjun Xiao, and Bin Xiao. Edjam: Effective dynamic jamming against ieee 802.15. 4-compliant wireless personal area networks. In *IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2011.